Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

РАДЧЕНКО КОСТЯНТИН ОЛЕКСАНДРОВИЧ

УДК 004.75:004.94

ДИСЕРТАЦІЯ

МЕТОДИ, МОДЕЛІ ТА ЗАСОБИ ПРОГНОЗУВАННЯ НАВАНТАЖЕННЯ НА ВЕБСЕРВЕР

Шифр і назва спеціальності 123 – Комп’ютерна інженерія
Галузь знань 12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ К.О. Радченко

Науковий керівник: Терейковський Ігор Анатолійович, д.т.н., проф.

Київ – 2025

АНОТАЦІЯ

Радченко К.О. Методи, моделі та засоби прогнозування навантаження на вебсервер. – Кваліфікаційна праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії з галузі знань 12 Інформаційні технології за спеціальністю 123 Комп'ютерна інженерія. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2025.

Дисертація присвячена розв'язанню актуальної науково-практичної задачі підвищення ефективності процесу прогнозування навантаження на вебсервер. В результаті проведеного аналізу була визначена перспективність вдосконалення шаблонів нормальної поведінки вебсерверів шляхом впровадження в їх математичне забезпечення сучасних методів частотно-часового аналізу сигналів на базі теорії вейвлет-перетворень. Було досліджено методи прогнозування навантаження на вебсервер з використанням вейвлет-перетворень у поєднанні з нейронними мережами.

Актуальність теми обумовлена необхідністю забезпечення стабільної роботи вебресурсів в умовах нестабільного трафіку та різких пікових навантажень. Проведено аналіз теоретичних основ вейвлет-аналізу, класифікацію базисних функцій і розроблено критерії оцінки їх ефективності в задачах часових рядів. Запропоновано інтегровану модель, в якій вейвлет-декомпозиція застосовується для попередньої обробки вхідного сигналу, що дозволяє виокремити значущі компоненти навантаження на різних рівнях масштабу. Розроблено методику вибору оптимального порядку вейвлет-декомпозиції, що забезпечує компроміс між глибиною аналізу та збереженням корисної інформації.

На відміну від методів Фур'є, вейвлети дозволяють локалізувати характеристики сигналу як у часовій, так і у частотній області, що дає змогу виділяти тренди, піки навантаження, флуктуації та періодичні аномалії. Проте в поточному науковому дискурсі мало вивченими залишаються питання вибору оптимального вейвлет-базису для задач такого типу, побудови комбінованих

моделей, а також критеріїв оцінювання якості таких моделей у порівнянні з класичними підходами. У зв’язку з цим, мета даного дослідження полягає у підвищенні ефективності прогнозування навантаження на вебсервери шляхом розробки вейвлет-орієнтованих моделей і програмних засобів, а також демонстрації їх переваг у порівнянні з традиційними методами.

Було виконано поглиблений аналіз та формалізацію завдання прогнозування навантаження на вебсервери загального призначення з урахуванням особливостей їхньої роботи в комп’ютерних мережах. У процесі дослідження було уточнено предметну область та визначено комплекс ключових експлуатаційних характеристик вебсервера, що мають безпосередній вплив на рівень його навантаження. До таких параметрів належать кількість запитів у черзі, середній час відповіді, швидкість обробки запитів та інші метрики, що відображають стан і продуктивність системи. Для формалізації цих характеристик запропоновано опис області працездатності вебсервера через множину контрольованих параметрів з чітко встановленими граничними значеннями. Вихід будь-якого параметра за межі допустимого діапазону розглядається як ознака перевантаження та потенційного погіршення якості обслуговування.

На основі отриманих даних розроблено концептуальну модель прогнозування навантаження, яка описує взаємозв’язок між динамікою параметрів вебтрафіку та прогнозними оцінками майбутніх станів системи. Ключовою особливістю моделі є автоматичний вибір типу базисного вейвлету залежно від характеру вхідного сигналу, що дозволяє підвищити ефективність аналізу. Обґрунтовано застосування вейвлет-перетворень для багатомасштабного аналізу даних, який дає змогу одночасно виявляти довгострокові тенденції та короткочасні пікові зміни у вебтрафіку, що є критично важливим для своєчасного реагування на зміни навантаження. Методологія моделі передбачає використання процедур фільтрації та попередньої обробки даних перед виконанням прогнозування, що зменшує вплив шумів та випадкових коливань на точність результатів. Визначено критерії ефективності вибору типу базисного вейвлету: точність відтворення сигналу, швидкість обчислень та стійкість до завад.

Запропонована концептуальна модель формує основу для побудови гібридних систем прогнозування, зокрема рішень на базі Wavelet+LSTM, здатних зменшити час аналізу вебтрафіку та підвищити точність прогнозів завдяки адаптивному вибору базису. Гнучкість моделі забезпечує її сумісність із наявними системами моніторингу та управління вебресурсами без необхідності суттєвих змін в архітектурі. поетапно реалізовано модель прогнозування навантаження на вебсервер, яка охоплює збір логів та метрик, їх попередню обробку, вейвлет-декомпозицію, моделювання та подальшу інтеграцію результатів у робоче середовище. Такий підхід дозволяє ефективно розділяти сигнал на трендові та коливальні компоненти, що підвищує точність прогнозування як довготривалих змін, так і короткочасних пікових навантажень.

Концептуальна модель процесу прогнозування навантаження на вебсервер являє собою абстрактне подання процесу, яке описує основні сутності, взаємозв’язки та інформаційні потоки між компонентами системи прогнозування. У межах цієї моделі визначаються ключові етапи – збір даних із серверних логів, попередня обробка сигналів, вейвлет-декомпозиція, формування ознак, навчання прогнозувальної нейронної мережі та оцінювання точності. Концептуальна модель відображає логічну структуру процесу, дозволяючи зрозуміти що саме виконує система та як пов’язані її функціональні блоки.

Запропонована інтегрована модель процесів прогнозування навантаження на вебсервер являє собою об’єднання функціональних, аналітичних і алгоритмічних компонентів у взаємопов’язаний повний цикл прогнозування – від збору даних до прийняття рішень. Інтегрована модель реалізує взаємодію модулів моніторингу, вейвлет-аналізу, нейромережевого прогнозування, оптимізації параметрів та візуалізації результатів. Вона забезпечує цілісну інтеграцію інформаційних, аналітичних і керувальних процесів у єдиному середовищі, що дає змогу точніше формувати прогнози вебтрафіку й автоматично адаптувати конфігурацію серверних ресурсів під змінні умови роботи вебсистеми.

Прогнозування навантаження на вебсервер зазвичай розглядається як задача часових рядів, де майбутні значення метрик навантаження передбачаються на

основі історичних даних. Особливість цієї задачі полягає у високій динамічності середовища, сезонності запитів, наявності пікових навантажень та неочікуваних стрибків, зумовлених зовнішніми факторами.

Застосування вейвлет-аналізу дало змогу покращити роботу з нестаціонарними часовими рядами, адже він забезпечує високу деталізацію у часово-частотному просторі. Поєднання статистичних методів із глибинними нейронними моделями дозволило отримати більш збалансовані результати та знизити похибку прогнозів. Порівняльний аналіз показав перевагу інтегрованої вейвлет-моделі над традиційними підходами, такими як ARIMA чи класичні нейромережі. Практична цінність моделі полягає у можливості завчасно виявляти перевантаження та запобігати їм завдяки автоматичному масштабуванню, оптимізації використання обчислювальних ресурсів та інтеграції з інструментами моніторингу.

Розроблена модель може застосовуватись адміністраторами вебресурсів як у тестових середовищах для генерації синтетичних навантажень, так і у реальних умовах для підтримання стабільності роботи серверів, здатних підлаштовуватися під динаміку трафіку, у використанні більших наборів даних для навчання та в аналізі зовнішніх факторів (сезонність, рекламні кампанії, події). Це відкриває перспективи для ще більш точного прогнозування та гнучкого управління вебсерверами у змінному середовищі.

Запропоновано метод визначення ефективного типу базисного вейвлету, що призначений для розробки шаблону нормальної поведінки. Для цього обґрунтовано ряд положень та перелік критеріїв ефективності, які дозволяють забезпечити вибір ефективного типу базисного вейвлету у відповідності до значимих вимог задачі розробки означеного шаблону. Суть методу визначення найефективнішого типу материнського вейвлету полягає в систематичному багатокритеріальному оцінюванні вейвлет-функцій з урахуванням їхніх математичних, спектральних і структурних властивостей щодо характеру досліджуваного сигналу навантаження вебсервера. Метод передбачає порівняння вейвлетів-кандидатів за рядом кількісних і якісних критеріїв, що відображають придатність вейвлету для часово-

частотного аналізу трафіку.

На основі результатів вибору оптимального базису також розроблено метод прогнозування навантаження на вебсервер, який поєднує вейвлет-декомпозицію з нейромережевим прогнозуванням. Застосування обраного материнського вейвлету забезпечує ефективне виділення трендових і високочастотних компонентів трафіку, що, у свою чергу, підвищує адаптивність моделі до змін середовища та знижує середньоквадратичну похибку прогнозу.

Суть методу прогнозування навантаження на вебсервер полягає у використанні інтегрованого поєднання часово-частотного аналізу сигналу за допомогою вейвлет-перетворення з глибинним нейронним прогнозуванням, що дозволяє з високою точністю передбачати зміну інтенсивності вебзапитів і забезпечувати оптимальне використання обчислювальних ресурсів серверної інфраструктури. здійснено огляд основних підходів до оцінки показників навантаження вебсерверів, що зводиться до пошуку ефективного компромісу між вимогами до безпеки та ресурсами, необхідними для їхнього забезпечення.

Аналіз експериментальних результатів підтверджує, що комбінування вейвлет-фільтрації з подальшим прогнозуванням дозволяє досягати прийнятної точності на різних часових масштабах, а також забезпечує наочну інтерпретацію динаміки навантаження, що важливо для оперативного та стратегічного управління ресурсами вебсерверу. Однією з ключових переваг розробленої системи є можливість автоматизованого підбору оптимального вейвлету без потреби у тривалих обчислювальних експериментах, що суттєво скорочує час побудови моделі та підвищує ефективність процесу прогнозування.

Ключові слова: нейронна мережа, дані, вебсервер, трафік, навантаження, моделювання, метод, прогнозування, система, тестування, модульність, цифровий двійник, вимірювання, корекція і контроль помилок.

ABSTRACT

Radchenko K.O. Methods, models and tools for predicting web server load. – Qualifying scientific work, the manuscript.

Ph.D. thesis in the field of knowledge 12 Information technologies in a specialty 123 Computer engineering. – National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2025.

The thesis is devoted to solving the current scientific and practical task of improving the efficiency of the process of web server load forecasting. As a result of the conducted analysis, the feasibility of enhancing normal-behavior patterns of web servers through the integration of modern time–frequency signal analysis methods based on wavelet transform theory into their mathematical support was identified. Methods for forecasting web server load using wavelet transforms in combination with neural networks were thoroughly examined.

The relevance of the topic is driven by the need to ensure the stable operation of web resources under conditions of unstable traffic and sudden peak loads. A detailed analysis of the theoretical foundations of wavelet analysis was conducted, including a classification of basis functions and the development of criteria for evaluating their effectiveness in time series tasks. An integrated model was proposed in which wavelet decomposition is used for preprocessing the input signal, enabling the extraction of significant load components at different scale levels. A methodology for selecting the optimal order of wavelet decomposition was developed, ensuring a balance between the depth of analysis and the preservation of useful information.

Unlike Fourier-based methods, wavelets allow for the localization of signal characteristics in both time and frequency domains, making it possible to identify trends, load spikes, fluctuations, and periodic anomalies. However, in current scientific discourse, issues related to selecting an optimal wavelet basis for such tasks, designing combined models, and defining evaluation criteria for comparing such models with classical approaches remain insufficiently studied. Therefore, the aim of this research is to enhance the effectiveness of web server load forecasting through the development of

wavelet-oriented models and software tools, as well as to demonstrate their advantages over traditional methods.

A detailed analysis and formalization of the task of forecasting the load on general-purpose web servers was carried out, taking into account the specifics of their behavior within computer networks. The main emphasis was placed on developing a conceptual forecasting model, which became a logical continuation and specified the components and parameters necessary for the practical implementation of a forecasting system. During the research, the problem domain was refined and a set of key operational characteristics affecting the level of web server load was defined. These parameters include queue length, average response time, request processing speed, and other metrics that reflect system state and performance. To formalize these characteristics, the operational region of a web server was described as a set of controlled parameters with clearly defined boundary values. Any deviation of such parameters beyond the allowed range is considered an indicator of overload and potential service degradation.

Based on the obtained data, a conceptual load-forecasting model was developed, describing the relationship between the dynamics of web traffic parameters and the predicted future states of the system. A key feature of the model is the automatic selection of a wavelet basis type depending on the characteristics of the input signal, which improves analysis efficiency. The use of wavelet transforms for multiscale data analysis was justified, as it enables the simultaneous detection of long-term trends and short-term peak changes in web traffic – a feature that is critically important for timely response to load fluctuations. The methodology includes filtering and preprocessing procedures prior to forecasting, which reduce the influence of noise and random fluctuations on prediction accuracy. Efficiency criteria for wavelet basis selection were defined: signal reconstruction accuracy, computational speed, and noise robustness. The obtained results are valuable for further research and practical applications.

The proposed conceptual model forms the foundation for building hybrid forecasting systems, particularly Wavelet+LSTM-based solutions capable of reducing traffic-analysis time and improving forecast accuracy through adaptive basis selection. The flexibility of the model ensures compatibility with existing web-resource monitoring

and management systems without requiring significant architectural changes. A step-by-step implementation of a web server load-forecasting model was carried out, covering log and metric collection, preprocessing, wavelet decomposition, modeling, and integration of results into the working environment. This approach effectively separates the signal into trend and oscillatory components, thus improving the accuracy of both long-term and short-term peak load forecasting.

The conceptual model of the web server load-forecasting process represents an abstract depiction of the system, describing the core entities, relationships, and information flows between its components. The model defines the key stages: data collection from server logs, preprocessing, wavelet decomposition, feature generation, neural-network training, and accuracy evaluation. It reflects the logical structure of the process, allowing one to understand what the system performs and how its functional blocks interact.

The proposed integrated forecasting model combines functional, analytical, and algorithmic components into a unified prediction cycle – from data collection to decision-making. The integrated model ensures the interaction of monitoring modules, wavelet analysis, neural-network forecasting, parameter optimization, and result visualization. It provides seamless integration of information, analytical, and control processes within a single environment, enabling more accurate traffic forecasts and automatic adaptation of server-resource configurations to changing operational conditions.

Web server load forecasting is typically addressed as a time-series problem, where future metric values (number of requests, CPU usage, response latency, etc.) are predicted based on historical data. The complexity of this task stems from high environmental dynamics, request seasonality, peak loads, and unexpected spikes caused by external factors (e.g., marketing campaigns, system updates).

Wavelet analysis enabled improved handling of nonstationary time series by providing high resolution in the time–frequency domain. Combining statistical methods with deep neural models resulted in more balanced outcomes and reduced prediction error. Comparative analysis demonstrated the superiority of the integrated wavelet-based model over traditional approaches such as ARIMA or classical neural networks. The

practical value of the model lies in its ability to detect overloads in advance and prevent them through automatic scaling, resource optimization, and integration with monitoring tools.

The developed model can be used by web-resource administrators both in testing environments for generating synthetic loads and under real conditions to ensure stable server operation capable of adapting to traffic dynamics, large training datasets, and external factors (seasonality, ad campaigns, events). This opens new prospects for more accurate forecasting and flexible web-server management in a dynamic environment.

A method for determining the most effective wavelet basis type was proposed, intended for developing a normal-behavior pattern. A set of principles and efficiency criteria was justified to support the selection of an appropriate wavelet basis according to the requirements of the task. The essence of the method lies in systematic multi-criteria evaluation of wavelet functions with respect to the mathematical, spectral, and structural properties of the web-server load signal. The method involves comparing candidate wavelets (Daubechies, Symlet, Coiflet, Biorthogonal, Meyer, Haar, etc.) based on several quantitative and qualitative criteria reflecting their suitability for time–frequency analysis of web traffic.

Based on the optimal basis selection results, a load-forecasting method was developed that combines wavelet decomposition with neural-network prediction. Using an appropriate mother wavelet enables effective extraction of trend and high-frequency components of traffic, improving model adaptability and reducing mean squared error.

The essence of the proposed forecasting method is the integrated use of time–frequency analysis via wavelet transforms together with deep-learning-based prediction, enabling accurate estimation of incoming request intensity and ensuring optimal utilization of server computing resources. A review of the main approaches to evaluating web server load metrics was conducted, highlighting the need to balance security requirements and the computational resources necessary to maintain system performance.

Analysis of the experimental results confirms that combining wavelet filtering with subsequent neural forecasting yields acceptable accuracy across multiple time scales, and

also provides clear interpretability of load dynamics, which is essential for both operational and strategic resource management. One of the key advantages of the developed system is the capability for automated optimal wavelet selection without requiring extensive computational experiments, significantly reducing model-building time and improving forecasting efficiency.

Key words: neural network, data, web server, traffic, load, modeling, method, forecasting, system, testing, modularity, digital twin, measurement, error correction and control.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Radchenko K. Applying Wavelet Transforms for Web Server Load Forecasting / Z. Hu, I. Tereykovskiy, L. Tereykovska, M. Tsiutsiura, K. Radchenko // *In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer, Cham*. https://doi.org/10.1007/978-3-030-16621-2_2 (Закордонне видання, реферується наукометричною базою «Scopus»). *Здобувач обґрунтував переваги та перспективи використання вейвлет-перетворень при прогнозуванні навантаження на вебсервер, виконав аналітичний огляд методів і підготував експериментальні приклади застосування; Z. Hu долучився до формування загальної структури статті; I. Tereykovskiy брав участь у перевірці коректності обраних базисних вейвлетів; L. Tereykovska проаналізувала властивості вебтрафіку та їх відповідність модельним припущенням; M. Tsiutsiura допоміг у формуванні архітектури програмного експерименту та узагальненні результатів.*

2. Радченко К. О. Концептуальна модель забезпечення ефективності прогнозування навантаження на вебсервер / К. О. Радченко // *Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія : Технічні науки*. – 2020. – Т. 31 (70), № 6. – С. 135–141. https://doi.org/10.32838/TNU-2663-5941/2020.6-1/23 (Фахове наукове видання України категорії Б). *Здобувач представив концептуальну модель ефективного прогнозування навантаження на вебсервер, визначив модель параметрів вебтрафіку.*

3. Радченко К.О. Застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер комп’ютерних мереж загального призначення / К.О. Радченко // *Комп’ютерно-інтегровані технології: освіта, наука, виробництво*. – Луцьк. – № 45. – 2021. – С. 90 – 96. https://doi.org/10.36910/6775-2524-0560-2021-45-13 (Фахове наукове видання України категорії Б). *Здобувач дослідив можливості застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер*.

4. Радченко К.О. Особливості прогнозування рівня вебтрафіку у комп'ютерних мережах загального призначення / К.О. Радченко // *Проблеми інформатизації та управління*. – Національний авіаційний університет – № 3(71). – 2022. – С. 41 – 50. https://doi.org/10.18372/2073-4751.71.17002 (Фахове наукове видання України категорії Б). *Здобувач проаналізував особливості прогнозування рівня вебтрафіку*.

5. Дичка, І., Радченко, К., Терейковський, І., & Терейковська, Л. (2024). Концептуальна модель процесу прогнозування навантаження на вебсервер. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (54), 74-83. https://doi.org/10.36910/6775-2524-0560-2024-54-09 (Фахове наукове видання України категорії Б). *Здобувач сформулював основи концептуальної моделі прогнозування навантаження на вебсервер, визначив функціональну структуру системи, узагальнив методи обробки часових рядів; І. Дичка забезпечив загальне методологічне керівництво дослідженням; І. Терейковський перевірив логічну узгодженість моделі; Л. Терейковська оцінила відповідність моделі реальним умовам функціонування вебсервера*.

6. Радченко, К., & Романенко, М. (2024). Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (56), 261-268. https://doi.org/10.36910/6775-2524-0560-2024-56-33 (Фахове наукове видання України категорії Б). *Здобувач дослідив особливості попередньої обробки часових рядів за допомогою дискретного вейвлет-перетворення, розробив структуру гібридної моделі; М. Романенко виконала аналіз отриманих результатів*.

7. K. Radchenko and I. Tereykovskyi, "Integrated Neural Network and Wavelet-Based Model for Web Server Load Forecasting", *SISIOT (Security of Infocommunication Systems and Internet of Things)*, vol. 2, no. 2, p. 02006, Dec. 2024, https://doi.org/10.31861/sisiot2024.2.02006 (Фахове наукове видання України категорії Б). *Здобувач розробив інтегральну модель прогнозування навантаження на вебсервер, провів експериментальне моделювання; І. Терейковський допоміг у валідації результатів та їх формальному описі*.

8. Радченко, К. О., & Терейковський, І. А. (2025). Метод прогнозування навантаження на вебсервер з урахуванням вейвлет-аналізу веблогів та вебтрафіку. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (60), 247-259. https://doi.org/10.36910/6775-2524-0560-2025-60-27 (Фахове наукове видання України категорії Б). *Здобувач презентував удосконалений метод прогнозування навантаження на вебсервер, виконав експериментальну оцінку точності; І. Терейковський долучився до аналізу результатів.*

9. Radchenko K. O. The application of discrete wavelet transform for forecasting the operational load of the Web server of the distance education system / L. A. Tereikovska, K. O. Radchenko, K. O. Kharlamov. // *The Eighth World Congress "Aviation in the XXI-st Century" - «Safety in Aviation And Space Technologies»*. – Kyiv Aviation University 10.10 – 12.10.2018. – Pp. 3.2.13 – 3.2.17. *Здобувач презентував обґрунтування використання дискретного вейвлет-перетворення при обробці часових рядів навантаженості вебсервера*.

10. Радченко К. О. Особливості архітектури нейромережевої моделі розпізнавання кібератак / К. О. Радченко. // *Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах: Матеріали VII Міжнародної науково-практична конференції*. – Чернівці: «Місто». – 2018. С. 149 – 150. *Здобувач презентував особливості архітектури нейромережевої моделі під час обробки часових рядів навантаження на вебсервер*.

11. Радченко К.О. Концепція прогнозування навантаженості Вебсерверів за допомогою вейвлет-перетворень / К.О. Радченко, О.І. Терейковський, К.О. Харламов. // *Матеріали науково-практичної конференції «Сучасні інформаційні технології та кібербезпека»*. – К.: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. – С. 205 – 207. *Здобувач презентував концепцію прогнозування навантаженості вебсерверу з використанням вейвлет-перетворень, виконав підбір методів; Терейковський О.І. здійснив практичний аналіз стійкості та придатності вейвлет-базисів для задач обробки вебтрафіку; Харламов К.О. виконав порівняння характеристик різних типів вебнавантаження та їх впливу на роботу серверів*.

12. Radchenko K. Wavelet transforms for web server load calculating / Kostiantyn Radchenko, Oleh Tereikovskyi // *ITSec: Безпека інформаційних технологій: IX міжнародна науково-технічна конференція*, 22-27 березня 2019 р. – К.: НАУ, 2019. – pp. 28 – 29. *Здобувач презентував особливості застосування вейвлет-перетворень під час аналізу рівня навантаженості вебсервера; O. Tereikovskyi здійснив математичний аналіз точності та стійкості перетворення*.

13. Радченко К.О. Аналіз самоподібності рівня вебтрафіку у комп'ютерних мережах загального призначення / К.О. Радченко, Л. О. Терейковська // *Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 25 листопада 2022 [Електронний ресурс]. – К: НУХТ, 2022. – С. 112 – 113. – Режим доступу: https://nuft.edu.ua/naukova-diyalnist/naukovi-konferencii. *Здобувач проаналізував рівень самоподібності вебтрафіку; Терейковська Л.О. надала методичний супровід щодо методів оцінювання самоподібності, виконала оцінювання коректності висновків.*

14. Стіренко С.Г., Радченко К.О., Апанасенко В.П. Прогнозування рівня навантаження на вебсервер комп'ютерних мереж загального призначення // *Кібербезпека державних інституцій та подолання кризових станів: матеріали I Міжнародної науково-практичної конференції* (Київ – Вроцлав, 26 травня 2022 р.). Київ, 2022. – С. 59 – 60. *Здобувач презентував концепцію використання вейвлет-перетворень під час прогнозування навантаження на вебсервер; Стіренко С.Г. визначив системні вимоги до моделі; Апанасенко В.П. виконав експериментальні дослідження*.

15. Радченко К.О. Прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень Добеші // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІРТК*-2023). *Шістнадцята міжнародна науково-практична конференція* 23 – 24 *травня* 2023 *р.,* Київ, Україна. – К.: НАУ, 2023. – С. 335 – 336. *Здобувач презентував прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень*.

16. Радченко К.О. Виявлення різких змін рівня вебтрафіку за допомогою вейвлет-перетворень Добеші // *Матеріали II Міжнародної науково-практичної конференції "Кібербезпека державних інституцій та подолання кризових станів" : в 2 т.* Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2023. Т. 1. С. 190. *Здобувач презентував визначення пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень*.

17. Kostiantyn Radchenko, Xing Tao. Design and implementation of an automated Big Data analysis module // *Матеріали XI Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 27 листопада 2024 [Електронний ресурс]. – К: НУХТ, 2024. С. 40 – 41. *Здобувач презентував особливості впровадження модульного підходу під час обробки великих масивів даних часових рядів; Xing Tao розробив архітектуру автоматизованого модуля Big Data analytics*.

18. K. Radchenko, I. Tereykovskyi, Xing Tao. Automated Big Data Analysis Module // *XVII Науково-практична конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг"* (*ПМК*-2024) (м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 20-22 листопада 2024 р.), С. 239 – 245. *Здобувач презентував архітектуру модульного підходу під час обробки великих масивів даних часових рядів; I. Tereykovskyi розробив логіку даних та оптимізацію методів фільтрації; Xing Tao здійснив технічну реалізацію прототипу*.

19. Романкевич В.О., Радченко К.О., Романенко М.В. Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж // *XVII Науково-практична конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг"* (*ПМК*-2024) (м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 20-22 листопада 2024 р.), С. 424 – 428. *Здобувач реалізував гібридний підхід до прогнозування часових рядів з використанням вейвлет-перетворення та нейронних мереж; Романкевич В.О. розробив постановку задачі; Романенко М.В. виконала навчання моделі*.

20. Радченко К.О., Статечний С.В. Адаптивні інтелектуальні системи для динамічного керування навантаженням вебсерверів // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІІРТК*-2025). *Вісімнадцята міжнародна науково-практична конференція* 20-21 *травня* 2025 *р*., Київ, Україна. – К.: КАІ, 2025. – С. 375 – 377. *Здобувач розробив адаптивну інформаційну систему для динамічного керування навантаженням вебсерверів; Статечний С.В. допоміг з валідацією роботи системи у прикладних сценаріях.*

21. Радченко К.О., Черненький А. О. Прогнозування вебнавантаження для підвищення надійності та доступності обчислювальних систем // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІІРТК*-2025). *Вісімнадцята міжнародна науково-практична конференція* 20-21 *травня* 2025 *р.*, Київ, Україна. – К.: КАІ, 2025. – С. 378 – 380. *Здобувач презентував метод прогнозування навантаження на вебсервери, що забезпечує превентивне балансування ресурсів для підтримки стабільної роботи обчислювальної інфраструктури; Черненький А.О. виконав експериментальне моделювання.*

22. Радченко К.О. Комп'ютерна програма «Wavelet analysis» / Терейковський Ігор Анатолійович; Радченко Костянтин Олександрович; Дичка Іван Андрійович; Романкевич Віталій Олексійович; Тарасенко-Клятченко Оксана Володимирівна – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» // Свідоцтво про реєстрацію авторського права на твір №7089 від 15.04.2024р. https://sis.nipo.gov.ua/en/search/detail/1803492/ Опубліковано 31.05.2024, бюл. №81. *Здобувач розробив модулі, що забезпечують автоматизоване прогнозування навантаження на вебсервери за допомогою гібридної моделі Wavelet Neural Network (WNN) та Long Short-Term Memory (LSTM), забезпечив тестування; Терейковський І. А. надав методичні рекомендації щодо побудови архітектури програмного забезпечення; Дичка І. А. структурував методи обробки сигналів та результати тестування; Романкевич В. О. структурував технічні характеристики, алгоритми роботи програми; Тарасенко-Клятченко О. В. здійснювала технічне та методичне редагування документації.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI (Artificial Intelligence, Штучний інтелект) – це галузь комп'ютерних наук, яка зосереджена на створенні машин та програм, що можуть виконувати завдання, які зазвичай вимагають людського інтелекту. Основною метою AI є створення систем, які можуть імітувати людську поведінку, таку як розуміння, навчання, планування, рішення, мова і навіть креативність.

CNN (Convolutional Neural Networks, Згорткові нейронні мережі) – це клас штучних нейронних мереж, спеціально розроблений для ефективної обробки даних, що мають просторову структуру, як-от спектрограми або будь-які інші двочи багатовимірні сигнали.

CSM (Content Switching Module) – це модуль керування перемиканням контенту, що використовується в балансуванні навантаження між серверами у кластерній інфраструктурі. Його основна функція – ефективний розподіл запитів користувачів, перенаправляючи їх на найбільш відповідний сервер, що мінімізує затримки та підвищує ефективність роботи системи.

HTTP (HyperText Transfer Protocol) — це протокол прикладного рівня, який використовується для передачі гіпертекстових даних (тексту, зображень, відео, скриптів тощо) між веб-клієнтом (наприклад, браузером) і веб-сервером.

Integrated Wavelet Model for Web Traffic Load Prediction (IWMLP) – експериментальна установка, розроблена в рамках реалізації дисертаційного дослідження. Назва передає суть – поєднання (integrated) різних підходів у межах моделі, вказує на використання вейвлетів (wavelet model), прогнозування навантаження (load prediction) у контексті вебтрафіку (web traffic).

Long Short-Term Memory (LSTM) – тип рекурентної нейронної мережі (RNN), який здатен запам’ятовувати довгострокові залежності у часових рядах.

MAE (Mean Absolute Error, середнє абсолютне відхилення) – визначає середнє відхилення прогнозного значення від фактичного.

MAPE (Mean Absolute Percentage Error, середня абсолютна відносна помилка) – дозволяє оцінювати точність прогнозів незалежно від масштабу навантаження.

ML (Machine Learning, машинне навчання) є підгалуззю штучного інтелекту, яка зосереджена на розробці алгоритмів, які дозволяють комп'ютерам вивчати з досвіду і робити прогнози або приймати рішення без явного програмування. Іншими словами, ML використовує дані для створення моделей, які можуть автоматично покращуватися з часом.

RMSE (Root Mean Square Error, середньоквадратичне кореневе відхилення) – корінь із середньоквадратичної похибки, підкреслює вплив значних відхилень.

TCP/IP (Transmission Control Protocol / Internet Protocol) – протокол керування передаванням у мережі.

Wavelet Neural Network (WNN) – це нейронна мережа, яка використовує вейвлет-функції як активаційні функції або для попередньої обробки сигналу.

Багатокритеріальне прийняття рішень (MCDM, Multiple-Criteria Decision Making) – це галузь науки про прийняття рішень, яка вивчає методи вибору або ранжування альтернатив, коли оцінка кожної альтернативи залежить одночасно від кількох, часто конфліктних критеріїв.

Базисний вейвлет (basis wavelet) – це конкретний вайвлет, що утворює базис для простору функцій, у якому проводиться розкладання. У контексті дискретних вейвлет-перетворень, базисний вайвлет є елементом базису, який дозволяє відновити функцію після її розкладу на хвилі з відповідними масштабами та зсувами.

Вейвлет-перетворення (Wavelet Transform, WT, ВП) є інструментом для аналізу сигналів і даних, особливо коли потрібна локалізація змін у часі або частоті. Вейвлет-перетворення дозволяє представляти сигнал на різних масштабах та з різною роздільною здатністю, що робить його ефективним для обробки сигналів з нестаціонарними властивостями.

Вебсервер загального призначення (ВЗП) – це серверне програмне забезпечення або апаратно-програмний комплекс, призначений для обробки запитів HTTP(S) від клієнтів (веббраузерів) та надання у відповідь вебресурсів (HTML-сторінок, зображень, скриптів тощо) незалежно від предметної галузі чи специфіки використання.

Дискретне вейвлет-перетворення (Discrete Wavelet Transform, DWT) – це математичний інструмент аналізу сигналів, який дозволяє поділити сигнал на компоненти з різною частотністю та часовою локалізацією. На відміну від традиційного перетворення Фур’є, DWT забезпечує як частотний, так і часовий опис сигналу, що робить його особливо ефективним для обробки нестаціонарних сигналів, до яких належать, наприклад, вебтрафік чи часові ряди навантаження на сервер.

Комп'ютерна мережа загального призначення (КМЗП) – це система, яка забезпечує об'єднання комп'ютерів та інших пристроїв для спільного використання ресурсів, обміну даними та доступу до різноманітних мережевих сервісів.

Материнський вейвлет (mother wavelet) – це основна функція, з якої генеруються всі інші вайвлети через масштабування (dilatation) і зсув (translation). Це базова форма, яка використовується для побудови конкретних варіантів вайвлетів в процесі перетворення.

Неперервне вейвлет-перетворення (Continuous Wavelet Transform, CWT) – це математичний інструмент аналізу сигналів, який дозволяє отримати детальний часо-частотний опис сигналу з високою точністю. На відміну від дискретного вейвлет-перетворення, неперервне не потребує субдискретизації, що забезпечує повніше охоплення інформації про сигнал.

Стаціонарне вейвлет-перетворення (SWT) використовується для аналізу сигналів без необхідності зсуву або масштабу, що зберігає точність аналізу для стаціонарних сигналів. У SWT використовуються однакові параметри масштабу для всіх рівнів, що дозволяє уникнути помилок, пов'язаних зі зміщенням сигналу.

ВСТУП

Актуальність теми. У зв'язку з інтенсивним зростанням розподілених мережевих комп'ютерних систем, упровадженням таких систем в різні галузі людської діяльності проблема забезпечення надійності їх функціонування стає дедалі важливою і актуальною. Майже всі сучасні мережеві комп'ютерні системи містять у собі один або кілька серверів, що забезпечують інтеграцію з глобальною комп'ютерною мережею Інтернет. Типові функції вебсерверів включають надання послуг: веб, FTP, електронну пошту тощо. Практичний досвід свідчить про неодноразові порушення працездатності програмного забезпечення вебсервера, як внаслідок надмірного навантаження на сервер, так і внаслідок успішної атаки типу «відмова в обслуговуванні». Очевидно, що запобігання зазначеним порушенням тісно пов'язане з розробкою ефективних інструментів прогнозування навантаженості вебсерверів. При цьому на думку дослідників [5] підвищити ефективність таких інструментів не можливо без розробки нових методів і моделей, на основі яких можна формувати досить достовірне прогнозування значень експлуатаційних параметрів. Зазначимо, що отримуваний прогноз є базою як для планування роботи мережевих ресурсів, так і для визначення дій оперативного управління над ними. Таким чином, ефективність функціонування комп'ютерних мереж безпосередньо залежить від достовірності методів і моделей прогнозування експлуатаційних параметрів.

При цьому в більшості відомих системах захисту та системах моніторингу технічного стану вебсерверів засоби прогнозування не враховують складний нестаціонарний характер навантаження, що призводить як до можливого вичерпання обчислювальних ресурсів під час пікових навантажень, так і до надлишкого резервування вказаних ресурсів в періоди стабілізації. Крім того, відомі системи прогнозування мають високу вартість та закритий характер, що ускладнює їх інтеграцію в вітчизняні комп'ютерні мережі та системи. В такій постановці є актуальною задача розробки моделей, методів та засобів, що дозволяють підвищити точність прогнозування навантаження на вебсервер.

Дослідження вітчизняних та зарубіжних вчених Бодянського Є. В., Гусєва М. М., Козловського В. В., Різника О. М., Руденка О. Г., Д. Хінтона вказують на те, що перспективним шляхом вирішення задачі прогнозування навантаження на вебсервер є застосування теорії вейвлет-перетворень. Це пояснюється доведеною ефектвністю застосування теорії вейвлет-перетворень для вирішення подібних задач прогнозування складних нестаціонарних процесів в різних галузях науки та техніки. Методологічну і теоретичну основу для ефективного впровадження засобів на основі вейвлет-перетворень складають теоретичні розробки та досвід створення коп'ютерних систем та мереж таких науковців як Білощицького А. О., Бушуєва С. Д., Вінцюка Т. К., Зайцева В. Г., Міхайленка В. М., Подчасової Т. П., Рабінера Л. Р., Тарасенка В. П., Терейковського І. А., Терейковської Л. О., Шафера Р. В. тощо.

Таким чином, задача застосування вейвлет-перетворень для розробки ефективних методів, моделей та засобів прогнозування на вебсервер зумовила актуальність наукових досліджень і розробок, яким присвячена дисертаційна робота.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційні дослідження виконувалися на кафедрі системного програмування та спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» в рамках науково-дослідницьких держбюджетних робіт:

- «Методи, моделі та комп’ютерні засоби виявлення деструктивного впливу в медіапросторі», № держреєстрації 0121U110662 (з 2021р.);
- «Методи, моделі та засоби моніторингу закликів до тероризму у онлайн соціальних мережах», № держреєстрації 0124U003866 (з 2024р.).

Мета і завдання дослідження. *Метою роботи* є підвищення ефективності прогнозування навантаження на вебсервер, що функціонують у комп’ютерних мережах загального призначення, шляхом розробки математичних, алгоритмічних та програмних засобів багаторівневого прогнозування із використанням вейвлет-перетворень, нейронних мереж та аналітичного моделювання трафіку з

урахуванням коротко-, середньо- та довгострокових трендів трафіку на основі аналізу логів запитів. Додатково передбачається впровадження механізмів адаптивного масштабування і балансування ресурсів для забезпечення стабільності та доступності сервісу.

Основні задачі дослідження у відповідності до поставленої мети полягають в наступному:

- проаналізувати теоретичні основи вейвлет-перетворень, можливості їх застосування для часових рядів і прогнозування навантаження у вебсистемах, виявити обмеження й перспективи комбінування з нейронними моделями, сформувати методологічну основу побудови систем прогнозування у вигляді концентуальної моделі, у якій вейвлет-декомпозиція виконує роль інтелектуального фільтра для багаторівневого представлення сигналу та виділення різнострокових тенденцій;
- розробити інтегровану модель прогнозування навантаження, що поєднує дискретне вейвлет-перетворення та рекурентні нейронні мережі типу LSTM для зменшення похибки прогнозування й підвищення стійкості моделі до шуму;
- розробити метод визначення найефективнішого типу материнського вейвлету на основі формалізованих критеріїв ефективності (ортогональність, симетричність, компактність, масштабованість, інформативність), який дозволить уникнути довготривалих чисельних експериментів;
- розробити метод прогнозування навантаження на вебсервер, що враховує специфіку контролю параметрів у мережах загального призначення та представляє їх у вигляді багатоперіодичного ряду даних;
- провести експериментальні дослідження на основі реальних логів вебсерверів з метою оцінки точності прогнозування, стабільності результатів і впливу масштабу вейвлет-представлення на якість прогнозу, розробити програмне забезпечення для реалізації запропонованих методів прогнозування, що включає модулі вейвлет-декомпозиції, автоматичного

вибору характеристик сигналу, навчання моделей і візуалізації результатів.

Об'єктом дослідження є процеси прогнозування навантаження на вебсервер, що застосовується в комп'ютерних системах загального призначення.

Предметом дослідження є вейвлет-методи та вейвлет-моделі прогнозування навантаження на вебсервер, що застосовується в комп'ютерних системах загального призначення.

Методи дослідження. При виконанні досліджень у межах дисертаційної роботи було використано комплекс методів, що охоплюють математичні, статистичні та комп'ютерні підходи:

- методи теорії ймовірності та математичної статистики – застосовувалися для аналізу стохастичних характеристик навантаження на вебсервер, побудови статистичних моделей часових рядів, оцінювання якості прогнозування (зокрема, за показниками MAE, RMSE, MAPE) та для обґрунтування достовірності отриманих результатів;
- методи вейвлет-перетворень – використовувалися для багаторівневої декомпозиції сигналів вебнавантаження з метою виявлення прихованих трендів, шумів і характеристик різної частотної природи. Зокрема, реалізовано застосування дискретного вейвлет-перетворення (у тому числі алгоритму à trous) для підвищення інформативності вхідних даних при прогнозуванні;
- методи абстрагування, формалізації та математичного моделювання – дозволили розробити аналітичні моделі, що описують поведінку навантаження в часовому просторі, а також сформулювати новий підхід до інтеграції вейвлет-декомпозиції з нейронними мережами;
- методи комп'ютерного моделювання – використовувалися для побудови, навчання й тестування моделей прогнозування на основі Python, TensorFlow/Keras, MATLAB та інших засобів. За допомогою моделювання проведено серію експериментів на основі реальних логів вебсерверів, що дозволило оцінити ефективність запропонованих рішень у порівнянні з базовими підходами;

- методи візуалізації та порівняльного аналізу – застосовувалися для інтерпретації результатів, побудови графіків вейвлет-коефіцієнтів, прогнозів та похибок, а також для демонстрації переваг розробленої моделі над альтернативними.

Крім того, у процесі дослідження використовувалися засоби об'єктно-орієнтованого програмування та бібліотеки машинного навчання для реалізації експериментальних програмних прототипів.

Наукова новизна одержаних результатів визначається наступними положеннями:

- вперше розроблено модель прогнозування функціональних параметрів вебсерверу комп'ютерних мереж загального призначення, яка відрізняється від існуючих тим, що враховує особливості контролю функціональних параметрів вебсерверу; представляє функціональні параметри у вигляді багатоперіодичного ряду даних, з аналітичним апаратом фільтрації вказаних параметрів та забезпечує можливість розробки методу застосування вейвлет-перетворень для прогнозування навантаження на даний вебсервер;
- вперше розроблено метод вибору типу базисного вейвлету, який відрізняється від існуючих тим, що за рахунок застосування запропонованих критеріїв оцінки ефективності типів базисного вейвлету дозволяє уникнути довготривалих експериментів, пов'язаних з визначенням найбільш ефективного типу базисного вейвлету;
- отримала подальший розвиток концептуальна модель процесу прогнозування навантаження на вебсервер, що за рахунок розроблених критеріїв оцінки ефективності типів базисного вейвлету в рамках запропонованого методу вибору типу базисного вейвлету забезпечує можливість автоматичного визначення типу базисного вейвлету;
- отримав подальший розвиток метод застосування вейвлет-перетворень для прогнозування навантаження на вебсервер, який відрізняється від існуючих тим, що використовує розроблені моделі прогнозування

функціональних параметрів вебсерверу і розроблений метод вибору типу базисного вейвлету та забезпечує достатню точність прогнозування навантаження на вебсервер, який застосовується в комп'ютерних мережах загального призначення.

Практичне значення одержаних результатів дисертаційного дослідження полягає у наступному:

- розроблений метод вибору типу базисного вейвлету, дозволяє приблизно в 1,5 рази зменшити обчислювальні витрати, пов'язані з визначенням найбільш ефективного типу базисного вейвлету, що підтверджується актом впровадження в діяльність Федерації роботодавців ЖКГ України;
- розроблене на базі запропонованого методу застосування вейвлет-перетворень для прогнозування навантаження на вебсервер спеціалізоване програмне забезпечення може бути застосоване в організаціях, що займаються проєктуванням, експлуатацією та оцінкою показників надійності і захищеності комп'ютерних мереж загального призначення, що підтверджується актом впровадження в діяльність Федерації роботодавців ЖКГ України;
- розроблені програми, що реалізують запропоновані моделі та методи, використовуються на кафедрі системного програмування і спеціалізованих комп'ютерних систем КПІ ім. Ігоря Сікорського в курсі лекцій «Цифрова обробка сигналів та зображень», що підтверджується відповідним актом.

Особистий внесок здобувача. Всі результати, наведені в основній частині дисертаційної роботи, отримані здобувачем самостійно.

У наукових працях, опублікованих у співавторстві: [33] – у статті автором обґрунтовано переваги та перспективи використання вейвлет-перетворень при прогнозуванні навантаження на вебсервер; [103] – у статті автором запропоновано використання методу визначення найефективнішого типу вейвлету; [102] – у статті автор сформулював основи концептуальної моделі прогнозування навантаження на вебсервер; [117] – у статті автор дослідив особливості попередньої обробки часових рядів за допомогою дискретного вейвлет-перетворення; [70] – у статті автор

представив інтегральну модель прогнозування навантаження на вебсервер; [119] – у статті автор представив розроблений метод прогнозування навантаження на вебсервер; [118] – у тезі автор проаналізував рівень самоподібності вебтрафіку; [120] – у тезі автор презентував концепцію використання вейвлет-перетворень під час прогнозування навантаження на вебсервер; [71] – у тезі автор презентував особливості впровадження модульного підходу під час обробки великих масивів даних часових рядів; [72] – у тезі автор презентував архітектуру модульного підходу під час обробки великих масивів даних часових рядів; [121] – у тезі автор презентував гібридний підхід прогнозування часових рядів акцій з використанням вейвлет-перетворення та нейронних мереж; [69] – у тезі автор презентував обґрунтування використання дискретного вейвлет-перетворення при обробці часових рядів навантаженості вебсервера; [123] – у тезі автор презентував концепцію прогнозування навантаженості вебсерверу з використанням вейвлет-перетворень; [83] – у тезі автор презентував особливості застосування вейвлет-перетворень під час аналізу рівня навантаженості вебсервера; [121] – у тезі автор презентував гібридний підхід прогнозування часових рядів з використанням вейвлет-перетворення та нейронних мереж.

У одноосібних наукових працях: [113] – у статті автор представив концептуальну модель ефективного прогнозування навантаження на вебсервер, визначив модель параметрів вебтрафіку; [111] – у статті автор дослідив можливості застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер; [115] – у статті автор проаналізував особливості прогнозування рівня вебтрафіку; [114] – у тезі автор презентував особливості архітектури нейромережевої моделі під час обробки часових рядів навантаження на вебсервер; [116] – у тезі автор презентував прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень; [110] – у тезі автор презентував визначення пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень.

Апробація результатів дисертації. Основні результати дисертації доповідались та обговорювались на 13 науково-технічних конференціях, з них 9 – міжнародних: Науково-технічних семінарах кафедри системного програмування і

спеціалізованих комп’ютерних систем КПІ ім. Ігоря Сікорського, м. Київ (2018 – 2025 рр.); The Eighth World Congress “Aviation in the XXI-st Century” – «Safety in Aviation And Space Technologies». – Kyiv Aviation University 10.10 – 12.10.2018; VII Міжнародній науково-практичній конференції "Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах" 8 – 10 листопада 2018 року, м. Чернівці, Україна; Науково-практичній конференції «Сучасні інформаційні технології та кібербезпека» (СІТК – 2018), Інститут спеціального зв’язку та захисту інформації Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”, 15 – 16 листопада 2018 року; The Second International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2019) 26 – 27 January 2019, Kiev, Ukraine; ITSec: Безпека інформаційних технологій: IX міжнародна науково-технічна конференція, 22 – 27 березня 2019 р. – К.: НАУ, 2019; XVI Всеукраїнській науково-практичної конференції студентів, аспірантів та молодих вчених (м. Київ, 08 – 09 грудня 2020 р.); IX Міжнародній науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», 25 листопада 2022. – К: НУХТ, 2022; I Міжнародній науково-практичній конференції (Київ – Вроцлав, 26 травня 2022 р.);. XVI міжнародній науково-практичній конференції "Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2023)" 23-24 травня 2023 р., К.:, НАУ, 2023; II Міжнародній науково-практичній конференції “Кібербезпека державних інституцій та подолання кризових станів”, К.: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2023; XI Міжнародній науково-технічній Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», 27 листопада 2024. – К: НУХТ, 2024; XVII Науково-практичній конференції магістрантів та аспірантів "Прикладна математика та комп’ютинг" (ПМК-2024) (м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 20-22 листопада 2024 р.) тощо.

Публікації. Основні положення дисертаційної роботи опубліковані в 22

наукових працях, серед яких 8 статей у наукових фахових виданнях (в тому числі 1 стаття у закордонних наукових періодичних виданнях, що входять до наукометричної бази «Scopus», та 7 статей у наукових фахових виданнях України, які включені до наукометричних баз даних), 13 публікацій в збірниках тез доповідей науково-технічних конференцій, 1 свідоцтво про реєстрацію авторського права на комп'ютерну програму.

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, чотирьох розділів, висновків і додатків. Загальний обсяг роботи становить 277 сторінок друкованого тексту (серед яких основна частина – 222 сторінки), 48 рисунків, 25 таблиць, 8 додатків та списку використаної літератури з 126 найменувань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ЗАДАЧ ДОСЛІДЖЕННЯ

1.1. Науково-практична задача прогнозування навантаження на вебсервер

Одна з ключових характеристик розвитку сучасного суспільства – швидке зростання обсягу інформації, що обробляється та передається через локальні й глобальні комп'ютерні мережі та системи. Тому для прогнозування завантаженості вебсерверів необхідно розглянути низку проміжних завдань, зокрема: аналіз вимог до системи захисту інформації, вибір відповідних засобів безпеки, розробку комплексної системи захисту та практичне застосування механізмів безпеки. Це дозволяє оцінити стійкість парольного захисту, налаштувати ефективні параметри безпеки систем управління базами даних, оцінити рівень захищеності вебсерверів від атак, а також розпізнавати мережеві загрози та ефективно використовувати відповідні засоби безпеки [34].

Отже, важливою науково-практичною задачею є забезпечення ефективного рівня навантаженості вебсервера та прогнозування майбутнього вебтрафіку.

На сьогодні забезпечення стабільної роботи вебсерверів ґрунтується на комплексному поєднанні організаційних, інженерно-технічних та програмно-апаратних заходів, спрямованих на гарантування конфіденційності, цілісності, доступності, достовірності інформації та ефективного моніторингу [106]. Аналіз наявних досліджень [29, 40] свідчить, що організація роботи вебсерверів зводиться до пошуку ефективного балансу між необхідним рівнем захисту та доступними ресурсами. Потреби у захисті залежать від значущості та обсягів інформації, що обробляється, умов її зберігання, передачі та використання, тоді як ресурси визначаються їхньою максимальною доступністю або вимогами досягнення певного рівня безпеки.

Окремо постає питання моніторингу та прогнозування стану вебсерверів, що є основою для їхньої оптимізації [106]. Вдосконалення моделей прогнозування навантаження дозволяє точніше визначати поточний стан серверів, прогнозувати

їхню завантаженість та ухвалювати ефективні управлінські рішення. Це безпосередньо пов’язано з ключовими завданнями у сфері управління навантаженістю вебсерверів, такими як контроль параметрів безпеки [4], виявлення та протидія атакам чи спаму [7, 106], оптимізація захисних механізмів [15, 44], захист від вірусів [34].

Для вирішення цих питань дослідники пропонують розробку моделей моніторингу та оптимізації роботи вебсерверів [21], хоча наголошується на обмеженості традиційних алгоритмічних методів прийняття рішень. У зв’язку з цим перспективним підходом є застосування методів теорії вейвлет-перетворень, які продемонстрували ефективність у розв’язанні економічних, фінансових та технічних задач [45, 46]. Водночас у науковій літературі відсутня загальна методика розробки вейвлет-моделей для оптимізації роботи вебсерверів. Тому для визначення подальших напрямів досліджень було проведено аналіз можливостей застосування теорії вейвлет-перетворень у контексті розв’язання зазначених завдань.

Інтернет функціонує на основі багаторівневої архітектури «клієнт–сервер». Клієнтом у цій системі є робоча станція – персональний комп’ютер або віддалений термінал, через який користувач отримує доступ до інтернет-ресурсів.

Сервер – це програмний компонент обчислювальної системи, який виконує сервісні функції на запит клієнта, надаючи йому доступ до певних ресурсів. Під час взаємодії з клієнтом (або з декількома клієнтами одночасно) сервер розподіляє необхідні ресурси міжпроцесної взаємодії, такі як спільна пам’ять, пайпи чи сокети, і очікує запити на встановлення з'єднання. Залежно від типу ресурсу сервер може обслуговувати процеси як у межах однієї системи, так і на віддалених машинах через мережеві з'єднання. Деякі сервери залишаються в режимі очікування, якщо немає запитів, тоді як інші можуть виконувати фонові завдання, наприклад, збір інформації, для яких робота з клієнтами є вторинною. Також термін «сервер» нерідко застосовується до апаратного забезпечення, на якому працює відповідне програмне забезпечення.

Обмін даними між вузлами Інтернет здійснюється за допомогою спеціальних

протоколів, що регламентують спосіб і правила взаємодії. Основним стандартом передачі інформації є стек протоколів TCP/IP (Transport Control Protocol / Internet Protocol), який визначає, як різні комп'ютери з різними операційними системами можуть взаємодіяти між собою.

Однією з найпопулярніших складових Інтернету є веб (World Wide Web – всесвітня павутина). Її популярність пояснюється зручністю роботи з інформацією різних форматів – від простого тексту до мультимедійного контенту. Окремий інформаційний ресурс вебмережі, що має власну адресу, називається вебсторінкою або вебсайтом. Для передачі даних у вебсередовищі використовується HTTP (Hypertext Transfer Protocol) – протокол прикладного рівня, що забезпечує взаємодію між клієнтами та вебсерверами.

Вебсервер – це сервер, що обробляє запити користувачів (клієнтів) відповідно до протоколу HTTP, забезпечуючи актуалізацію, збереження інформації вебсторінок та взаємодію з іншими серверами [107]. Для доступу до певного вебресурсу (вебсайту) клієнтський додаток (браузер) встановлює TCP-з'єднання з відповідним вебсервером. Це з'єднання здійснюється через сервер провайдера та Інтернет.

Кожен вебресурс має унікальну URL-адресу, яка використовується для його ідентифікації та доступу. Браузер, встановивши з'єднання, надсилає вебсерверу запит на обробку відповідно до протоколу HTTP.

Як показано у [18, 37], режими роботи вебсерверів можуть бути як стаціонарними, так і нестаціонарними, характеризуючись багатоперіодичною зміною функціональних параметрів. Нестаціонарність проявляється у зміні моментів виникнення та зникнення періодичних складових.

Типи вебсерверів можна класифікувати за кількома основними критеріями – архітектурою, функціональністю, платформою, типом обслуговування та способом обробки запитів.

За архітектурою вебсервери поділяються на монолітні та модульні. Монолітні сервери (наприклад, старі версії Apache) мають цілісну структуру, де всі функції реалізовані в єдиному процесі. Натомість модульні архітектури (як Nginx або

LiteSpeed) забезпечують гнучкість, розподіл навантаження та легке розширення функціоналу за рахунок підключення незалежних модулів. Такий підхід є більш ефективним для високонавантажених систем і дозволяє оптимізувати використання ресурсів.

За функціональністю вебсервери можуть бути універсальними або спеціалізованими. Універсальні вебсервери (Apache, Nginx, IIS) підтримують широкий спектр протоколів і технологій – HTTP/HTTPS, CGI, PHP, FastCGI, WebSocket тощо. Спеціалізовані сервери, навпаки, оптимізовані для певних задач, наприклад, сервери застосунків (Tomcat, Node.js) або сервери статичного контенту (Caddy, Lighttpd).

За платформою вебсервери бувають кросплатформеними (Apache, Nginx, Lighttpd), які працюють на різних операційних системах (Windows, Linux, macOS), або орієнтованими на певне середовище – наприклад, IIS, що інтегрований у Windows Server, або OpenLiteSpeed, який найкраще оптимізований під Linux.

За типом обслуговування розрізняють відкриті (публічні) вебсервери, які надають доступ до ресурсів через Інтернет, і внутрішні (локальні), що використовуються в межах корпоративних мереж.

Нарешті, за способом обробки запитів сервери можна поділити на багатопроцесні, багатопотокові та подієво-орієнтовані. Багатопроцесна модель (Apache Prefork) створює окремий процес на кожен запит, що забезпечує ізоляцію, але споживає більше пам’яті. Багатопотокова (Apache Worker) дозволяє обслуговувати багато клієнтів у межах одного процесу, а подієво-орієнтована (Nginx, Node.js) працює на неблокуючій моделі вводу-виводу, що робить її найефективнішою при обробці великої кількості одночасних запитів.

У рамках дисертаційного дослідження було обрано вебсервер Apache HTTP Server, оскільки він є одним із найпоширеніших, відкритих та добре документованих рішень, що забезпечує широкий спектр можливостей для аналізу, моніторингу та моделювання навантаження. Apache підтримує модульну архітектуру, яка дозволяє гнучко керувати обробкою HTTP-запитів, розширювати функціональність за допомогою додаткових модулів (mod_status, mod_rewrite,

mod_proxy, mod_logio) та збирати детальну статистику вебтрафіку, необхідну для формування навчальної вибірки.

Крім того, вебсервер Apache є платформонезалежним і може бути розгорнутий як у локальних мережах, так і в хмарних середовищах, що дає змогу експериментально перевірити ефективність розроблених моделей прогнозування навантаження на різних інфраструктурних рівнях. Його архітектура також забезпечує сумісність із системами збору логів, інструментами аналітики та засобами машинного навчання, що робить Apache оптимальним вибором для дослідження динаміки вебнавантаження та реалізації інтегрованої моделі прогнозування.

За архітектурою Apache належить до модульних багатопотокових вебсерверів. Його архітектура побудована на основі гнучкої системи модулів (Multi-Processing Modules), які визначають спосіб обробки запитів: prefork – створює окремий процес для кожного з'єднання (підходить для сумісності зі старими бібліотеками без потокобезпеки); worker – використовує багатопотокову модель для підвищення ефективності; event – реалізує подієво-асинхронну архітектуру, оптимізовану для високих навантажень і великої кількості одночасних клієнтів.

За функціональністю Apache є універсальним вебсервером загального призначення (ВЗП), який може обслуговувати як статичний контент (HTML, CSS, зображення), так і динамічні вебдодатки через інтеграцію з мовами програмування (PHP, Python, Perl) або платформами (CGI, FastCGI, mod_wsgi). Завдяки модульності Apache може виконувати функції зворотного проксі, балансувальника навантаження, SSL/TLS-сервера та вебкешу.

За платформою Apache HTTP Server є кросплатформним – він підтримує операційні системи Linux, Windows, macOS, BSD, Solaris та інші UNIX-подібні системи. Це забезпечує його універсальність для розгортання в різних інфраструктурних середовищах – від локальних серверів до хмарних дата-центрів.

За типом обслуговування Apache може працювати як локальний сервер (для внутрішніх корпоративних систем), так і публічний вебсервер (для загальнодоступних вебресурсів). У режимі балансування навантаження він може

виступати фронтенд-сервером, що розподіляє запити між кількома бекенд-серверами. За способом обробки запитів Apache використовує гібридну модель обробки запитів – поєднання процесної, потокової та подієвої парадигм.

Це дозволяє адаптувати його роботу до характеру трафіку: у середовищах із невеликою кількістю клієнтів він може працювати у процесному режимі, тоді як у високонавантажених системах – у неблокуючому асинхронному режимі (event-driven). Будь-який сервер, і вебсервер в тому числі, складається з чотирьох ключових апаратних компонентів: центрального процесора, оперативної пам’яті, накопичувача (дискової підсистеми) та мережевого інтерфейсу. Для оцінки рівня навантаження сервера необхідно здійснювати збір і аналіз статистичних даних про кожну з цих складових.

Зокрема, навантаження на процесор не повинно перевищувати 10–20% у звичайному режимі. Виняток становлять спеціалізовані сервери, наприклад, для обробки мультимедійних файлів або зображень. Якщо завантаження перевищує 20%, доцільно перевірити ефективність роботи дискової та мережевої підсистем.

Інколи вебсервер може функціонувати повільніше, ніж очікується: затримки у завантаженні сторінок, тривале виконання скриптів тощо. У сфері мережевих показників сам по собі високий трафік не завжди є проблемою. Проте, коли обсяги наближаються до граничних значень пропускної здатності, виникає потреба у масштабуванні.

Згідно зі звітом Google Analytics (рис. 1.1) за 2024 рік, спостерігається закономірність у щоденному глобальному використанні Інтернету залежно від часу доби. Найбільша активність припадає на період між 9:00 ранку та полуднем, після чого спостерігається поступове зниження протягом дня. У проміжку з 19:00 до 22:00 інтенсивність використання дещо стабілізується, а потім знижується до мінімального рівня о 4:00 ранку [24].

Вебсервери, що забезпечують віддалений мережевий доступ до інформаційних ресурсів за протоколом HTTP, є ключовим елементом територіально розподілених комп'ютерних систем. Ступінь їхньої завантаженості безпосередньо впливає на ефективність роботи всієї інформаційної системи підприємства, тому до

їхньої надійності та безперебійної роботи висуваються високі вимоги. Незважаючи на наявність сучасних рішень у сфері управління навантаженням вебсерверів, часті перенавантаження вказують на наявність прогалин у прогнозуванні навантаження. Це підтверджують випадки відмов у роботі серверів великих соціальних мереж, IT-компаній та державних установ.

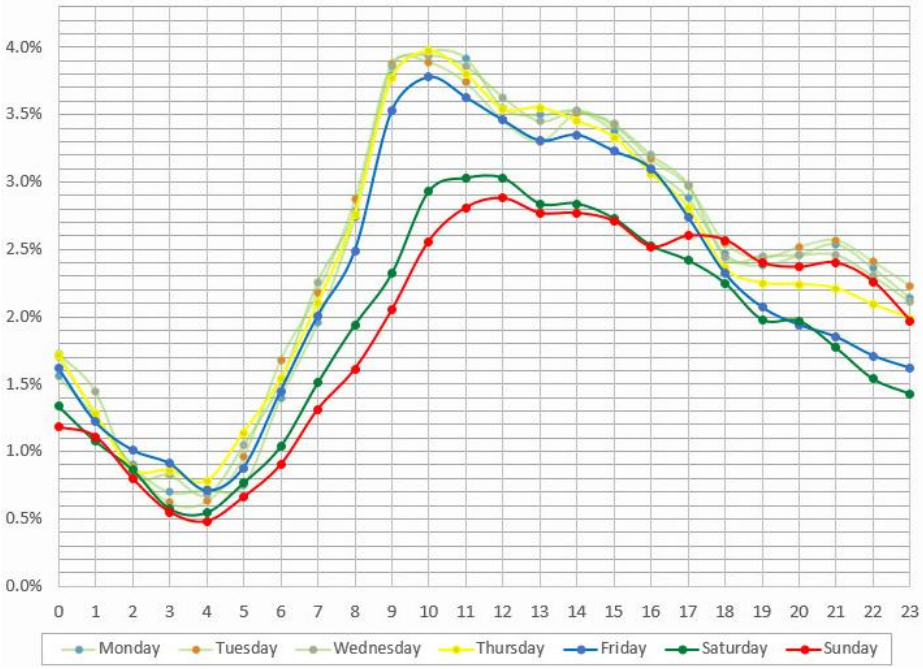

Рис. 1.1. Коливання глобального вебтрафіку за тиждень

Дослідження [34, 37] аналізують можливі атаки на вебсервери, зокрема атаки типу відмова в обслуговуванні (DoS/DDoS), спрямовані на порушення конфіденційності та цілісності інформації. Основний механізм DoS-атаки полягає в тому, що операційна система сервера має обмежену кількість відкритих з'єднань і можливостей обробки запитів, що залежить від обчислювальних ресурсів (швидкодії процесора, обсягу оперативної пам'яті, пропускної здатності мережі).

Основними типами атак на вебсервери є:

1. Передача некоректного запиту – якщо у вебсервері чи ОС є вразливості, можливе зависання системи. Проте сучасні версії ПЗ знижують ефективність таких атак [106].
2. Масова відправка запитів з однієї IP-адреси – ефективність атаки залежить

від рівня адміністрування серверу та продуктивності обладнання [34].

3. DDoS-атака – аналогічна другому типу, але здійснюється з великої кількості пристроїв (бот-мереж). Ефективність атаки зростає пропорційно числу атакуючих пристроїв.
4. Зловживання серверними додатками – атака використовує легітимні запити до вебсервера через скриптові технології (PHP, Java, ASP.NET тощо). Найбільш небезпечний тип атаки, оскільки викликає надмірне використання ресурсів сервера [17, 106].

Методи захисту від DoS-атак включають оновлення ПЗ, вдосконалення адміністрування, застосування потужного та резервного обладнання, інтегрованого з системами балансування навантаження [19, 42]. Однак захист від масованих атак потребує значних економічних витрат через надлишкові апаратні ресурси.

Водночас найбільш загрозливими є атаки, які використовують санкціоновані звернення до серверних додатків. Це підкреслює важливість розробки нових методів прогнозування навантаження вебсерверів. Зокрема, перспективним підходом є використання вейвлет-перетворень, що дозволяють передбачати пікові навантаження та забезпечувати відмовостійкість вебсерверів. Однак у доступних джерелах не міститься детального обґрунтування вибору типу вейвлет-моделей та розрахунку їхніх параметрів, що відкриває перспективи для подальших досліджень.

1.2. Основні положення в області теорії вейвлет-перетворень

Сам термін "вейвлет" означає математичну функцію, яка дозволяє аналізувати частотні компоненти сигналів на різних масштабах. У загальному випадку аналіз сигналів виконується у просторі вейвлет-коефіцієнтів, що враховує масштаб, час та рівень амплітуди (Scale-Time-Amplitude). На відміну від класичного перетворення Фур'є, вейвлет-спектрограми забезпечують точну часову локалізацію спектральних компонентів сигналу [16, 20].

У вейвлет-перетвореннях використовуються базисні функції з компактною підтримкою, які можуть мати різну природу – модульовані імпульсами синусоїди, функції зі стрибками рівня тощо. Такі функції добре підходять для аналізу сигналів

із локальними особливостями, зокрема з стрибками, розривами та різкими перепадами [10].

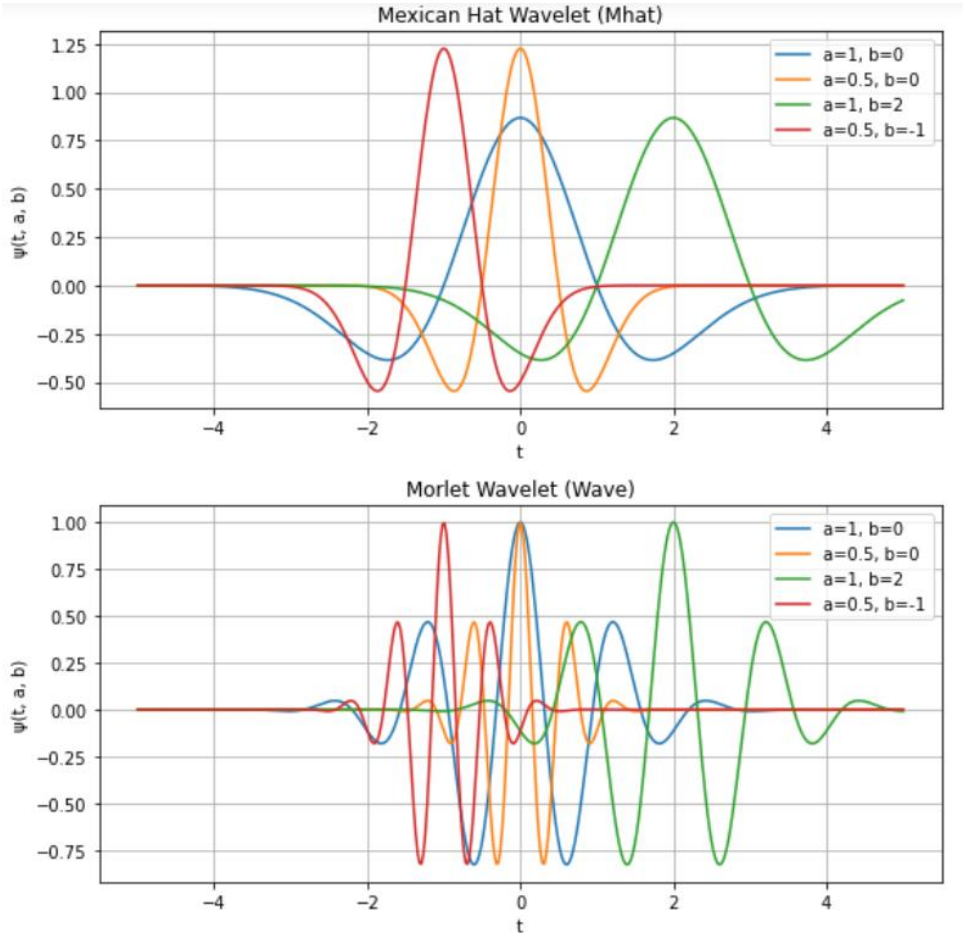

Рис. 1.2. Приклад представлення вейвлетів Mexican Hat (Mhat) і Morlet (Wave) з різними параметрами a (масштаб) та b (зсув) (авторська реалізація)

Для даних, змінних у часі потрібно забезпечити однозначність як процесу декомпозиції, так і відновлення сигналу. Досягти цього можливо лише при використанні ортогональних та біортогональних вейвлетів [22].

Вейвлет-перетворення є потужним математичним інструментом для аналізу сигналів, що дозволяє виявляти їх локальні особливості як у часовій, так і у частотній областях. Основою вейвлетного аналізу є вейвлет-функція, яка задовольняє певним умовам нормування та обмеженості (рис. 1.2).

Неперервне вейвлет-перетворення сигналу $f(t) \in L^2(\mathbb{R})$, яке застосовується для якісного аналізу в часово-частотній області, є аналогом перетворення Фур'є,

але замість гармонічного базису $e^{-j\omega t}$ застосовується вейвлетний базис $\psi\left(\frac{t-b}{a}\right)$.

Тут L^2 – це простір квадратноінтегровних функцій, також відомий як гільбертів простір. Формально він визначається як множина усіх функцій $f(t)$, для яких виконано умову [54]:

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty. \tag{1.1}$$

Цей простір є повним, тобто будь-яка збіжна послідовність функцій у цьому просторі збігається до функції, яка також належить L^2. Саме тому всі розклади, і вейвлет-перетворення зокрема, виконуються в рамках L^2, де забезпечується ортогональність та збереження енергії сигналу [25, 89].

У загальному випадку неперервне вейвлет-перетворення (Continuous Wavelet Transform, CWT) функції *f(t)* зі скінченною енергією у гільбертовому просторі $L^2(\mathbb{R})$ записується наступним чином [59]:

$$W_{f(a,b)} = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} f(t)\psi^*\left(\frac{t-b}{a}\right) dt, \tag{1.2}$$

де $f(t)$ – вхідний сигнал, $\psi(t)$ – материнський (базовий) вейвлет, a – параметр масштабу (обернено пропорційний частоті), b – параметр зсуву (відповідає часу), $\psi^*(t)$ – комплексно-спряжена функція від $\psi(t)$, множник $(\frac{1}{\sqrt{|a|}})$ забезпечує незалежність норми функцій від масштабування a.

Комплексно спряжений вейвлет визначається як [60]:

$$\psi^*(t) = \overline{\psi(t)}. \tag{1.3}$$

і використовується у випадку комплексних вейвлетів, таких як вейвлет Морле: $\psi(t) = e^{-t^2} e^{i\omega_0 t}$. У реальних вейвлетах (наприклад, Хаара, Добеші, Коіфмана) $\psi^*(t) \equiv \psi(t)$, оскільки вони є дійсними функціями.

На відміну від Фур'є-спектра, який залежить лише від частоти, вейвлетний масштабно-часовий спектр $W(a,b)$ є функцією двох параметрів. Оскільки параметри a та b можуть набувати довільних значень у своїх визначених областях, вейвлет-перетворення забезпечує гнучкий аналіз локальних особливостей сигналу.

Для точного кількісного аналізу (розкладу сигналів із можливістю

подальшого відновлення) як вейвлетний базис можна використовувати будь-які локалізовані функції $\psi(t)$, якщо існує відповідна функція вейвлет-двійник (вейвлет-реконструктор) $\psi^{\#}(t)$, що утворює разом із основним вейвлетом (вейвлетом-аналізатором) $\psi(t)$ парний базис простору $L^2(\mathbb{R})$. Тоді будь-яку функцію у цьому просторі можна подати у вигляді ряду [39]:

$$f(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} W(a,b)\psi^{\#}{}_{(a,b)}(t)dadb, \tag{1.4}$$

де коефіцієнти $W(a,b)$ визначаються проекцією сигналу на вейвлетний базис:

$$W(a,b) = \langle f(t), \psi_{(a,b)}(t)\rangle = \int f(t)\psi_{(a,b)}(t)dt. \tag{1.5}$$

Функція вейвлета-двійника $\psi^{\#}(t)$ використовується у біортогональних вейвлетних системах для забезпечення точного відновлення сигналу після його розкладу у вейвлетні коефіцієнти. Якщо вейвлетна система ортогональна, то вейвлет-двійник збігається з основним

$$\psi^{\#}(t) \equiv \psi(t) \tag{1.6}$$

і базис також буде ортогональним. Проте, навіть якщо вейвлет не є ортогональним, але має функцію-двійника $\psi^{\#}(t)$, можна сформувати біортогональні сімейства $\{\psi_{(m,k)}(t)\}, \{\psi^{\#}{}_{(z,p)}(t)\}$, які задовольняють умові біортогональності [80]:

$$\langle \psi_{(m,k)}(t), \psi^{\#}{}_{(z,p)}(t)\rangle = \delta_{mz}\delta_{kp}, m,k,z,p \in I. \tag{1.7}$$

Символи Кронекера δ_{mz}, δ_{kp} визначають ортогональність функцій у базисі:

$$\delta_{ij} = \begin{cases} 1, i = j, \\ 0, i \neq j. \end{cases} \tag{1.8}$$

Це дозволяє виконувати розкладання сигналів у вейвлетні ряди із можливістю деконволюції сигналу та його точного відновлення за допомогою зворотного перетворення [52, 53, 77, 78, 109].

Дискретне вейвлет-перетворення (DWT) реалізується шляхом дискретизації параметрів масштабування та зсуву, зазвичай у вигляді:

$$a = 2^j, b = k \cdot 2j, \tag{1.9}$$

де j,k – цілі числа, що визначають рівень масштабування та часовий зсув відповідно.

Тоді дискретне вейвлет-перетворення можна записати як:

$$W(j,k) = \sum_{-\infty}^{\infty} f(n)\,\psi_{j,k}(n), \tag{1.10}$$

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t - k2^j}{2^j}\right) = 2^{-j/2}\psi\left(2^{-j}t - k\right), \tag{1.11}$$

Така дискретизація дозволяє ефективно обчислювати коефіцієнти і застосовувати їх у цифровій обробці сигналів.

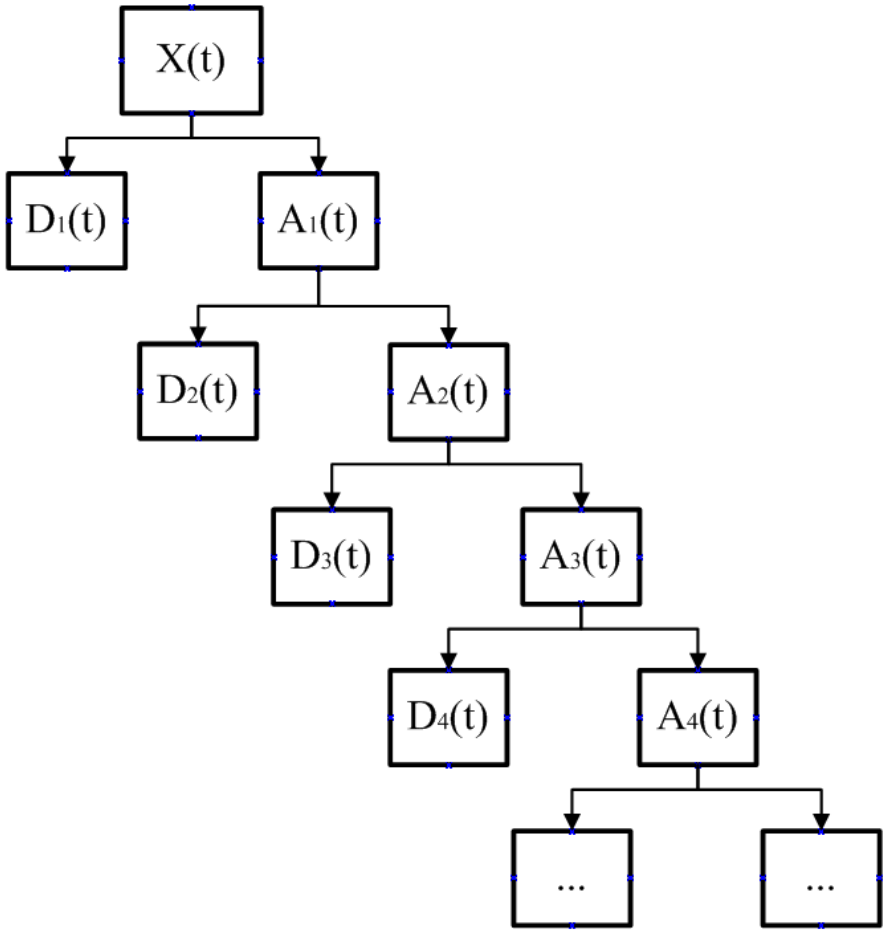

Рис. 1.3. Деревоподібна структура дискретного вейвлет-перетворення

DWT представляється через набір коефіцієнтів апроксимації $c_{j,k}$ та коефіцієнтів деталей $d_{j,k}$ (рис. 1.3):

$$c_{j,k} = \int_{-\infty}^{\infty} f(t)\phi_{j,k}(t)dt, \tag{1.12}$$

$$d_{j,k} = \int_{-\infty}^{\infty} f(t)\psi_{j,k}(t)dt, \tag{1.13}$$

де $\phi_{j,k}(t)$ – масштабуюча функція, яка визначає основний тренд сигналу, а $\psi_{j,k}(t)$ – вейвлет-функція, що відображає локальні деталі.

Вейвлет-функції $\psi(t)$повинні задовольняти наступним умовам:

- нормування за енергією забезпечує сталу енергетичну характеристику вейвлету:

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt = 1; \tag{1.14}$$

- умова нульового середнього значення (осциляторність) гарантує, що вейвлети вловлюють лише змінні компоненти сигналу та нечутливі до постійних складових:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0; \tag{1.15}$$

- локалізація у часі та просторі за допомогою формули визначення материнського вейвлету, що дозволяє аналізувати сигнали з хорошою часово-частотною роздільною здатністю.

Відновлення сигналу з його неперервного вейвлет-перетворення здійснюється за допомогою рівняння:

$$f(t) = \frac{1}{\mathrm{C}_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_{f(a,b)} \psi_{(a,b)}(t) \frac{dadb}{a^2}, \tag{1.16}$$

де C_ψ – енергетична константа вейвлету, яка залежить від вибору вейвлет-функції і визначається як:

$$\mathrm{C}_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega. \tag{1.17}$$

Відповідно, $\Psi(\omega)$ – це спектральне представлення (Фур'є-образ) вейвлет-функції $\psi(t)$. Іншими словами, $\Psi(\omega)$ – це перетворення Фур'є для вейвлету $\psi(t)$:

$$\Psi(\omega) = \int_{-\infty}^{\infty} \psi(t) e^{-j\omega t} dt. \tag{1.18}$$

У дискретному випадку відновлення сигналу, яке за своєю суттю є відновленням функції набору вейвлет-коефіцієнтів, здійснюється за допомогою

суми відповідних вейвлет-коефіцієнтів:

$$f(n) = \sum_j \sum_k W(j,k)\psi_{j,k}(n), \tag{1.19}$$

де сума ведеться за всіма рівнями масштабування *j* і зміщеннями *k*.

Вейвлет-перетворення є потужним інструментом аналізу сигналів, що поєднує часову та частотну локалізацію. Вони забезпечують ефективне представлення та обробку складних сигналів, що робить їх незамінними в багатьох наукових і технічних сферах, зокрема в задачі аналізу часових рядів при прогнозуванні та виявленні можливих аномалій поведінки [97].

Вейвлет-аналіз є потужним інструментом для обробки нестаціонарних сигналів, що дозволяє розкласти сигнал на компоненти з різними частотами і локалізувати їх у часі. Це відрізняє вейвлет-аналіз від традиційних методів, таких як перетворення Фур'є, які дають інформацію лише про частоти, але не про те, коли ці частоти з'являються у часі [95].

Вейвлет-аналіз використовується для розбиття часового ряду на компоненти різної частоти, що дозволяє виділяти особливості, які традиційні методи (наприклад, Фур'є-перетворення) можуть не виявити. Це є корисним для прогнозування нерегулярного та нестабільного навантаження на вебсерверах, оскільки можна врахувати коротко-, середньо- та довгострокові коливання. Проте водночас вейвлет-аналіз може бути складним для налаштування, оскільки потребує вибору відповідного вейвлету та параметрів. А його чутливість до шуму може бути як перевагою, так і недоліком залежно від якості даних та їх зашумлення. Крім того, вейвлет-перетворення потребує значної великої обчислювальної потужності, що не завжди є прийнятним для моделювання системи реального часу [99].

Альтернативним способом для моделювання систем, стан яких змінюється з часом залежно від деякої випадковості є використання марківських процесів. Це підходить для прогнозування стану серверів, оскільки можна оцінювати ймовірність переходу з одного стану (наприклад, високого навантаження) в інший. Однак марківські моделі часто передбачають обмеження на кількість станів і їхній фіксований характер, що може не враховувати всіх факторів впливу на сервер. Такі

моделі можуть не враховувати залежності довгого періоду, які є важливими при прогнозуванні серверного навантаження.

З іншого боку методи динамічного програмування є ефективними для вирішення задач, де важливо враховувати довгострокові наслідки кожного окремого рішення та складні взаємозалежності між станами системи та прийнятими діями. Методи динамічного програмування використовуються для оптимізації багатокрокових процесів, що змінюються з часом, як-от керування ресурсами вебсервера. Це дозволяє адаптивно змінювати розподіл ресурсів залежно від поточного та прогнозованого навантаження. Водночас методи динамічного програмування можуть бути дуже обчислювально інтенсивними. Вони вимагають точного знання ймовірностей та переходів, що може бути складно досягти у реальних умовах, а їх ефективність залежить від точності прогнозів, на основі яких будуються оптимізаційні стратегії.

Таким чином кожен із розглянутих методів має свої переваги та недоліки. Зокрема, вейвлет-аналіз дозволяє краще обробляти часи коливань та шум у даних, марківські процеси забезпечують врахування ймовірностей переходів між станами, а динамічне програмування дає можливість адаптивно налаштовувати стратегії керування навантаженням.

Відомі промислові платформи, що використовуються для прогнозування навантаження на вебсервери та оцінки рівня вебтрафіку, зазвичай інтегрують методи аналізу даних, статистичні моделі, нейронні мережі та алгоритми машинного навчання для аналізу трафіку та передбачення можливих піків навантаження. Найбільш популярними з них є Google Analytics, New Relic, AWS CloudWatch тощо. У порівняльній таблиці 1.1 можна побачити їх специфікацію та обмеження з ключовими характеристиками. З аналізу у таблиці 1.1 можна зробити висновки, що для забезпечення ефективного моніторингу та аналізу в реальному часі найкраще підходять промислові рішення на базі Apache Kafka з інтеграцією машинного навчання (ML), а також інструменти New Relic та Dynatrace. Для глибокого аналізу та прогнозування на довготривалий період ефективно

використовувати платформу TensorFlow Extended (TFX), що забезпечує високоякісну обробку великих обсягів даних та автоматизацію робочих процесів.

Таблиця 1.1

Порівняльна таблиця промислових платформ для прогнозування навантаження на вебсервери з ключовими характеристиками

Платформа	Математичний апарат	Тривалість прогнозу	Оперативність прогнозу	Ресурсоємність	Обсяг статистики
Google Analytics	Статистичний аналіз, регресія	Хвилини – години	Висока	Середня	Великий обсяг історичних даних
New Relic	Евристичні методи, статистичний аналіз	Секунди – хвилини	Висока	Висока	Постійний потік даних
AWS CloudWatch	Статистика, ML (AWS AI/ML інтеграція)	Хвилини – години	Висока	Висока	Від малого до великого
Dynatrace	AI/ML, кореляційний аналіз	Секунди – години	Дуже висока (автоматичний аналіз)	Висока	Великий обсяг потокових даних
Apache Kafka + ML	Машинне навчання, потокова аналітика	Мілісекунди – хвилини	Дуже висока (реальний час)	Висока	Великий обсяг потокових даних
Elasticsearch + Kibana	Пошукова аналітика, кореляційний аналіз	Секунди – години	Висока	Середня	Від середнього до великого
Grafana + Prometheus	Часові ряди, статистика	Секунди – години	Висока	Середня	Від малого до середнього
TensorFlow Extended (TFX) + Time-Series Analysis	Глибоке навчання, рекурентні нейронні мережі	Хвилини – дні	Середня	Висока	Великий набір історичних даних

Для гнучкого моніторингу та візуалізації даних широкого використання набули комбінації Grafana з Prometheus, а також Elasticsearch з Kibana, які дозволяють здійснювати детальний аналіз і візуалізувати дані в реальному часі.

Ці системи та інструменти мають різні рівні складності та можливості для прогнозування навантаження на вебсервери. Промислові платформи для прогнозування навантаження на вебсервери та оцінки рівня вебтрафіку надають потужні інструменти для моніторингу, аналізу та прогнозування. Однак їх використання може бути обмеженим у контексті специфічних задач, що виникають у наукових дослідженнях або нестандартних випадках, зокрема, у дослідженні інноваційних підходів до адаптивного прогнозування навантаження чи оптимізації взаємодії вебсерверів.

Першою причиною для пошуку власного наукового рішення є те, що комерційні платформи зазвичай оптимізовані для широкого спектру стандартних задач і мають обмеження в гнучкості налаштувань або алгоритмів, що можуть бути необхідними для розв'язання конкретних наукових проблем. Наприклад, вони часто працюють за універсальними моделями, які не враховують специфічні особливості або нові підходи в прогнозуванні навантаження, що можуть бути виведені з новітніх наукових досліджень у галузі штучного інтелекту або адаптивних систем.

Іншою важливою причиною є відсутність можливості інтеграції з специфічними новими методами та алгоритмами, які можуть бути предметом дослідження. Відомі промислові платформи часто обмежуються встановленими методами прогнозування, що, хоча й ефективні, можуть не враховувати інноваційні підходи або специфічні умови роботи вебсервера, які є предметом наукових досліджень.

1.3. Аналіз підходів до вибору базисних вивлетів для задачі прогнозування навантаження на вебсервер

Прогнозування навантаження на вебсервер є складним завданням, оскільки вебтрафік може містити як короткочасні пікові навантаження, так і довготривалі

тренди. Вейвлет-перетворення дозволяє аналізувати такі динамічні зміни, оскільки добре локалізує сигнали як у часі, так і у частотній області [122].

Існує багато різних типів вейвлетів, які використовуються залежно від завдання. Найпопулярніші з них представлені у таблиці 1.2. Кожен вейвлет має свої унікальні характеристики в часовому просторі. Використання різних вейвлетів дозволяє виявляти специфічні властивості аналізованих даних.

Таблиця 1.2

Основні материнські вейвлети

Вейвлет	Аналітичний запис $\psi(t)$	Особливості
Дійсні неперервні базиси		
Mexican Hat (Ricker wavelet, мексиканський капелюх)	$\psi(t) = (1 - t^2)e^{-t^2/2}$	Друга похідна гаусіана, виявляє локальні піки та тренди, підходить для прогнозування стрибків навантаження
Гаусовий вейвлет n-го порядку	$\psi_n(t) = (-1)^n \frac{\partial^n}{\partial t^2}(e^{-t^2/2})$	Висока локалізація в часі та частоті, підходить для аналізу сигналів із плавними змінами. Аналіз геофізичних даних, біомедичних сигналів.
LP-вейвлет (Littlewood&Paley)	$\psi(t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t}$	Застосовується у методах стиснення та фільтрації сигналів, де важливо точно виділити окремі частотні компоненти, а також у реконструкції сигналів після видалення шумів.
Дійсні дискретні базиси		
Вейвлет Хаара (Haar)	$\psi(t) = \begin{cases} 1, & 0 \le t < 0.5 \\ -1, & 0.5 \le t < 1 \\ 0, & \text{інакше} \end{cases}$	Найпростіший, швидкий у розрахунках, підходить для різких змін
French Hat (Французький капелюх)	$\psi(t) = G_{\sigma_1}(t) - G_{\sigma_2}(t)$ $G_\sigma(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{t^2}{2\sigma^2}}$	Використовується для виявлення особливостей сигналу, таких як зміни амплітуди. Має хорошу часову локалізацію.

Вейвлет	Аналітичний запис $\psi(t)$	Особливості
Вейвлети Добеші (Daubechies, db2,db4,db8 тощо)	$\psi(t) = \sqrt{2}\sum_k g_k \phi(2t-k)$ $\phi(t) = \sqrt{2}\sum_k h_k \psi(2t-k)$	Згладжують тренди, підходять для багаторівневої декомпозиції, Чим вищий порядок N, тим гладша функція вейвлета і краща частотна локалізація, але зростає обчислювальна складність.
Симлети (Symlets)	$\psi(t) = \sum_k g_k \phi(2t-k)$	Фільтри симлетів мають більшу симетричність у порівнянні з фільтрами Добеші, що є важливим для аналізу і відновлення зображень та сигналів, де важлива висока точність при збереженні структури
Коіфлети (Coiflets)	$\psi(t) = \sum_k g_k \phi(2t-k)$	Фільтри коїфлетів є майже симетричними, мають більше нулів у похідних функцій порівняно з вейвлетами Добеші що дозволяє забезпечити більшу гладкість
Комплексні базиси		
Вейвлет Морле (Morlet)	$\psi(t) = \frac{1}{\sqrt{\sigma}} e^{-t^2/(2\sigma^2)} e^{i\omega_0 t}$	Підходить для аналізу періодичних коливань у навантаженні. Використовується для вивчення сигналів, що мають зміни як в часі, так і в частоті. Змінна σ контролює часову локалізацію, а ω_0 – частотну локалізацію.

Дійсні базиси часто формуються на основі похідних від функції Гауса $g_0(t) = \mathrm{e}^{-t^2/2}$, оскільки вона забезпечує ефективну локалізацію як у часовій, так і в частотній областях. Вищі похідні цієї функції мають більше нульових моментів, що

дозволяє досліджувати особливості даних вищих порядків (рис. 1.4).

У комплексному часовому просторі часто застосовується локалізований вейвлет Морле.

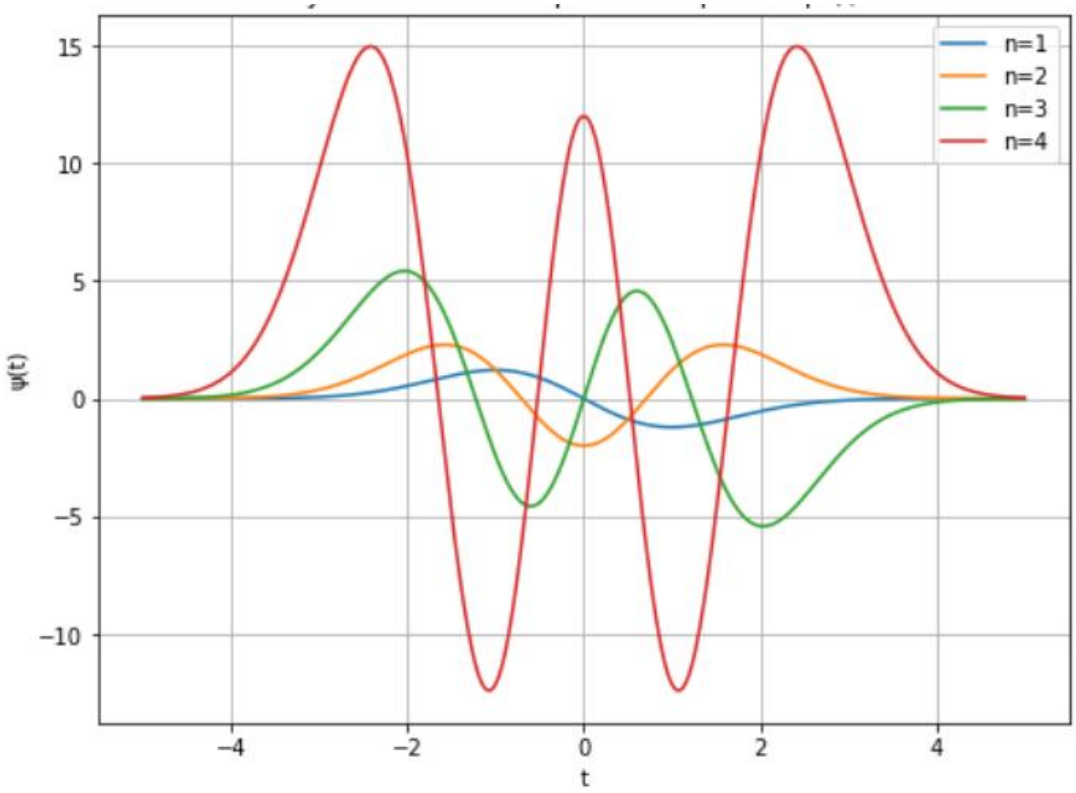

Рис. 1.4. Гаусові вейвлети перших чотирьох порядків.

При прогнозуванні навантаження на вебсервер на основі вейвлет-перетворень важливим є правильний вибір базисних вейвлетів, оскільки від цього залежить якість декомпозиції сигналу та точність прогнозу [23, 123].

При виборі вейвлетної бази для прогнозування навантаження слід враховувати [48, 50]:

- *локалізацію в часі та частоті* – важливо враховувати короткочасні стрибки в запитах. Вейвлет повинен добре відображати особливості часових рядів, зокрема стрибки, тренди та періодичні складові. Більш згладжені вейвлети (наприклад, вейвлети Добеші) краще підходять для аналізу плавних трендових змін, а більш компактні (наприклад, Хаара) – для виявлення різких змін;

- *стійкість до шуму* – реальні дані можуть містити аномалії, які не повинні впливати на прогноз. Якщо часовий ряд містить шум, вибирають вейвлети з хорошими фільтраційними властивостями, наприклад, коіфлети або симлети. Для роботи з шумними сигналами часто застосовують методи вейвлетного порогового відсічення (Wavelet Thresholding);
- *збереження основних компонентів сигналу* – необхідно зберігати як довготривалі тренди, так і короткі імпульсні зміни;
- *обчислювальну ефективність* – важливо вибрати вейвлети, які швидко обчислюються. Ортогональні вейвлети (Добеші, коіфлети, симлети) забезпечують компактне представлення сигналу, що корисно для стискання даних та виявлення трендів. Біортогональні вейвлети (Biorthogonal) забезпечують хорошу реконструкцію сигналу та використовуються в задачах відновлення даних.

Популярними вейвлетами [47] для задачі прогнозування часових рядів, зокрема і для прогнозування навантаження на вебсервер є наступні (табл. 1.2):

- Хаара (Haar, D2) – найпростіший вейвлет, добре підходить для аналізу різких змін;
- вейвлети Добеші (Daubechies, DbN) – одне із найпоширеніших сімейств, забезпечує баланс між точністю та обчислювальною ефективністю;
- симлети (Symlets, SymN) – модифікація сімейства Daubechies із покращеною симетрією, що важливо для довгострокового прогнозування;
- коіфлети (Coiflets, CoifN) – мають хороші частотні характеристики і часто використовуються у фінансових та економічних часових рядах;
- біортогональні (Biorthogonal, BiorN.M) – застосовуються у випадках, коли потрібна висока точність відновлення сигналу.

У якості стандартних методів вибору ефективного вейвлета зазвичай використовується наступні підходи [49]:

- критерій мінімізації середньоквадратичної похибки (MSE), коли обирається вейвлет, який забезпечує мінімальну похибку відновлення сигналу;

- крос-валідація для різних вейвлетів, які тестуються на навчальних даних, і обирається найкращий на основі точності прогнозу;
- кореляційний аналіз – аналізується кореляція вейвлетних коефіцієнтів з початковим часовим рядом для вибору найбільш інформативного вейвлета.

Таким чином, вибір базисного вейвлета залежить від характеру даних, цілей прогнозування та вимог до точності відновлення часових рядів.

Для задач прогнозування часових рядів зазвичай використовують дискретне вейвлет-перетворення (DWT), а не неперервне (CWT), з кількох причин [33, 85]:

- ефективне представлення даних, оскільки DWT дозволяє розкласти сигнал на апроксимації та деталі на різних масштабах, що зручно для аналізу трендів і короткострокових коливань, на відміну від CWT, яке обчислює коефіцієнти для всіх можливих значень масштабу та зміщення, що призводить до надлишкової інформації та збільшує обчислювальні витрати;
- менша обчислювальна складність, оскільки DWT використовує фільтрувальні банки і малий набір масштабів (зазвичай кратних 2), що значно прискорює розрахунки, на противагу CWT, яке вимагає інтегрування по всіх масштабах і зсувам, що робить його менш ефективним у реальних застосуваннях;
- компресія даних та усунення шуму, оскільки DWT дозволяє виділяти головні складові сигналу та відсікати шум без надмірності CWT, що важливо для якісного прогнозування.

1.4. Аналіз засобів прогнозування навантаження на вебсервер

Зростання популярності інтернет-ресурсів та збільшення кількості користувачів вимагають від вебсерверів високого рівня продуктивності та стабільності. Одним із головних завдань адміністраторів є прогнозування навантаження на сервер, що дозволяє запобігти його перевантаженню, збоям і забезпечити безперебійну роботу. Традиційні підходи до моніторингу та аналізу навантаження часто виявляються недостатньо ефективними для обробки раптових

піків активності, що може спричинити серйозні проблеми у функціонуванні вебсервісів. У сучасних умовах швидкого розвитку інформаційних технологій зростає потреба у використанні більш точних і гнучких методів прогнозування, здатних ефективно працювати з нестаціонарними даними.

Прогнозування навантаження на вебсервери є критично важливим завданням у сфері сучасних інформаційних систем. Вебсервери забезпечують доступ до цифрового контента та онлайн-сервісів, і їх стабільне функціонування напряму впливає на якість обслуговування користувачів і безперервність бізнес-процесів. Нестабільні навантаження можуть спричинити втрату доступу до сервісів, зниження швидкодії та погіршення користувацького досвіду.

Протягом останніх років було проведено багато досліджень у сфері прогнозування навантаження на вебсервери. Традиційні підходи [75], такі як лінійна регресія, ARIMA (AutoRegressive Integrated Moving Average) та інші методи аналізу часових рядів, мають свої обмеження у випадках значної динамічності та непередбачуваних змін навантаження. Лінійні моделі погано справляються з прогнозуванням складних, нелінійних процесів, що є типовими для вебсерверів. Хоча ARIMA-моделі демонструють певну гнучкість, їх точність знижується при наявності різких змін трендів та сезонних коливань. Крім того, автокореляційний аналіз [81], що застосовується для виявлення повторюваних шаблонів у навантаженні, має обмежену ефективність для нестаціонарних процесів, оскільки не забезпечує достатньої часової роздільної здатності.

Швидке перетворення Фур'є (FFT) [51] є класичним методом аналізу частотного спектру стаціонарних сигналів. Однак його головний недолік полягає у відсутності часової локалізації, що ускладнює аналіз динамічних процесів. Частково цю проблему вирішує короткочасне перетворення Фур'є (STFT), яке розбиває сигнал на короткі сегменти та застосовує FFT до кожного з них, забезпечуючи одночасний аналіз у часовій і частотній областях. Проте ключовим обмеженням STFT є фіксований розмір вікна: вузькі вікна забезпечують високу часову роздільну здатність, але низьку частотну, тоді як широкі вікна, навпаки, покращують частотну роздільність за рахунок зниження часової.

Альтернативним підходом до моделювання систем із динамічними змінами є використання марківських процесів [53]. Вони дозволяють оцінювати ймовірність переходу між станами, наприклад, від низького до високого навантаження на сервер. Це робить їх придатними для прогнозування поведінки вебсерверів. Водночас марківські моделі мають свої обмеження: вони передбачають дискретну кількість станів із фіксованими характеристиками, що не завжди відображає всю різноманітність впливових факторів. Крім того, вони погано враховують довгострокові залежності, які є важливими при прогнозуванні серверного навантаження.

Натомість дослідження [85, 90] підтверджують ефективність вейвлет-аналізу для аналізу часових рядів шляхом розкладу сигналу на різні частотні компоненти. Це дозволяє виявляти локальні особливості динаміки навантаження, що є особливо корисним для прогнозування нерегулярних та нестабільних змін на вебсерверах. Вейвлет-аналіз забезпечує врахування як короткострокових, так і довгострокових коливань, що дає змогу будувати більш точні прогностичні моделі. Однак одним із його недоліків є складність налаштування, зокрема вибору ефективної вейвлет-функції для конкретного завдання [111, 115]. Крім того, цей метод потребує значних обчислювальних ресурсів, що може бути обмеженням у реальних системах. Його чутливість до шуму також може як покращувати, так і погіршувати точність прогнозів, залежно від рівня зашумленості вихідних даних.

Праці [14, 43] демонструють широке застосування вейвлет-аналізу в прогнозуванні часових рядів у різних сферах, підкреслюючи його переваги над традиційними методами. Зокрема, його здатність до мультишкального аналізу робить його ефективним інструментом для обробки нестаціонарних даних і складних динамічних процесів, що актуально для задач прогнозування навантаження на вебсервери.

Нейронні мережі широко застосовуються [66] для прогнозування часових рядів завдяки їхній здатності навчатися на великих обсягах даних та виявляти складні нелінійні залежності. У сфері прогнозування навантаження на вебсервери вони можуть ефективно моделювати взаємозв’язки між вхідними параметрами,

дозволяючи передбачати динаміку змін із високою точністю.

Процес прогнозування часових рядів часто реалізується за допомогою нейронних мереж, зокрема рекурентних моделей RNN, таких як мережі з довгою короткочасною пам’яттю LSTM. Завдяки своїй здатності зберігати довготривалі залежності у даних, LSTM стали ключовим інструментом для аналізу часових рядів. Проте їхня ефективність може бути обмежена недостатньою здатністю до виявлення локальних патернів у даних [65].

Гібридні підходи, які поєднують згорткові нейронні мережі CNN із LSTM, забезпечують покращену точність прогнозування. CNN ефективно виділяють локальні особливості часових рядів, тоді як LSTM аналізують довготривалі залежності. Дослідження [40, 55, 67] підтверджують ефективність такого підходу, демонструючи підвищення точності прогнозування у порівнянні з традиційними методами.

Однак використання нейронних мереж пов’язане з певними викликами. Вони потребують налаштування великої кількості гіперпараметрів, що може призвести до проблеми перенавчання. Крім того, для ефективного навчання потрібен значний обсяг даних, а чутливість до шуму може впливати на якість прогнозів.

Прогнозування часових рядів є складною задачею через їхню нелінійність, стохастичність і складні часові залежності [74]. Одним із перспективних підходів є поєднання вейвлет-перетворення з нейронними мережами, що належить до гібридних методів аналізу. Використання вейвлет-перетворення дозволяє ефективно обробляти часові ряди, розкладаючи їх на різні масштабні компоненти, що покращує якість вхідних даних для нейромережевих моделей. Така комбінація сприяє підвищенню точності прогнозування завдяки адаптивному аналізу та потужним обчислювальним можливостям нейронних мереж.

Отже, для підвищення точності прогнозування навантаження на вебсервери необхідно застосовувати методи, здатні адаптуватися до змінних умов і враховувати складну динаміку навантаження, такі як нейронні мережі, вейвлет-аналіз та гібридні підходи.

Так, у праці [30] запропоновано метод прогнозування використання ресурсів

вебсервера на основі екстраполяції, проте зазначено, що він не враховує пікові навантаження, що знижує точність прогнозів. У роботі [31] розглянуто питання оцінки потреб у ресурсах для веборієнтованих систем при заданому навантаженні з обмеженням часу відповіді та використанням процесора не більше ніж на 90%, водночас автор наголошує на необхідності подальших досліджень для визначення граничного навантаження.

У дослідженні [32] представлено методику розрахунку середнього навантаження за умови пуасонівського потоку запитів, при цьому вказано, що збільшення середнього навантаження може спричинити критичне зростання часу очікування. Автор роботи [100] акцентує увагу на необхідності прогнозування поведінки інформаційної системи залежно від режимів навантаження та підкреслює важливість створення ефективного модуля для прогнозу.

Дослідження [101] застосовує методи теорії часових рядів для прогнозування навантаження, зокрема враховується самоподібність вхідного потоку запитів. У праці [108] проведено аналіз вебтрафіку та побудовано модель клієнт-серверної взаємодії, яка виявляє самоподібність реального трафіку.

У роботах [57, 115] запропоновано використання узагальнено-регресійної нейронної мережі для прогнозування короткострокового навантаження.

Публікації [13, 33, 96] демонструють, що одним із важливих напрямів підвищення ефективності вебсервера є вдосконалення механізмів виявлення кібератак шляхом аналізу шаблонів нормальної поведінки параметрів захисту, які враховують складний багатоперіодичний характер динаміки цих параметрів. На основі аналізу літератури зроблено висновок про доцільність удосконалення шаблонів нормальної поведінки вебсерверів шляхом впровадження сучасних методів частотно-часового аналізу сигналів, зокрема методів вейвлет-перетворення [96, 105, 119, 124].

Оцінюючи навантаження на вебсервер КМЗП, були розглянуті дослідження, присвячені прогнозуванню цього процесу [125]. Аналіз проведено з метою визначення підходів до створення ефективних методів прогнозування навантаження.

У роботі [106] представлено технологію моделювання завантаження каналу зв'язку для вебсервера КМЗП. Визначено основні чинники, що впливають на пропускну здатність каналу, зокрема: обсяг передаванх даних, використання мультимедійних компонентів, вимоги до затримки під час передавання даних, кількість активних користувачів, прийнятний час очікування, особливості побудови навчальних курсів, завантаження каналу іншими програмами та наявність проксі-сервера. Зазначено, що максимальний термін отримання результатів запиту для клієнта не повинен перевищувати 5 секунд. Для прогнозування навантаження застосовано пуасонівський потік запитів. Запропонована модель дозволяє оцінити середній час завантаження вебсторінки та рівень завантаження каналу зв'язку, ґрунтуючись на припущенні, що користувач перебуває на сторінці від 40 до 90 секунд. Дослідження демонстру періодичний характер навантаження вебсервера.

Запропонована формула для визначення ефективної продуктивності вебсерверів на основі критерію мінімізації часу обробки запитів на вебсторінці виглядає наступним чином (106):

$$\min_{\tau_i,\beta_i} S(\tau_1,\dots,\tau_N,\beta_1,\dots,\beta_N) = \min_{\tau_i,\beta_i} \sum_{i=1}^{N} \left(\frac{{\tau_i}^3 {\beta_i}^2 \Lambda^2}{2(1-\tau_i\beta_i\Lambda)} + \frac{\tau_i(1-\tau_i\beta_i\Lambda)}{\beta_i} \right), \quad (1.20)$$

$$\Lambda = \sum_{i=1}^{N} \frac{\lambda_i}{\tau_i}, \sum_{i=1}^{N} \tau_i = 1, \quad (1.21)$$

де $\tau_i \geq 0$ – штраф за очікування запиту в черзі до *i*-го сервера за одиницю часу; N – загальна кількість серверів; $\beta_i \in (0; 1/\Lambda)$ – середня тривалість обслуговування запиту; Λ – загальна інтенсивність потоку запитів на кластер; λ_i – інтенсивність потоку запитів, що надходять на *i*-й сервер.

Робота [41] є оглядовим дослідженням, у якому аналізуються такі методи прогнозування навантажень на вебсервер: регресійний, метод ковзного середнього, нелінійні підходи та метод експоненційного згладжування. Відзначається необхідність удосконалення цих підходів для більш точної моделі прогнозування типового навантаження на вебсервер.

У дослідженні [57] розглядається метод прогнозування використання обчислювальних ресурсів вебсервера під час пікових навантажень на основі екстраполяції. Продемонстровано, що рівень завантаження процесора вебсервера залежить від кількості активних процесів. При цьому граничне завантаження процесора не перевищує 0,85. Виявлено періодичний характер завантаження процесора із наявністю екстремальних точок. Запропоноване прогнозування не враховує пікові навантаження, що, може призвести до похибки прогнозування в межах 20-30%.

Також у дослідженні [57] розглядається проблема оцінки потреб у ресурсах для веборієнтованих систем за умов встановленого навантаження. Визначено, що однією з ключових вимог до таких систем є обмеження часу відповіді вебсервера на запити користувачів до 5 секунд. Додатково зазначається, що рівень завантаження процесора не повинен перевищувати 90%. Розглядаються методи визначення потреб у ресурсах на основі пікового навантаження, для оцінки якого рекомендовано проведення додаткових досліджень.

У роботі [28] запропонована методика розрахунку середнього навантаження на вебсервер за умов пуасонівського потоку запитів. Виявлено, що якщо середнє навантаження перевищує 75% від максимально допустимого, то це призводить до різкого зростання середнього часу очікування запитів.

Моделюванню прогнозування навантаження за допомогою методів теорії масового обслуговування присвячена праця [53]. Основна мета моделювання – передбачення роботи вебсервера за різних режимів навантаження. Відзначається необхідність розробки ефективного модуля прогнозування навантаження. Також представлено алгоритм розподілу навантаження при доступі до хмарних систем зберігання, який враховує потреби мультимедійного контенту. Підкреслюється, що навантаження на основні ресурси вебсервера є періодичним і нерівномірним: збільшення потоку запитів обмежується обчислювальними ресурсами, що ускладнює підтримку якісного доступу до мультимедійних додатків. Інтенсивність звернень до ресурсів змінюється залежно від зовнішніх умов, при цьому до 90% навантаження є передбачуваним завдяки попередній реєстрації користувачів. Для

прогнозування обсягів навантаження запропоновано використовувати різні статистичні розподіли: розподіл Парето – для відеотрафіку, розподіл Вейбулла – для статичних даних (бінарний трафік), χ^2-розподіл – для невеликих статичних файлів.

Відома графо-аналітична модель [26] обліку навантаження вебсистем, що включає граф переходів користувачів та математичний опис станів системи. Кожен перехід характеризується часом завантаження сервера та ваговим коефіцієнтом, який відображає відносну частоту запитів (визначається аналітиком системи). Для зменшення навантаження пропонується застосовувати завантажувальне тестування з метою виявлення критичних переходів, класифікації навантаження та вибору ефективних методів його зниження. Також розроблена модель обслуговування користувачів корпоративного вебсервера, згідно з якою основне навантаження генерують запити низького пріоритету. Для прогнозування навантаження запропоновано використання математичного апарату теорії часових рядів, що дозволяє врахувати самоподібність потоку вхідних запитів.

Методологія оцінки навантаження вебсервера, представлена у [106], передбачає використання таких показників: загальна кількість цільової аудиторії, число користувачів у пікові періоди (з розподілом за годинами), частота оновлення даних користувачами, кількість запитів за секунду, а також число транзакцій на один сервер баз даних. Проведено аналіз продуктивності вебсервера Apache у контексті пуасонівського розподілу запитів. Використана модель вебсервера, що включає розподілений процесорний вузол із чергою з n-завдань. Наведено математичний вираз для розрахунку середнього часу обробки запиту X та максимально допустимої кількості запитів K:

$$\left(\hat{X}, \hat{K}\right) = \arg\min_{X,K} \sum_{i=1}^{N} \frac{(\hat{T}_i - T_i)^2}{\hat{\sigma}_i^2}, \tag{1.22}$$

де T_i – середній час модельної відповіді; $\hat{T}_i$ – середній час відповіді за певної інтенсивності запитів λ; i=1..N – номер запиту; $\hat{\sigma}_i^2$ – дисперсія процесу.

Встановлено, що ефективні значення X та K можуть бути визначені за допомогою методу найшвидшого спуску, алгоритму сполучених градієнтів або

шляхом прямого перебору. Зазначено, що традиційні схеми використання вебсервісів демонструють низьку ефективність. Для оптимізації навантаження на канали зв'язку запропоновано застосування кеш-серверів, що не лише зменшує навантаження, а й прискорює отримання інформації користувачем.

Аналіз трафіку центру дистанційного навчання показав, що реальний трафік має самоподібну природу, що суттєво відрізняється від прогнозів, отриманих на основі класичних пуасонівських моделей. Розроблено модель клієнт-серверного навантаження в термінах теорії масового обслуговування. Виведено аналітичні вирази для розрахунку коефіцієнта використання серверних ресурсів та середньої продуктивності сервера з урахуванням самоподібності трафіку. Показано [106], що застосування класичних ланцюгів Маркова та експоненційних моделей для моделювання навантаження навчального вебсервера є малоефективним.

У дослідженні [57] розглянуто методи підвищення ефективності вебсерверів на основі прогнозування часових рядів із використанням нейронних мереж радіально-базисного типу. Для короткострокового прогнозування навантаження на сервер рекомендовано узагальнено-регресійну нейромережеву модель GRNN. Проаналізовано сучасні підходи до балансування та розподілу навантаження вебсервера, доведено їхню недостатню ефективність, оскільки у середньому 40-60% запитів залишаються необробленими. Основною причиною цього є невідповідність поширених моделей прогнозування реальному періодичному характеру навантаження вебсервера.

Доведено [106], що поведінка користувачів у вебмережі може бути описана за допомогою степеневих функцій, популярність вебоб'єктів підпорядковується закону розподілу Зіпфа, а розміри файлів – розподілу Парето. Виявлено кореляцію між послідовними запитами користувача, незалежність запитів різних користувачів до одного файлу та експоненційний характер розподілу часу між такими запитами. Вебтрафік вебсервера має автомодельну природу, демонструючи «пульсуючий» характер при використанні різних часових інтервалів. Виявлено, що вебтрафік містить пікові навантаження, які можуть перевищувати середнє значення у 5-10 разів, що в окремих випадках перевищує обчислювальні можливості сервера.

Також доведено, що навіть незначні коливання трафіку можуть суттєво впливати на продуктивність вебсервера.

Додатково [106] розглянуто алгоритм балансування навантаження CSM (Content Switching Module) для серверного кластера, який здійснює перенаправлення мережевих підключень на сервер із найменшою кількістю активних з'єднань. Підкреслено важливість урахування циклічності під час прогнозування навантаження.

У дослідженні [57] запропоновано модель прогнозування стану сервера, що базується на методах аналізу часових рядів і нечітких експертних оцінок. Представлено алгоритм комплексної оцінки продуктивності сервера, а також метод регресійно-нечіткого моделювання, що дозволяє прогнозувати стан сервера на основі набору різнорідних даних.

Результати, отримані в роботі [106], підтверджують висновки дослідження [57] щодо можливості прогнозування навантаження вебсервера за допомогою багатоперіодичної моделі клієнтських запитів.

У дослідженні [63] запропоновано гібридну модель прогнозування часових рядів, що поєднує можливості Prophet та LSTM, дозволяє досягти підвищеної точності й надійності прогнозів завдяки комбінуванню переваг обох підходів. Prophet, створений командою Facebook, ефективно моделює тренди, сезонність і враховує святкові дні, забезпечуючи зручний інтерфейс навіть для недосвідчених користувачів. У свою чергу, LSTM – це тип рекурентної нейромережі, здатної захоплювати довгострокові залежності в послідовних даних. У гібридній моделі спочатку використовується Prophet для отримання базових прогнозів, після чого розраховуються залишки (різниця між фактичними значеннями і прогнозами), які слугують основою для навчання LSTM. Остаточний прогноз формується шляхом поєднання обох результатів, що дозволяє моделі точно відображати як глобальні тенденції, так і локальні коливання в часовому ряді.

У дослідженні [11] проведено порівняльну оцінку ефективності трьох моделей прогнозування енергоспоживання – SARIMA, LSTM RNN і Facebook Prophet – із базовою моделлю Центрального енергетичного агентства Індії (CEA),

яка базується на трендовому підході. Аналіз проводився для прогнозування сумарного та пікового щомісячного попиту. Результати показали, що хоча модель CEA демонструє прийнятну точність для загального енергоспоживання, вона суттєво недооцінює пікове навантаження. Найкращу точність за всіма метриками показала модель Fb Prophet, що дозволило дослідникам [11] рекомендувати її як основний інструмент для побудови майбутніх трендів прогнозів.

В результаті аналізу доступної спеціалізованої наукової літератури, було визначено, що ефективність прогнозування навантаження на вебсервери пов'язана з реалізацією в них множини базових процедур $\boldsymbol{G}$.

Таблиця 1.3

Характеристики проаналізованих моделей та методів прогнозування навантаження на вебсервери

№	Дослідження	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}
1	Tambe [81]	-	+	-	-	-	-	-	+	-	-
2	Lohrasbinasab [51]	-	-	-	-	-	+	-	-	-	+
3	Mahmood [53]	-	-	-	-	+	-	-	+	-	+
4	Wang [91]	+	+	-	-	-	+	-	-	-	-
5	Tian [85]	-	+	-	-	-	+	-	+	-	+
6	Prajam [66]	-	-	+	-	+	-	-	-	-	+
7	Irie [40]	-	-	+	-	+	-	-	+	-	+
8	Tedjopurnomo [82]	+	-	-	-	-	+	+	+	-	-
9	Mamun [55]	+	-	-	+	-	+	+	-	+	+
10	Saxena [74]	+	-	+	-	+	+	-	+	+	+
11	Palvel [63]	+	-	+	-	+	+	+	-	+	+
12	Chaturvedi [11]	+	+	+	-	+	+	-	-	+	+

В першому наближенні можна вважати, що компонентами ефективного прогнозування навантаження на вебсервер є процедури: врахування невизначеності та зовнішніх факторів в стохастичній моделі прогнозування (G_1),

аналізу часових залежностей і сезонності (G_2), попередньої обробки та нормалізації даних (G_3), вибору вейвлету для багатомасштабного аналізу (G_4), стійкості до шуму (G_5), адаптивної моделі (G_6), оптимізації продуктивності (G_7), комбінування прогнозів з різних моделей (G_8), класифікації запитів для визначення типу навантаження (G_9), оцінки якості прогнозу (G_{10}).

Наявність вказаних процедур в проаналізованих моделях та методах прогнозування навантаження на вебсервер наведено в табл. 1.3.

Сучасні вебсервери працюють в умовах непередбачуваного навантаження, обумовленого як зростанням легітимного трафіку, так і зловмисною активністю (DoS-атаки, спам, сканування тощо). Надмірне навантаження призводить до деградації якості обслуговування, порушення вимог SLA (Service-Level Agreement), або повного припинення надання послуг. Водночас надлишкове резервування обчислювальних ресурсів є економічно неефективним. У таких умовах зростає потреба у точному та адаптивному прогнозуванні навантаження на вебсервери, що дозволяє динамічно керувати ресурсами та запобігати перевантаженню систем.

Попри значну кількість підходів до моніторингу та оцінювання поточного стану серверів, більшість існуючих моделей орієнтовані на фіксацію вже наявних проблем, а не на їх випередження. Типові інструменти базуються на жорстко визначених правилах або лінійних регресійних моделях, які не враховують складну нелінійну структуру навантаження та динамічні стрибкоподібні зміни, що властиві сучасним вебсистемам. У цьому контексті прогнозування на основі традиційних статистичних методів часто демонструє низьку точність і не адаптується до змінних умов середовища.

Аналіз сучасних підходів до прогнозування навантаження на вебсервери свідчить про необхідність удосконалення методів, моделей і засобів, що забезпечують точніше прогнозування змін у навантаженні та оптимізацію розподілу ресурсів. Одним із ключових напрямків розвитку є підвищення точності прогнозування з урахуванням самоподібності трафіку та періодичності змін його інтенсивності.

Висновки до розділу 1

У цьому розділі здійснено огляд основних підходів до оцінки показників навантаження вебсерверів, що зводиться до пошуку ефективного компромісу між вимогами до безпеки та ресурсами, необхідними для їхнього забезпечення. Вимоги визначаються рівнем критичності та обсягами захищеної інформації, умовами її зберігання, передачі, обробки та використання. Натомість ресурси можуть обмежуватися максимальними доступними потужностями або регулюватися необхідністю досягнення заданого рівня захисту.

Окремим напрямом, що демонструє потенціал у сфері обробки нестаціонарних, мультишкальних сигналів, є використання вейвлет-перетворень. На відміну від методів Фур'є, вейвлети дозволяють локалізувати характеристики сигналу як у часовій, так і у частотній області, що дає змогу виділяти тренди, піки навантаження, флуктуації та періодичні аномалії. Проте в поточному науковому дискурсі відсутні системні підходи до застосування вейвлет-перетворень саме для задач прогнозування навантаження вебсерверів. Крім того, мало вивченими залишаються питання вибору оптимального вейвлет-базису для задач такого типу, побудови комбінованих моделей, а також критеріїв оцінювання якості таких моделей у порівнянні з класичними підходами.

У зв'язку з цим, мета даного дослідження полягає у підвищенні ефективності прогнозування навантаження на вебсервери шляхом розробки вейвлет-орієнтованих моделей і програмних засобів, а також демонстрації їх переваг у порівнянні з традиційними методами.

З метою розробки ефективних методів прогнозування необхідно:

- використовувати апроксимацію динаміки параметрів вебтрафіку із застосуванням методів теорії вейвлетів, що дозволить ефективно аналізувати нерівномірні та багатоперіодичні потоки запитів;
- оптимізувати методи балансування навантаження, ґрунтуючись на моделях прогнозування вебтрафіку, що дасть змогу своєчасно реагувати на пікові навантаження;
- використовувати сучасні нейромережеві підходи, зокрема узагальнено-

регресійні нейронні мережі (GRNN), для підвищення точності короткострокового прогнозування навантаження на вебсервер;

- зменшити обчислювальні витрати за рахунок використання спеціалізованих бібліотек та оптимізованих алгоритмів обробки даних;
- розробити ефективний програмний засіб прогнозування, орієнтований на інтеграцію з сучасними вебсерверними системами та операційними середовищами.

Розробка системи прогнозування навантаження передбачає виконання таких етапів:

- формування архітектури системи прогнозування для вебсерверів;
- розробка математичних моделей прогнозування навантаження;
- впровадження алгоритмів аналізу та прогнозування трафіку;
- реалізація програмного забезпечення та тестування його продуктивності;
- експериментальна перевірка ефективності запропонованих методів і моделей.

У межах дисертаційного дослідження необхідно вирішити такі завдання:

- проаналізувати існуючі моделі прогнозування навантаження та виявити їхні обмеження;
- розробити нові математичні моделі прогнозування, що враховують самоподібність трафіку та його багатоперіодичність;
- дослідити доцільність застосування різних типів вейвлетів (наприклад, Добеші, Морле, Койфлети) у задачах прогнозування вебнавантаження;
- запропонувати ефективні алгоритми розподілу навантаження на основі зібраних статистичних даних для прогнозування;
- розробити програмний комплекс, що реалізує запропоновані методи та моделі, та провести його тестування на реальних вебсерверах.

Комплексне вирішення зазначених завдань дозволить створити систему прогнозування навантаження на вебсервер, що сприятиме підвищенню продуктивності серверних ресурсів, забезпеченню стабільності їхньої роботи та покращенню якості обслуговування користувачів на базі методів теорії вейвлетів.

РОЗДІЛ 2. РОЗВИТОК МЕТОДОЛОГІЧНОЇ БАЗИ ПРОГНОЗУВАННЯ НАВАНТАЖЕННЯ НА ВЕБСЕРВЕР

2.1. Концептуальна модель процесу прогнозування навантаження на вебсервер

Як свідчать результати першого розділу одним із ключових аспектів забезпечення ефективності функціонування комп'ютерних систем і мереж є розробка ефективних методів прогнозування навантаження на вебсервер. Складність вирішення цього завдання значною мірою обумовлена недоліками існуючого методологічного підґрунтя. Саме тому дослідження, спрямовані на впровадження сучасних теоретичних підходів до прогнозування навантаження, є актуальними та важливими. Питання прогнозування навантаження та моделювання ефективної роботи вебсерверів при різних типах навантаження розглядаються у численних наукових публікаціях, у яких автори аналізують проблему з урахуванням специфіки функціонування вебсервера.

Водночас проведений аналіз досліджуваного аспекту у попередньому розділі дозволяє стверджувати, що в наявних джерелах питання адаптації типу базисного вейвлета до умов формування шаблону нормальної поведінки вебсервера розглянуто лише фрагментарно: у більшості робіт обмежуються загальним описом вейвлет-базисів без формалізації критеріїв їх відповідності особливостям вебтрафіку, не пропонуються методи автоматизованого вибору оптимального базису, а результати апробації, як правило, ґрунтуються на синтетичних, а не реальних даних. Цей науковий пробіл у даній роботі пропонується усунути шляхом розробки концептуальної моделі, що дозволяє обирати ефективний тип базисного вейвлета як основу для створення шаблону нормальної поведінки вебсервера, з урахуванням системи критеріїв ефективності, які будуть розкриті при описі відповідного методу.

Концептуальна модель процесу прогнозування навантаження на вебсервер являє собою абстрактне подання процесу, яке описує основні сутності, взаємозв'язки та інформаційні потоки між компонентами системи

прогнозування. У межах цієї моделі визначаються ключові етапи – збір даних із серверних логів, попередня обробка сигналів, вейвлет-декомпозиція, формування ознак, навчання прогнозувальної нейронної мережі та оцінювання точності. Концептуальна модель відображає логічну структуру процесу, дозволяючи зрозуміти що саме виконує система та як пов’язані її функціональні блоки.

У загальному розумінні концептуальна модель є відображенням предметної області, що включає сукупність взаємопов’язаних понять, які характеризують цю галузь, а також їх властивості, ознаки, класифікаційні категорії, типові ситуації та закономірності протікання відповідних процесів [102, 113]. Вона базується на певній концепції – сукупності суджень і підходів до інтерпретації явищ, які відображають основну ідею чи точку зору для системного висвітлення об’єкта дослідження.

Оскільки практичним результатом побудови концептуальної моделі має стати програмно-апаратне рішення для прогнозування навантаження на вебсервер, оцінювання ефективності цього процесу доцільно здійснювати з урахуванням визначень, прийнятих у комп’ютерній та програмній інженерії [125]. Згідно з міжнародними стандартами в цій галузі, ефективність визначається як сукупність атрибутів, що характеризують співвідношення між рівнем виконання програмної системи, використанням ресурсів (фінансових, апаратних, матеріальних) та якістю послуг, що надаються обслуговуючим персоналом.

До ключових показників ефективності програмно-апаратного комплексу належать:

– оперативність – відображає швидкість реакції системи, час обробки запитів і виконання функцій;

– ресурсоємність – характеризує обсяг і тривалість використання ресурсів під час роботи системи;

– узгодженість – демонструє відповідність системи чинним стандартам, правилам та нормативним вимогам.

Відповідно до наведених визначень, на початковому етапі створення концептуальної моделі було здійснено гармонізацію термінологічного апарату, який використовується в сфері прогнозування навантаження на вебсервери загального призначення. У результаті цього процесу були сформульовані ключові поняття, що відображають основні цілі та переваги впровадження систем прогнозування (рис. 2.1), а саме:

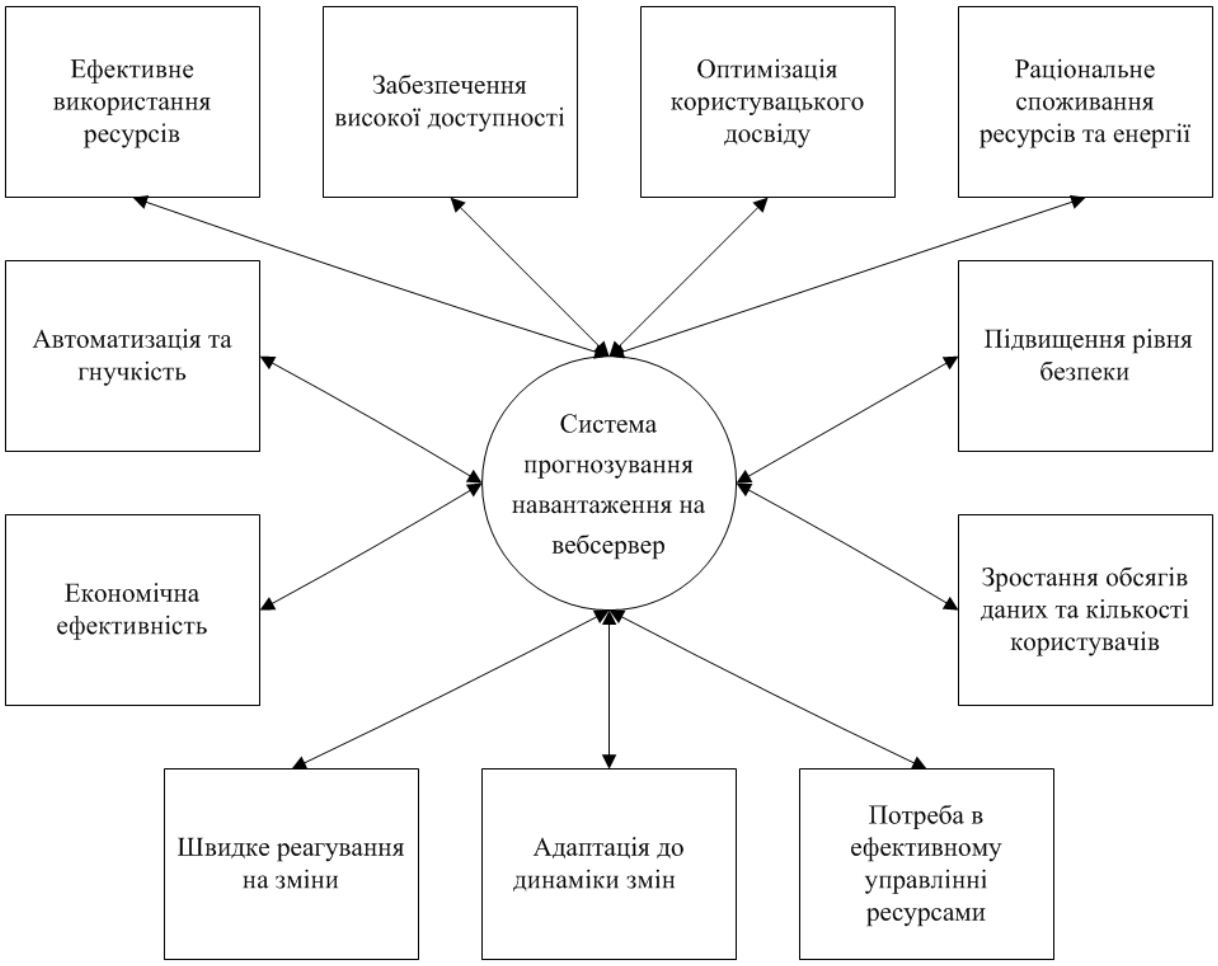

Рис. 2.1. Ключові поняття концептуальної моделі процесу прогнозування навантаження на вебсервер

– ефективне використання ресурсів – прогнозування навантаження дозволяє оптимізувати розподіл ресурсів вебсервера, запобігати перевантаженням, забезпечувати безперебійну роботу сервісів та ефективне масштабування відповідно до прогнозованих змін, що сприяє зниженню

експлуатаційних витрат;

– забезпечення високої доступності – завдяки прогнозуванню адміністратори можуть своєчасно реагувати на потенційні пікові навантаження, що знижує ризик збоїв і підвищує надійність та стабільність функціонування вебсервісів;

– оптимізація користувацького досвіду – заздалегідь підготовлена система гарантує швидкий доступ до ресурсів, що покращує задоволення користувачів, мінімізує затримки та сприяє плавній роботі вебзастосунків;

– раціональне споживання ресурсів та енергії – точне прогнозування зменшує потребу в постійному утриманні надлишкових серверних потужностей, що дозволяє скоротити витрати на енергоспоживання та інфраструктуру;

– підвищення рівня безпеки – аналіз навантаження сприяє своєчасному виявленню аномальної активності або потенційних кібератак, що дозволяє вжити заходів для підтримання стабільності та захисту системи;

– зростання обсягів даних та кількості користувачів – із розвитком цифрових технологій та зростанням популярності вебзастосунків, соціальних мереж і хмарних сервісів спостерігається постійне збільшення навантаження на вебресурси;

– потреба в ефективному управлінні ресурсами – оптимізація використання серверних потужностей дозволяє забезпечити високу продуктивність вебсервісу при раціональному використанні доступних ресурсів;

– адаптація до динаміки змін – коливання трафіку, зумовлені сезонними чи поведінковими факторами, вимагають постійного моніторингу та адаптивного прогнозування для збереження стабільної роботи системи;

– швидке реагування на зміни – здатність оперативно змінювати конфігурацію ресурсів у реальному часі відповідно до навантаження є критичною для запобігання як перевантаженням, так і простою ресурсів;

– економічна ефективність – завдяки прогнозуванню можливо знизити фінансові витрати, особливо при використанні хмарних технологій, де оплата здійснюється відповідно до обсягу спожитих ресурсів;

– автоматизація та гнучкість – системи прогнозування можуть бути інтегровані у процеси автоматизованого управління інфраструктурою вебсервера, а також адаптовані до агільних підходів у розробці, що забезпечує швидку реакцію на зміну вимог і навантаження.

Якість прогнозування визначається не лише точністю обраних моделей, а й здатністю системи адаптуватися до змін та ефективно впроваджувати результати для оптимізації використання ресурсів і підвищення продуктивності.

У зв’язку з цим концептуальна модель відіграє ключову роль як засіб формалізації причинно-наслідкових залежностей у процесі прогнозування навантаження. Вона покликана сприяти підвищенню ефективності управління вебінфраструктурою в умовах динамічного цифрового середовища.

Для досягнення високої ефективності процесу прогнозування в концептуальну модель доцільно інтегрувати такі практичні стратегії (рис. 2.2):

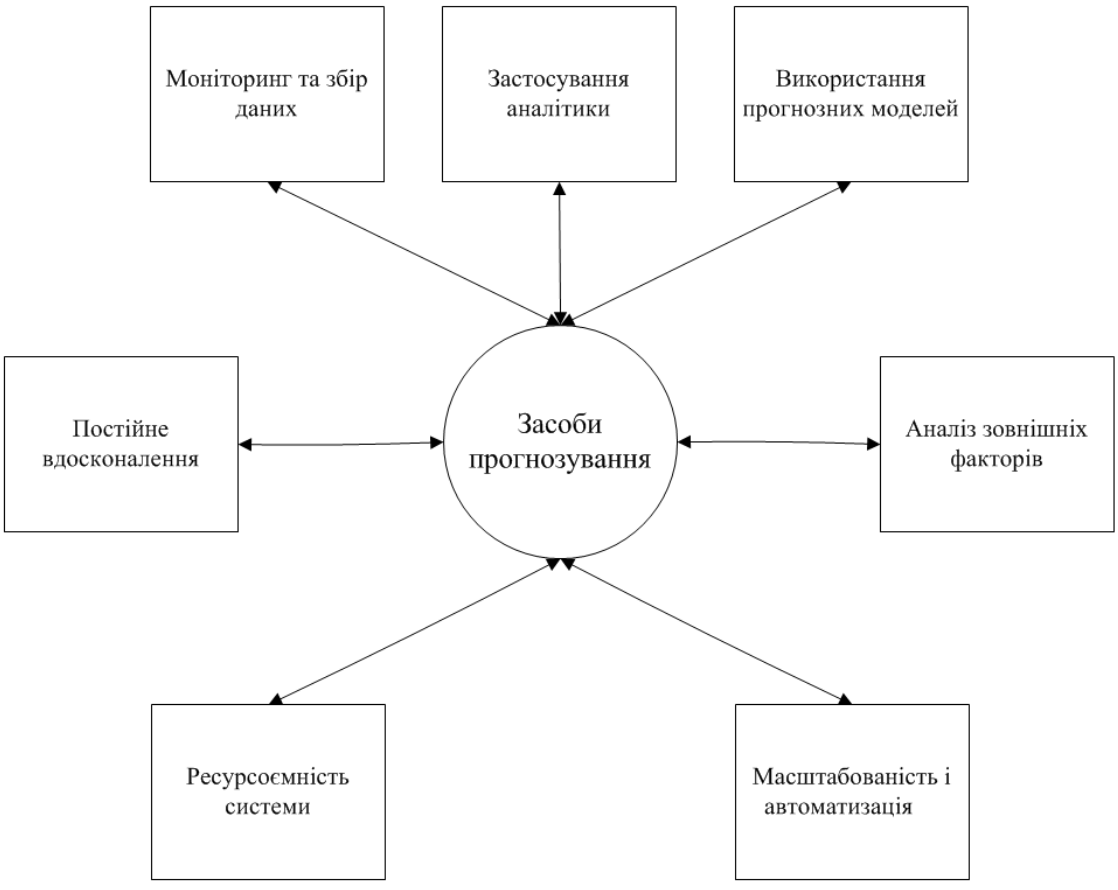

Рис. 2.2. Фактори впливу на ефективність засобів прогнозування

– моніторинг та збір даних – впровадження інструментів для безперервного збору інформації про використання ресурсів, трафік, навантаження та інші важливі метрики. Регулярне оновлення та накопичення історичних даних забезпечує надійну базу для побудови прогнозних моделей;

– застосування аналітики – аналіз поведінкових патернів користувачів, виявлення трендів, сезонних коливань та інших факторів, що впливають на зміну навантаження. Особливу увагу слід приділяти метрикам відвідуваності, конверсії тощо;

– використання прогнозних моделей – застосування методів аналізу часових рядів і машинного навчання (лінійна регресія, нейронні мережі, ансамблеві методи) для точного передбачення майбутніх змін у навантаженні;

– аналіз зовнішніх факторів – врахування подій поза межами системи (рекламні кампанії, інформаційні події, святкові періоди), що можуть впливати на рівень трафіку. Кореляційний аналіз дозволяє оцінити вплив подібних факторів;

– масштабованість і автоматизація – впровадження механізмів автоматичного масштабування для динамічного управління ресурсами сервера на основі прогнозів [126]. Автоматизовані процеси забезпечують гнучкість і оперативність у реагуванні на зміни навантаження;

– ресурсоємність системи – використання інструментів для рівномірного розподілу трафіку між серверами, а також CDN (мереж доставки контенту) для зменшення навантаження на основний сервер і підвищення швидкодії;

– постійне вдосконалення – регулярне оновлення моделей прогнозування з урахуванням нових даних і змін у функціонуванні системи. Важливо залучати зворотний зв’язок від розробників, адміністраторів і кінцевих користувачів для підвищення точності та актуальності прогнозів.

Таким чином, ефективне прогнозування навантаження на вебсервер потребує цілісного підходу, який поєднує технічні рішення, адаптивні стратегії та системне розуміння особливостей функціонування вебінфраструктури.

У результаті дослідження встановлено, що концептуальну модель забезпечення ефективного прогнозування навантаження на вебсервер можна

подати у вигляді аналітичного виразу:

$$E = f(E_A, E_B, E_C), \tag{2.1}$$

де E – інтегральна ефективність прогнозування навантаження на вебсервер; E_A – ефективність проєктування моделі прогнозування; E_B – ефективність впровадження моделі прогнозування; E_C – ефективність динамічної реєстрації вебтрафіку.

Після деталізації складових змінних моделі були отримані наступні вирази:

$$E_A = f(T_m, R), \tag{2.2}$$

$$E_B = f(P_s, D_c), \tag{2.3}$$

$$E_C = f\left(Q_s, Q_f, N_a\right), \tag{2.4}$$

де T_m – якість вибору типу моделі (Model Type Selection); R – якість налаштування параметрів моделі (Model Parameters Tuning); P_s – ефективність формування параметрів навчальних прикладів (Preprocessing Success); D_c – ефективність створення навчальної вибірки (Dataset Construction); Q_s – ефективність визначення набору запитів для реєстрації (Query Selection); Q_f – ефективність фільтрації зареєстрованих запитів (Query Filtering); N_a – ефективність нейромережевого аналізу запитів (Neural Analysis) з метою прогнозування поведінки вебсервера.

Зазначені компоненти охоплюють ключові етапи, що визначають загальну ефективність прогнозної моделі та її здатність реагувати на зміни у вебтрафіку.

На ефективність проєктування E_A впливають такі групи факторів як якість даних, доступних на етапі проєктування (повнота, точність, репрезентативність), рівень автоматизації процесу вибору T_m та налаштування R (використання AutoML, байєсівської оптимізації тощо), компетентність розробника моделі (досвід у побудові прогнозних систем для динамічних і стохастичних процесів).

На ефективність впровадження E_B суттєво впливають якість обробки даних на етапі формування навчальних прикладів (повнота врахування усіх значущих чинників), вибір стратегії побудови вибірки (метод ковзного вікна, рекурентна побудова тощо), рівень автоматизації процесу формування та оновлення навчальних даних (особливо важливо для систем із самооновленням моделей).

На ефективність динамічної реєстрації E_C впливають якість визначення релевантних запитів, ступінь очищення даних від шуму, потужність застосованих алгоритмів нейроаналізу, часова затримка між реєстрацією та прогнозуванням (вона має бути мінімальною).

Таким чином, остаточна форма інтегральної ефективності набуває вигляду:

$$E = F\left(f_A(T_m, R), f_B(P_s, D_c), f_C\left(Q_s, Q_f, N_a\right)\right), \quad (2.5)$$

де F – оператор агрегації, який є багатокритеріальною функцією інтеграції часткових оцінок.

Вибір типу моделі T_m має на меті знайти архітектуру, яка найкращим чином відображає закономірності у вебнавантаженні. Основні критерії вибору типу моделі включають: адаптивність до зміни трафіку та наявність сезонних/трендових компонент, можливість врахування нелінійних залежностей між характеристиками запитів і навантаженням, обчислювальна ефективність для наближеного реального часу, інтерпретованість результатів у випадку пояснення прогнозу.

Формально, вибір T_m можна розглядати як процес оптимізації функції відповідності:

$$T^* = arg \max_{T_m} Q_T, \quad (2.6)$$

де Q_T – функція якості моделі за вибраним набором метрик (MSE, R^2, обчислювальні витрати тощо).

В межах поточного аналізу докладно розпишемо формалізовані критерії оцінки $R = [r_{ij}]$ для параметрів моделі; визначимо значущості кожного критерію.

Сформулюємо зміст кожного критерія (Табл. 2.1): r_1 – наявність в вейвлет-коефіцієнтах надлишкової інформації; r_2 –можливість реалізації швидкого вейвлет-перетворення поєднана з емпіричним підрахунком швидкодії; r_3 – асиметричність базисної функції; r_4 – можливість масштабування функції; r_5 – компактність базисної функції; r_6 – характеристика спектрального розподілу енергії вейвлет-функції; r_7 – можливість повного відновлення сигналу; r_8 – скупість, як частка значущих коефіцієнтів істотно відмінних від нуля; r_9 – покращення співвідношення сигнал/шум; r_{10} – ступінь геометричної подібності базисної функції до характеру

аналізованого процесу.

Під час побудови моделі, за аналогією з підходами, описаними в [33, 106], було прийнято, що для першого наближення значення критеріїв з r_1 по r_9 можуть бути оцінені за шкалою [0–1]. Згідно з цією шкалою, для i-го типу базисного вейвлету значення k-го критерію дорівнює 1, якщо відповідна вимога виконується повністю, і 0 – якщо не виконується.

Таблиця 2.1

Критерії ефективності r_i базисних вейвлетів із ваговими коефіцієнтами w_i

Критерій	Опис	Позначення	Вага
r_1	Надлишковість вейвлет-коефіцієнтів	Redundancy	0.05
r_2	Швидкість вейвлет-перетворення	Fastness	0.10
r_3	Асиметричність базисної функції	Asymmetry	0.05
r_4	Масштабованість функції	Scalability	0.05
r_5	Компактність базисної функції	Compactness	0.05
r_6	Енергетична інформативність	Energy Ratio	0.20
r_7	Можливість повного відновлення сигналу	Reconstruction	0.20
r_8	Скупість значущих вейвлет-коефіцієнтів	Sparsity	0.10
r_9	Ефективність шумозаглушення	Noise Reduction Efficiency	0.15
r_{10}	Геометрична подібність	Similarity	0.05
	Сума ваг:		1.00

Критерій r_{10} оцінює наскільки форма вейвлет-базису – графік відповідної базисної функції $\psi(t)$ – відповідає характеру змін у часовому ряді вебтрафіку $x(t)$ (наявність імпульсів, періодичності, різких стрибків, асиметрії тощо).

Як метод оцінювання проводиться метричне порівняння графіка використовуючи формулу нормалізованої кореляції:

$$r_{10} = \max_{\tau}\left(\frac{\sum_t (x(t)-\bar{x})(\psi(t-\tau)-\bar{\psi})}{\sqrt{\sum_t (x(t)-\bar{x})^2}\sqrt{\sum_t (\psi(t-\tau)-\bar{\psi})^2}}\right), \qquad (2.7)$$

де τ – зсув (для пошуку найкращої локальної відповідності), $\bar{x}$, $\bar{\psi}$ – середні значення відповідних функцій.

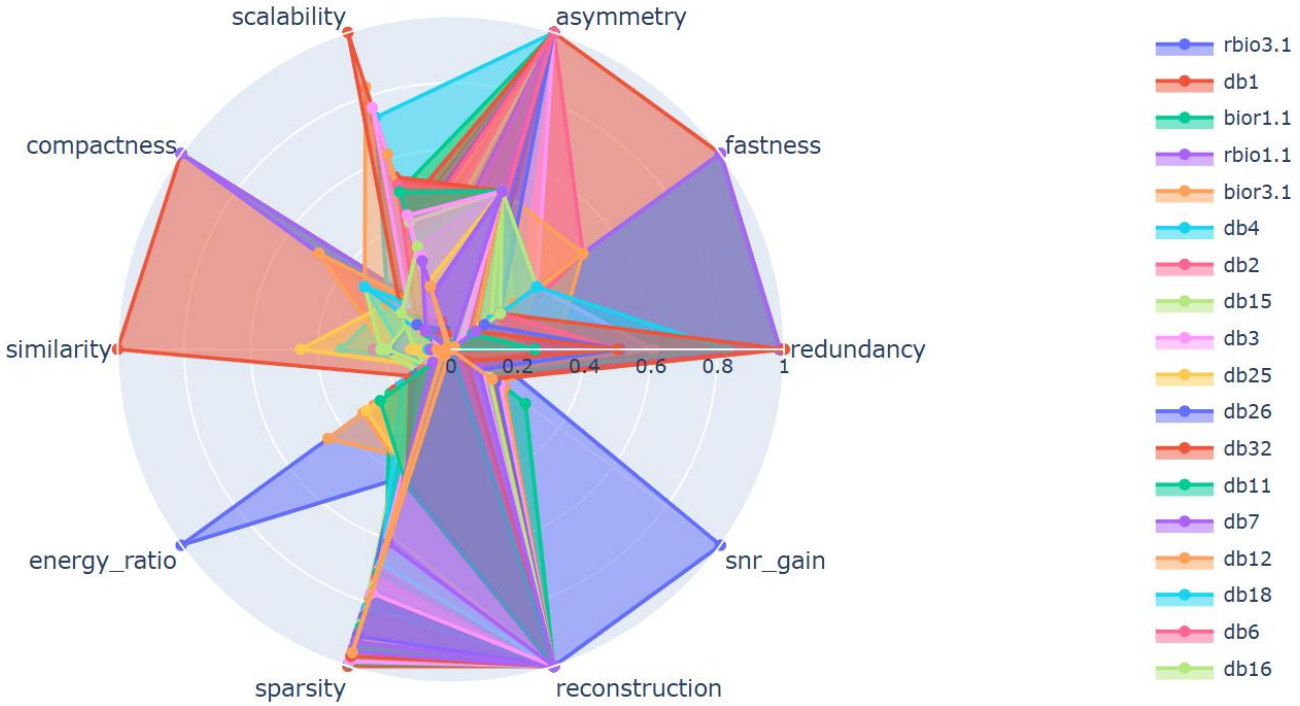

Рис.2.3. Інтерактивна радарна діаграма для багатокритеріальної оцінки вейвлетів

Рисунок 2.3 демонструє інтерактивну радарну діаграму, призначену для багатокритеріальної оцінки та ранжування вейвлетів у межах методології MCDM ранжування. Діаграма наочно відображає значення кожного з критеріїв r_i, що характеризують властивості вейвлет-базисів. Інтерактивність інструмента дозволяє досліднику динамічно порівнювати альтернативні вейвлети, коригувати вагові коефіцієнти критеріїв та в режимі реального часу оцінювати вплив кожного параметра на підсумкове ранжування.

Формування параметрів навчальних прикладів P_s передбачає перетворення вихідних даних про запити користувачів у формат, придатний для навчання моделі. До основних процедур формування належать:

– попередня обробка даних: нормалізація або стандартизація вхідних ознак;

інженерія ознак (генерація нових характеристик, наприклад, темп приросту запитів, середня кількість запитів за інтервал); обробка пропущених або некоректних значень;

– декомпозиція сигналу (наприклад, за допомогою вейвлет-перетворення) для виділення трендової та високочастотної складових вебтрафіку;

– виявлення та видалення аномалій, які можуть викликати перенавчання або знижувати якість прогнозу.

Формально, параметри навчальних прикладів можна подати як результат функціонального перетворення вихідного потоку запитів:

$$P_s = g(X), \tag{2.8}$$

де g – оператор попередньої обробки.

Ефективність P_s оцінюється за такими критеріями як ступінь підвищення точності прогнозу після обробки, обчислювальна складність процедури підготовки даних.

Створення навчальної вибірки D_c полягає у побудові послідовностей вхідних та цільових даних для навчання моделі прогнозування. Важливими аспектами цього процесу є:

– балансування вибірки: правильне представлення періодів високого, середнього та низького навантаження, уникнення зміщення навчання лише на часті сценарії (наприклад, тільки денне навантаження);

– врахування часової залежності: формування ковзних вікон для створення прикладів (input–output pairs), правильний вибір горизонту прогнозування (короткостроковий, середньостроковий або довгостроковий прогноз);

– забезпечення репрезентативності даних для відображення реальної роботи вебсервера (сезонність, стрибки трафіку, святкові періоди).

Формально навчальну вибірку можна описати як набір:

$$D_c = \{(x_i, y_i)\}_{i=1}^{N}, \tag{2.9}$$

де x_i – вхідний вектор характеристик (зокрема, із урахуванням часових лагів і перетворень), y_i – відповідне цільове значення (кількість запитів через заданий час).

Ефективність D_c оцінюється через середню похибку на тестовій вибірці, рівень узагальнення моделі на нових даних.

Реєстрація усіх запитів без розбору може призвести до надмірних обчислювальних витрат і зниження продуктивності системи. Тому необхідним є визначення релевантного набору запитів Q_s для моніторингу. Основні аспекти цього етапу: вибір запитів, які мають суттєвий вплив на навантаження сервера (динамічні сторінки, великі файли, API-запити); врахування характеристик запиту (тип ресурсу, розмір відповіді, код відповіді сервера, тривалість обробки запиту).

Формалізація задачі:

$$Q_s = \{q_i \in Q \mid I(q_i) > \theta\}, \tag{2.10}$$

де Q – множина усіх запитів, $I(q_i)$ – інформативність запиту q_i, θ – порогове значення інформативності.

Ефективність Q_s визначається співвідношенням кількості зареєстрованих запитів до їх внеску у загальне навантаження.

Після збору даних необхідним є проведення фільтрації для очищення від шумових запитів (наприклад, технічних, службових запитів, запитів ботів), спотворених або неповних записів, аномалій, що не відповідають нормальній поведінці користувачів.

Фільтрація зареєстрованих запитів здійснюється за такими критеріями як допустимі діапазони тривалості запиту; відповідність коду відповіді HTTP (наприклад, виключення кодів 404, 500); контроль за надмірною частотою запитів із однієї IP-адреси.

Формально, множина запитів, що пройшли фільтрацію:

$$Q_f = \{q_i \in Q \mid PassFilter(q_i) = 1\}, \tag{2.11}$$

Ефективність Q_f оцінюється через ступінь зниження шуму без втрати релевантної інформації.

Останнім етапом обробки зареєстрованого вебтрафіку є нейромережевий аналіз N_a, який дозволяє виявити приховані закономірності у поведінці користувачів, кластеризувати запити за подібними характеристиками навантаження, будувати короткострокові локальні прогнози на основі поточних

шаблонів запитів.

Нейромережеві методи аналізу можуть включати у себе автоенкодери для виявлення аномалій у запитах, класифікаційні моделі для визначення класу навантаження запиту, рекурентні нейронні мережі для моделювання послідовностей запитів у часі.

Формально результат аналізу можна подати як прогнозований профіль навантаження:

$$\hat{L}(t) = N_a\left(Q_f(t)\right), \tag{2.12}$$

де $\hat{L}(t)$ – прогноз навантаження у момент часу t.

Ефективність N_a оцінюється за показниками точності прогнозу на основі оперативних даних запитів.

Результати, представлені в роботах [27, 57, 106], свідчать про доцільність зосередження уваги на адаптації вейвлетів з найбільш перспективними базисними функціями для вирішення задач прогнозування навантаження на вебсервер. Крім того, аналіз існуючих джерел показав відсутність сформованої методики, яка б передбачала створення та застосування вейвлетів, орієнтованих на виявлення пікових значень навантаження у подібних системах.

Оскільки дані навантаження мають часову природу, задачу прогнозування навантаження на вебсервер доцільно розглядати як функцію часу. Аналітична модель запропонованого підходу має вигляд:

$$\langle \boldsymbol{S}, \boldsymbol{IO}, \boldsymbol{D}, \boldsymbol{K}, \boldsymbol{K_t}, \boldsymbol{K_s}, \boldsymbol{Z_P}, T, W_n \rangle \rightarrow \langle \boldsymbol{Z} \rangle, \tag{2.13}$$

$$\boldsymbol{Z} = \left\{\boldsymbol{z}^{(\phi_k)}\right\}_{K_\phi} = \left\{\boldsymbol{x}^{(\phi_k)}, y^{(\phi_k)}\right\}_{K_\phi}, \tag{2.14}$$

де $\boldsymbol{S}$ – множина типів серверів, які перебувають під спостереженням, $\boldsymbol{K}$ – множина зареєстрованих клієнтів, $\boldsymbol{IO}$ – множина співвідношень вхідного/вихідного трафіку з урахуванням типу дня (робочий, вихідний, передсвятковий, святковий, рухоме свято тощо), $\boldsymbol{D}$ – множина неочікуваних збурень у вебтрафіку згідно зібраної статистики, $\boldsymbol{K_t}$ – множина середньої тривалості сеансів користування вебресурсом, $\boldsymbol{K_s}$ – множина середньої кількості переглядуваних вебсторінок, T – тренд зміни навантаження протягом заданого періоду, $\boldsymbol{Z}$ – статистична вибірка, яка відображає

періодичні закономірності у трафіку (добові, тижневі, річні цикли), $\boldsymbol{Z_P}$ – множина обмежень, що накладаються при формуванні прогнозованої вибірки, W_n – обчислювальні ресурси вебсервера, $\boldsymbol{z}^{(\phi_k)}$ – нормалізовані приклади з вибірки $\boldsymbol{Z}$ для k-го елементу, $\boldsymbol{x}^{(\phi_k)}$ – множина вхідних параметрів для k-го елементу, $y^{(\phi_k)}$ – очікуване вихідне значення (прогнозоване навантаження) для k-го елементу, K_ϕ – кількість елементів, що враховуються у розрахунках.

Окрім цього, встановлено, що процес побудови ефективної та надійної системи прогнозування включає ряд послідовних етапів (рис. 2.4), серед яких:

– формулювання цілей і вимог – визначення основної мети системи (забезпечення стабільності, оптимізація ресурсів, адаптація до змін у трафіку) та технічних вимог (точність прогнозування, швидкість реагування тощо);

– збір і попередня обробка даних – організація збору інформації про вебтрафік за допомогою журналів сервера та інструментів моніторингу, очищення, нормалізація й агрегація даних для подальшого аналізу;

– вибір методів прогнозування – аналіз і підбір релевантних підходів: методів обробки часових рядів, алгоритмів машинного навчання, статистичних моделей, вейвлет-декомпозиції тощо – з урахуванням специфіки трафіку та інфраструктури вебсервера;

– розробка прогнозних моделей – побудова моделей, навчання на зібраних даних, валідація результатів і коригування параметрів для досягнення прийнятної точності;

– інтеграція з системою моніторингу – впровадження розробленої моделі у діючу систему спостереження з метою автоматичного збору нових даних і оперативного оновлення прогнозів;

– автоматизація процесу прогнозування – реалізація безперервного циклу навчання моделі та генерації прогнозів із можливістю динамічного масштабування ресурсів на основі прогнозованого навантаження;

– аналіз результатів і валідація – порівняння отриманих прогнозів із фактичними значеннями, обчислення метрик точності, виявлення відхилень і здійснення необхідних коригувань;

– розробка механізмів адаптивної оптимізації – забезпечення можливості самостійного коригування моделі відповідно до змін у трафіку та ресурсному стані сервера для збереження точності прогнозів;

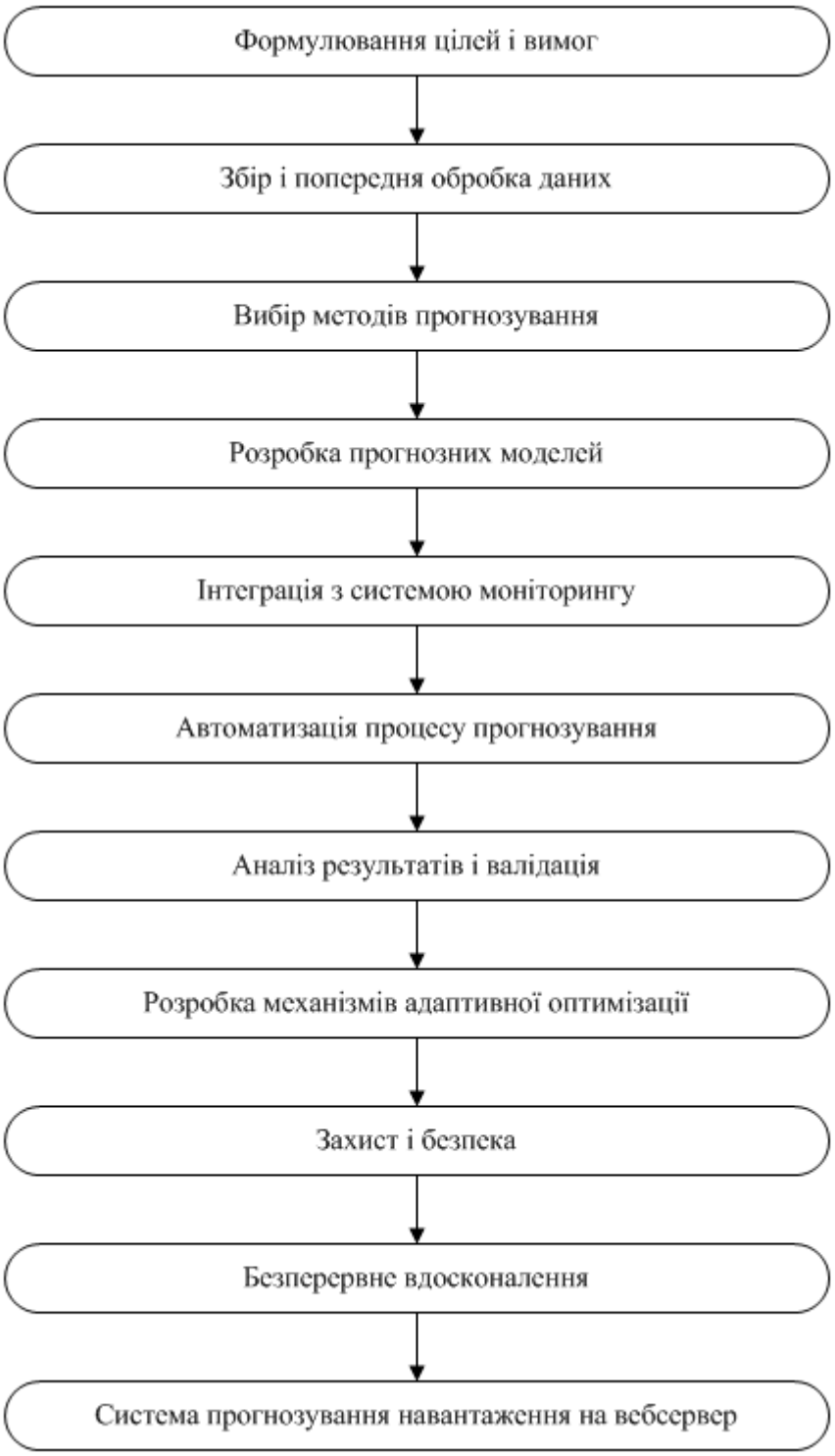

Рис. 2.4. Етапи розробки системи прогнозування навантаження на вебсервер

– захист і безпека – реалізація заходів із забезпечення цілісності даних, захисту від несанкціонованого доступу та врахування вимог конфіденційності при роботі з трафіком;

– безперервне вдосконалення – постійне оновлення моделей, адаптація до змін середовища, технічного прогресу та нових закономірностей у поведінці користувачів.

Загалом, побудова ефективної системи прогнозування навантаження на вебсервер вимагає системного підходу, регулярного моніторингу та постійної оптимізації її компонентів з урахуванням нових даних і змін у цифровій інфраструктурі.

Варто відзначити, що при застосуванні цієї концептуальної моделі для побудови ефективної системи прогнозування навантаження на вебсервер необхідно брати до уваги тривалість виконання окремих процесів, їхню ресурсоємність, точність, а також обсяг фінансових і технічних ресурсів, що виділяються на розробку та впровадження таких рішень.

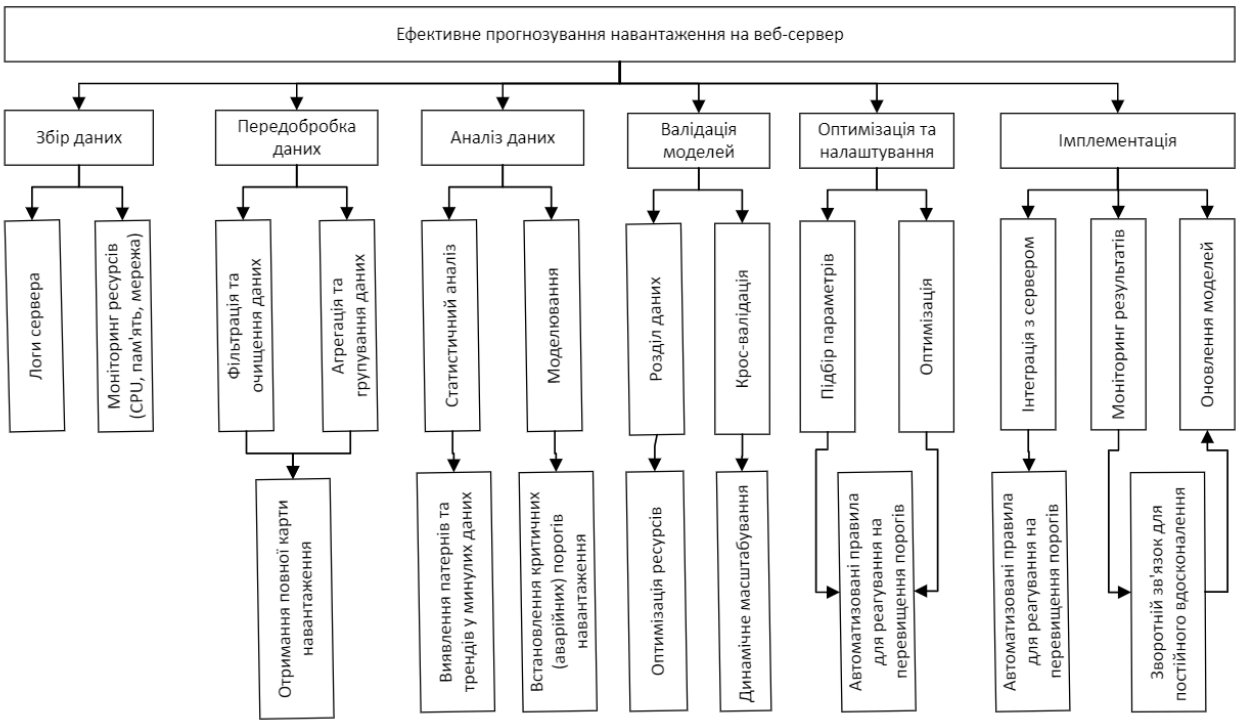

Рис. 2.5. Структурна схема концептуальної моделі прогнозування навантаження на вебсервер

Архітектура моделі тісно пов'язана з набором параметрів, які впливають на рівень навантаження на вебсервер. Зважаючи на характерні задачі, які виконує сервер загального призначення у комп'ютерній мережі, а також враховуючи використовувані мережеві протоколи, можна виділити три основні групи впливових параметрів: апаратні ресурси сервера, ресурси операційної системи, а також мережеві ресурси.

Наступним кроком у розробці концептуальної моделі стала розробка структурної схеми, зображеної на рис. 2.5, яка відображає ключові аспекти організації процесу прогнозування навантаження, де центральним джерелом даних є вебтрафік, що надходить на сервер.

Модель передбачає такі етапи:

– збір і аналіз даних вебтрафіку – використання логів сервера (HTTP-запити, відповіді, час обробки) для отримання детальної інформації про характер трафіку та його зміну в часі;

– моніторинг метрик використання ресурсів – фіксація показників роботи CPU, оперативної пам’яті, мережевого інтерфейсу та інших системних ресурсів, які безпосередньо залежать від інтенсивності вебтрафіку;

– агрегація даних вебтрафіку та системних метрик – поєднання інформації з логів і моніторингу в єдину часову шкалу;

– статистичний аналіз вебтрафіку – виявлення пікових періодів, трендів і аномалій на основі накопичених часових рядів;

– прогнозування параметрів вебтрафіку – застосування алгоритмів прогнозування часових рядів (включно з вейвлет-перетворенням) для передбачення майбутнього навантаження;

– моделі машинного навчання – оцінка впливу зовнішніх факторів (час доби, день тижня, святкові дні) на зміну інтенсивності трафіку;

– оптимізація ресурсів і масштабування – впровадження механізмів автоматичного масштабування ресурсів залежно від очікуваного рівня навантаження;

– балансування навантаження – реалізація технологій рівномірного

розподілу трафіку між серверами, включаючи можливість використання CDN;

– визначення порогових значень навантаження – встановлення критичних меж, перевищення яких свідчить про перенавантаження системи;

– автоматизоване реагування – налаштування автоматичних сценаріїв дій при перевищенні порогів, таких як масштабування чи надсилання сповіщень адміністраторам;

– механізми зворотного зв’язку – збір інформації після впровадження прогнозу для коригування моделей і покращення точності передбачень;

– оновлення параметрів моделей – регулярна адаптація алгоритмів і параметрів на основі нових вхідних даних та умов функціонування системи.

Сучасні вебсистеми функціонують в умовах динамічного та часто непередбачуваного трафіку, який залежить від поведінкових патернів користувачів, подій у зовнішньому середовищі, сезонності та інших стохастичних факторів. Відповідно, завдання забезпечення стабільної продуктивності вебсервера потребує не тільки потужних апаратних ресурсів, але й інтелектуальних механізмів моделювання та прогнозування навантаження. Одним із провідних підходів, що швидко розвивається у світовій науковій спільноті, є використання цифрових двійників (Digital Twins) – віртуальних реплік фізичних або логічних систем, які відтворюють їх поведінку у режимі, максимально наближеному до реального часу. У контексті вебсервера цифровий двійник виступає як адаптивна, самооновлювана модель, що інтегрує дані моніторингу, алгоритми обробки сигналів та системи машинного навчання для передбачення змін у робочому навантаженні.

Концепція цифрового двійника у цій сфері передбачає три ключові компоненти: цифрову модель, канал даних та механізм обчислювального прогнозування. Цифрова модель вебсервера включає структурні характеристики (архітектуру обробки запитів, пул потоків, кешування, мережеві параметри) та динамічні властивості (поточний RPS, час обробки запитів, завантаження CPU, використання пам'яті). Канал даних забезпечує постійне надходження телеметрії у вигляді логів, метрик продуктивності, трасувань системних подій.

Найважливішою складовою є модуль прогнозування, що виконує апроксимацію змін навантаження з використанням таких методів, як вейвлет-декомпозиція, WNN-моделі, LSTM-мережі або їх гібридні варіанти.

Особливу роль відіграє аспект наближеного до реального часу. Цифровий двійник вебсервера не є просто постфактум-аналітичною моделлю; він має реагувати на зміни вхідних потоків з мінімальною латентністю, близькою до часових характеристик самої системи. Це передбачає обробку сигналу у квазіонлайн режимі, оптимізацію вейвлет-перетворень для швидкої декомпозиції, використання легких моделей ML або схем інкрементального навчання. Усі ці елементи гарантують, що цифровий двійник підтримує актуальний стан фізичного сервера та здатен прогнозувати поведінку системи з горизонтом у кілька секунд чи хвилин, що є критичним для систем з високими вимогами до SLA (Service Level Agreement, гарантований постачальником рівень якості сервісу).

Формування цифрового двійника вебсервера забезпечує не лише оперативне прогнозування навантаження, але й створює основу для проактивного керування ресурсами. Наприклад, завдяки прогнозній моделі можливо завчасно ініціювати масштабування у хмарній інфраструктурі, оптимізувати балансування навантаження, змінювати параметри кешування або активувати додаткові механізми захисту у разі наближення аномальних піків трафіку. Така інтеграція є втіленням принципів кіберфізичних систем, де віртуальна компонента не лише аналізує, але й впливає на стан фізичної системи.

Наукова цінність концепції цифрового двійника вебсервера у моделі прогнозування навантаження полягає у створенні комплексної, багаторівневої системи, що поєднує методи інтелектуальної аналітики, сучасні підходи до обробки сигналів та реалістичне моделювання вебінфраструктури. Завдяки інтеграції вейвлетних перетворень як інструменту виділення характеристик сигналу та глибинних нейромереж як предикторів, цифровий двійник здатен виявляти складні патерни, маскування трендів та хвилеві структури навантаження. Це дозволяє будувати прогнози високої точності навіть у випадках глибокої нелінійності та шумності вхідних даних.

Таким чином, цифровий двійник вебсервера у системі прогнозування навантаження робить можливим перехід від реактивного управління до випереджувального, забезпечуючи стійкість, ефективність і надійність інформаційних систем. Його застосування у режимі, наближеному до реального часу, відкриває нові горизонти для оптимізації продуктивності, автоматизації прийняття рішень та побудови самокерованих вебплатформ майбутнього.

Вебсервери є ключовими компонентами сучасних комп'ютерних мереж, які обслуговують реальні запити користувачів. Їх використання дозволяє отримувати коректні та репрезентативні дані про навантаження, такі як кількість запитів, затримка, RPS, RTT та використання ресурсів серверу (CPU і RAM). Це створює необхідну основу для точного моделювання та аналізу роботи системи під різними умовами експлуатації.

Крім того, вебсервери надають можливість моделювати різноманітні сценарії трафіку, включаючи як статичний, так і динамічний контент. Така універсальність дозволяє імітувати реальні умови навантаження та оцінювати поведінку системи при пікових або аномальних потоках запитів, що важливо для перевірки ефективності алгоритмів прогнозування.

Вебсервери також підтримують широкий спектр інструментів моніторингу та логування, що дозволяє збирати детальну статистику роботи системи: час відповіді, кількість запитів на секунду, навантаження на процесор і пам'ять. Отримані дані є необхідними для застосування вейвлет-нейронних методів прогнозування, оцінки точності моделей та побудови алгоритмів адаптивного управління навантаженням.

Нарешті, використання open-source серверів, таких як Apache або Nginx, забезпечує високу гнучкість у налаштуванні, інтеграції додаткових модулів та застосуванні інструментів тестування і симуляції трафіку. Це дозволяє повторювано проводити експерименти та забезпечує наукову достовірність результатів.

Розроблена концептуальна модель забезпечує гнучкий і ефективний механізм прогнозування та управління навантаженням на вебсервер, сприяючи ефективному використанню обчислювальних ресурсів та підвищенню загальної продуктивності

системи.

Аналіз концептуальної моделі показує, що ключовим фактором, який визначає ефективність прогнозування навантаження на вебсервер, є якісний аналіз вебтрафіку. Як зазначено у низці авторитетних джерел [35, 56, 106], саме інтенсивність та характер вебтрафіку є головними чинниками, що впливають на ступінь навантаження вебсервера.

2.2. Вебтрафік як основний компонент аналізу навантаження на вебсервер

Вебтрафік – це кількість інформації, що передається через вебсервер за певний проміжок часу. Його вимірюють у пакетах, бітах, байтах або їхніх кратних одиницях (КБ, МБ тощо). Вебтрафік поділяється на:

- вихідний – передача даних із вебсервера до зовнішньої мережі;
- вхідний – отримання даних із зовнішніх джерел;
- внутрішній – обмін даними в межах локальної мережі;
- зовнішній – передача інформації через глобальну мережу інтернет.

Врахування вебтрафіку має свої особливості, що відрізняють його від інших видів навантаження в комп’ютерних мережах. По-перше, трафік є надзвичайно динамічним та нерівномірним: кількість запитів і їх інтенсивність змінюються в часі, проявляючи піки у години активності користувачів та аномальні сплески, пов’язані із зовнішніми подіями або атаками. Це вимагає застосування методів, здатних фіксувати короткострокові коливання та тренди, зокрема декомпозиційних та вейвлет-нейронних моделей.

По-друге, вебтрафік характеризується різноманітністю типів запитів і протоколів: статичні сторінки, динамічні запити, API-взаємодія, WebSocket-з’єднання тощо. Кожен тип запиту має власну тривалість обробки та вимоги до ресурсів сервера, тому при прогнозуванні необхідно враховувати структуру запитів і їх вплив на використання CPU, RAM та пропускну здатність мережі.

По-третє, вебтрафік містить сезонні та циклічні компоненти, які залежать від часу доби, дня тижня чи конкретних подій. Це накладає специфіку на збір та аналіз

даних: для коректного прогнозування потрібно враховувати часові патерни та кореляції між потоками запитів.

Нарешті, важливою особливістю є можливість збору детальної статистики за допомогою логів вебсерверів і систем моніторингу. Логи дозволяють фіксувати час обробки кожного запиту, його тип, IP-клієнта та стан відповіді сервера, що створює багатовимірний часовий ряд, необхідний для побудови точних моделей прогнозування навантаження.

У рамках дисертаційного дослідження при аналізі та прогнозуванні рівня вебнавантаження враховуються основні протоколи, що безпосередньо впливають на формування вебтрафіку, обробку запитів і передачу даних між клієнтом і сервером. До таких протоколів належать:

1. HTTP (HyperText Transfer Protocol) – базовий протокол прикладного рівня, що забезпечує обмін гіпертекстовими документами між клієнтом і вебсервером. Він визначає формат запитів (GET, POST, PUT, DELETE) та відповідей, структуру заголовків і механізми кешування.
2. HTTPS (HTTP Secure) – розширення протоколу HTTP із додатковим рівнем шифрування (SSL/TLS). Його врахування важливе через те, що більшість сучасних вебресурсів функціонує саме через захищене з’єднання. Це впливає на навантаження сервера через криптографічну обробку даних, що повинно бути враховано при моделюванні реального трафіку. У моделі прогнозування HTTPS використовується як основне джерело даних про кількість запитів, типи контенту та частоту звернень до серверу.
3. TCP (Transmission Control Protocol) – транспортний протокол, який гарантує надійну передачу даних, контроль черговості пакетів і відновлення втрат. Він визначає низькорівневу частину мережевої взаємодії, від якої залежать затримки (RTT) та пропускна здатність каналу. Моніторинг TCP-параметрів дозволяє оцінювати мережеву стабільність і якість зв’язку.
4. UDP (User Datagram Protocol) – альтернативний транспортний протокол, який використовується для швидкої, але ненадійної передачі даних, наприклад у потокових застосунках чи мікросервісах. У деяких сценаріях

навантаженого вебсервера UDP-запити (наприклад, DNS або телеметрія) також впливають на загальний рівень трафіку.

5. DNS (Domain Name System) – протокол перетворення доменних імен у IP-адреси. Кількість DNS-запитів може свідчити про активність користувачів, збільшення відвідуваності або зміну структури звернень до серверу, що є допоміжним параметром у прогнозуванні навантаження.
6. WebSocket – протокол двосторонньої взаємодії між клієнтом і сервером у реальному часі. Він дозволяє зменшити накладні витрати, характерні для HTTP, і часто використовується у вебдодатках з постійною передачею даних (чатах, аналітичних панелях). Урахування WebSocket-трафіку є важливим при моделюванні високочастотних запитів і подієвих потоків.

Таким чином, сучасний вебтрафік ґрунтується на взаємодії кількох протоколів різних рівнів – від транспортного до прикладного. Їхнє поєднання забезпечує надійність, швидкість і безпеку обміну даними, що є критично важливим для аналізу навантаження вебсерверів і побудови моделей прогнозування трафіку в умовах реальних мережевих процесів.

У контексті прогнозування навантаження рівень вебтрафіку розглядається як часовий сигнал, що відображає змінність користувацької активності в залежності від добових, тижневих або сезонних циклів. Його математичне моделювання за допомогою вейвлет-перетворення та нейронних мереж допомагає виявити приховані закономірності, тренди й аномалії, що стає основою для побудови адаптивних алгоритмів управління вебнавантаженням.

Таким чином, рівень вебтрафіку у дослідженні виступає не лише як статистичний показник, а як інформаційна модель стану вебсервера, яка забезпечує можливість прогнозування, оптимізації та підтримки стабільності роботи серверних систем під час змінної інтенсивності запитів користувачів.

Вебтрафік дозволяє аналізувати популярність сайтів, окремих сторінок і розділів за допомогою лог-файлів вебсервера. Кожне звернення до файлу (вебсторінки, зображення тощо) реєструється як хіт (рис. 2.6). Наприклад, одна сторінка з п’ятьма зображеннями може створити шість хітів. Дані можна також

зібрати через вставлення спеціального HTML-коду для сторонніх сервісів вебаналітики. Іншим методом збору статистики є перехоплення мережевих пакетів, що дозволяє створити вибірку для оцінки загального трафіку.

```
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/map.js?1760015388 HTTP/1.1" 200 18876 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/zvit.js?1760015388 HTTP/1.1" 200 17353 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/book_phone.js?1760015388 HTTP/1.1" 200 3260 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/Oneworks_material.js?1760015388 HTTP/1.1" 200 1348 "https://dispetcher.info-gkh.com.ua/index"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/Oneworks_services.js?1760015388 HTTP/1.1" 200 1356 "https://dispetcher.info-gkh.com.ua/index"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/Oneworks_performers.js?1760015388 HTTP/1.1" 200 12351 "https://dispetcher.info-gkh.com.ua/index"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:48 +0300] "GET /assets/js/report.js?1760015388 HTTP/1.1" 200 5473 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:51 +0300] "POST /index/chatlastcomments/ HTTP/1.1" 200 6043 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:52 +0300] "POST /index/goto_server/ HTTP/1.1" 200 998 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:52 +0300] "POST /index/close_user/ HTTP/1.1" 200 998 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:53 +0300] "GET /service-worker.js HTTP/1.1" 200 220 "https://dispetcher.info-gkh.com.ua/service-worker.js" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /index HTTP/1.1" 200 277426 "https://dispetcher.info-gkh.com.ua/ads/login" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/css/fast.css?1760015396 HTTP/1.1" 200 20247 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/css/paginator.css?1760015396 HTTP/1.1" 200 1118 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/js/fastjs.js?1760015396 HTTP/1.1" 200 64391 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/js/main.js?1760015396 HTTP/1.1" 200 3089 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/js/script.js?1760015396 HTTP/1.1" 200 38524 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/js/sockets.js?1760015396 HTTP/1.1" 200 14459 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
178.136.44.213 - - [09/Oct/2025:16:09:56 +0300] "GET /assets/js/Categories.js?1760015396 HTTP/1.1" 200 3737 "https://dispetcher.info-gkh.com.ua/index" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36"
```

Рис. 2.6. Приклад записів у логах вебсервера

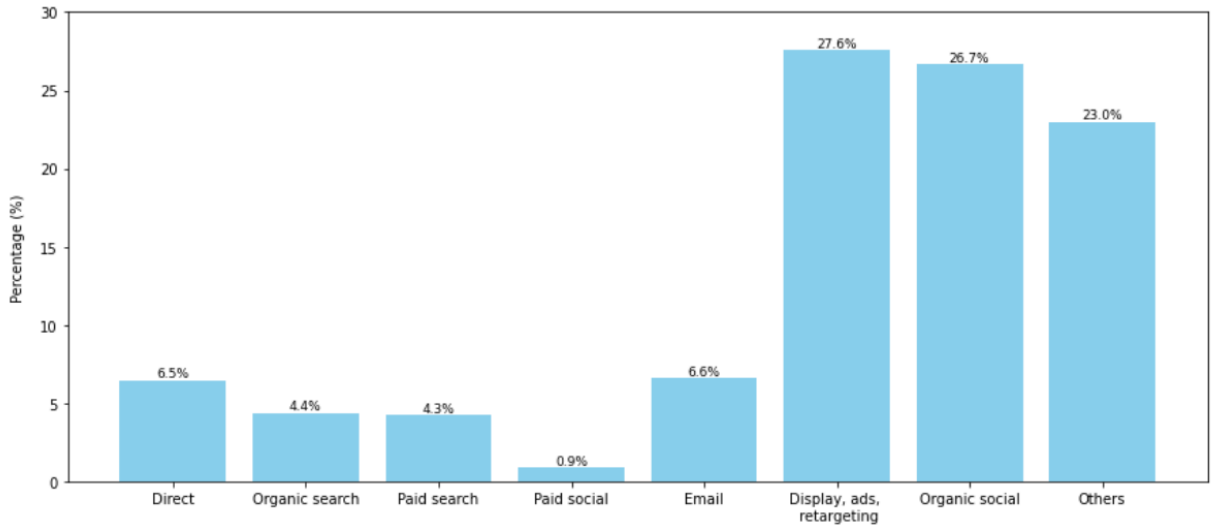

Рис. 2.7. Структура джерел вебтрафіку за відсотковим співвідношенням

Згідно з даними дослідження компанії Oberlo [87], опублікованого в листопаді 2024 року 27,6% вебтрафіку формують прямі переходи з браузера, 26,7% – через пошукові системи. Платний пошук забезпечує ще 23%, тоді як трафік із соціальних

мереж (платний та органічний) разом дає лише 7,4%. Частка трафіку з електронної пошти та банерної реклами з ретаргетингом становить близько 8,7%, інші джерела – 6,6% (рис. 2.7).

Варто зазначити, що відсоткові показники можуть змінюватися залежно від джерела та методології дослідження. Наприклад, інше дослідження від Marketing Insider Group [9] вказує, що органічний пошук становить понад 60% всього вебтрафіку.

У зв’язку з постійним зростанням кількості користувачів і збільшенням обсягів даних, важливість правильно спланованої пропускної здатності зростає. Згідно з аналітичним звітом VIAVI State of the Network 2023 [68], витрати на збільшення мережевої швидкості стрімко зростають: понад 80% глобального IT-бюджету було спрямовано на досягнення швидкостей до 400 Гбіт/с. У зв’язку з цим важливо мати чітке уявлення про параметри, що визначають ефективну роботу вебсервера.

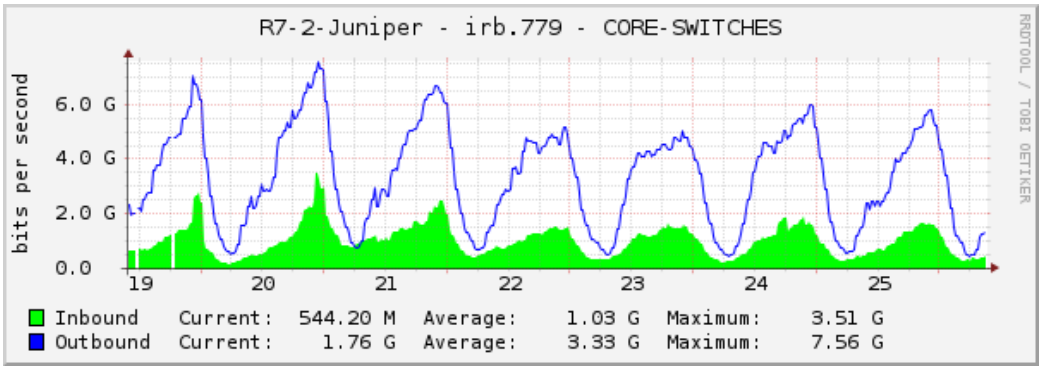

Рис. 2.8. Приклад тижневого вхідного-вихідного мережевого трафіку вебсерверу

Планування пропускної здатності є ключовим для забезпечення стабільної роботи вебсервера. Воно включає постійне відстеження обсягів трафіку, типів запитів та рівня їх обробки, що дозволяє своєчасно виявляти потенційні проблеми та реагувати на зміни навантаження. Зокрема, зниження продуктивності мережі призводить не лише до погіршення якості обслуговування, але й до зниження ефективності роботи персоналу, що в результаті негативно впливає на задоволеність кінцевих користувачів.

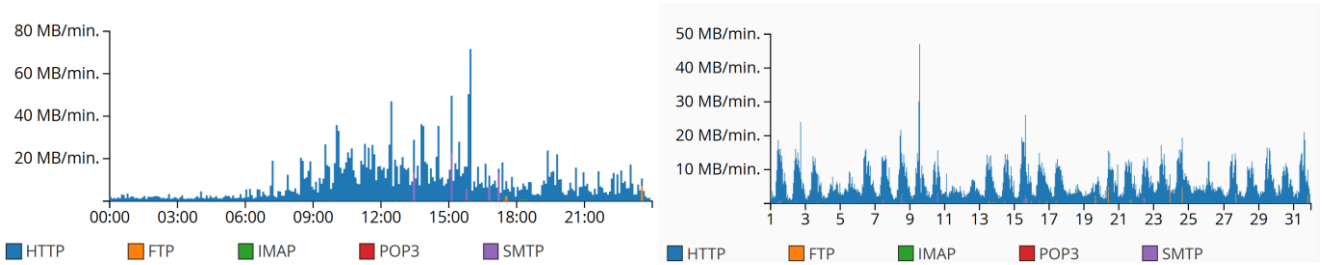

Рис. 2.9. Приклад добового і місячного мережевого трафіку вебсерверу

Аналіз отриманих мережевих графіків (рис. 2.8, рис. 2.9, додаток Е) досліджуваного вебсервера свідчить, що майже весь рівень навантаження на досліджуваному домені формується протоколом HTTP, який є основним транспортним механізмом для вебтрафіку. Переважна частина запитів до сервера надходить у вигляді стандартних HTTP-операцій (GET, POST, HEAD), що генерують як вхідний, так і вихідний трафік, показаний на графіку. Інші протоколи (SMTP, POP3, IMAP, FTP тощо) мають мінімальну частку або практично не впливають на загальний обсяг передачі даних. Таким чином, навантаження на інтерфейси мережевого обладнання корелює із веб-активністю користувачів, а пікові значення на графіках відповідають періодам інтенсивного HTTP-обміну – завантаженню сторінок, статичних ресурсів, API-запитів та потокового контенту. Це дозволяє зробити висновок, що саме вебтрафік є визначальним чинником загального мережевого навантаження для стандартного домену, що працює в КМЗП.

Отже, аналіз вебтрафіку є ключовим інструментом для розуміння взаємодії користувачів із вебресурсами та відіграє центральну роль у процесах прогнозування, моніторингу й оптимізації функціонування вебсервера. Саме завдяки цьому аналізу можна виявити закономірності навантаження, визначити критичні точки системи та своєчасно реагувати на зміни у поведінці користувачів.

До ключових метрик аналізу вебтрафіку, що можна виділити, належать (рис. 2.10):

– кількість запитів (Requests per Second, RPS) – кількісна характеристика HTTP-запитів, що надходять на сервер щосекунди. Високий RPS свідчить про підвищену активність користувачів або інтенсивний трафік, тоді як низький – про

спад навантаження;

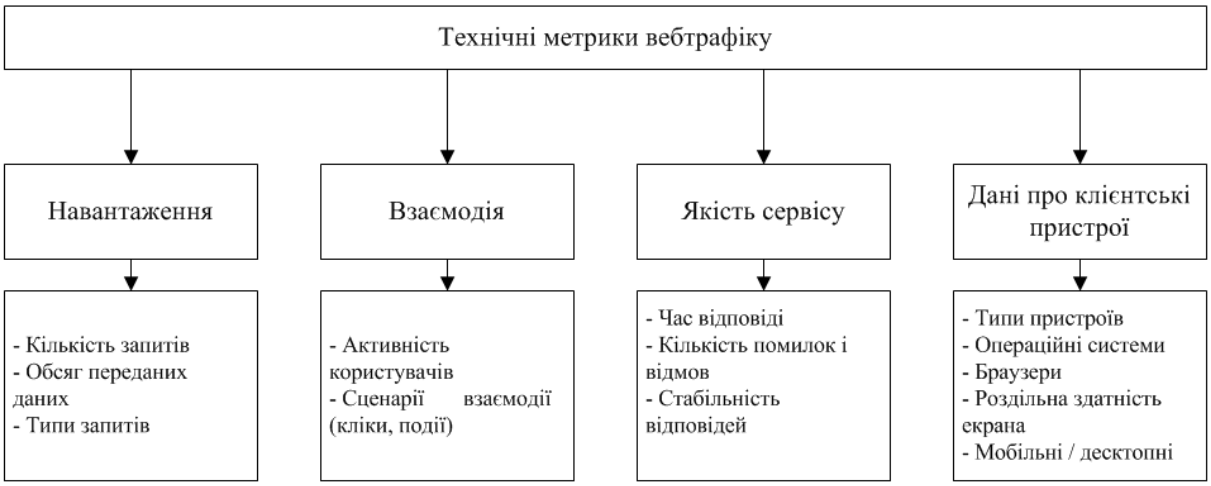

Рис. 2.10. Технічні метрики аналізу вебтрафіку

– обсяг переданих даних (Traffic Volume) – загальна кількість інформації, що передається через сервер, вимірюється в гігабайтах або терабайтах. Значні обсяги можуть впливати на продуктивність системи та швидкість доставки контенту;

– типи запитів (Request Types) – співвідношення між запитами типу GET, POST та іншими HTTP-методами. Аналіз цього параметра дозволяє краще зрозуміти характер користувацьких дій і рівень споживання ресурсів;

– час відповіді (Response Time) – період, необхідний серверу для обробки запиту та надсилання відповіді. Менші значення цього показника сприяють підвищенню швидкодії вебресурсу та покращують загальний користувацький досвід;

– активність користувачів (User Sessions) – кількість одночасно активних користувацьких сесій. Динаміка цього показника дозволяє ідентифікувати пікові періоди навантаження та оцінити популярність ресурсу;

– сценарії взаємодії (User Journeys) – типові маршрути, якими користувачі переміщуються в межах вебсайту або застосунку. Їх аналіз дає змогу виявити найнавантаженіші ділянки системи й оптимізувати використання ресурсів;

– кількість помилок і відмов (Errors and Failures) – показник стабільності

роботи сервера, який відображає кількість невдалих запитів. Виявлення таких випадків дозволяє оперативно усувати недоліки в системі;

– відомості про клієнтські пристрої (User Agents and Devices) – дані про браузери, операційні системи та типи пристроїв, які використовують відвідувачі. Ця інформація сприяє адаптації вебресурсу до різних платформ і підвищенню його сумісності.

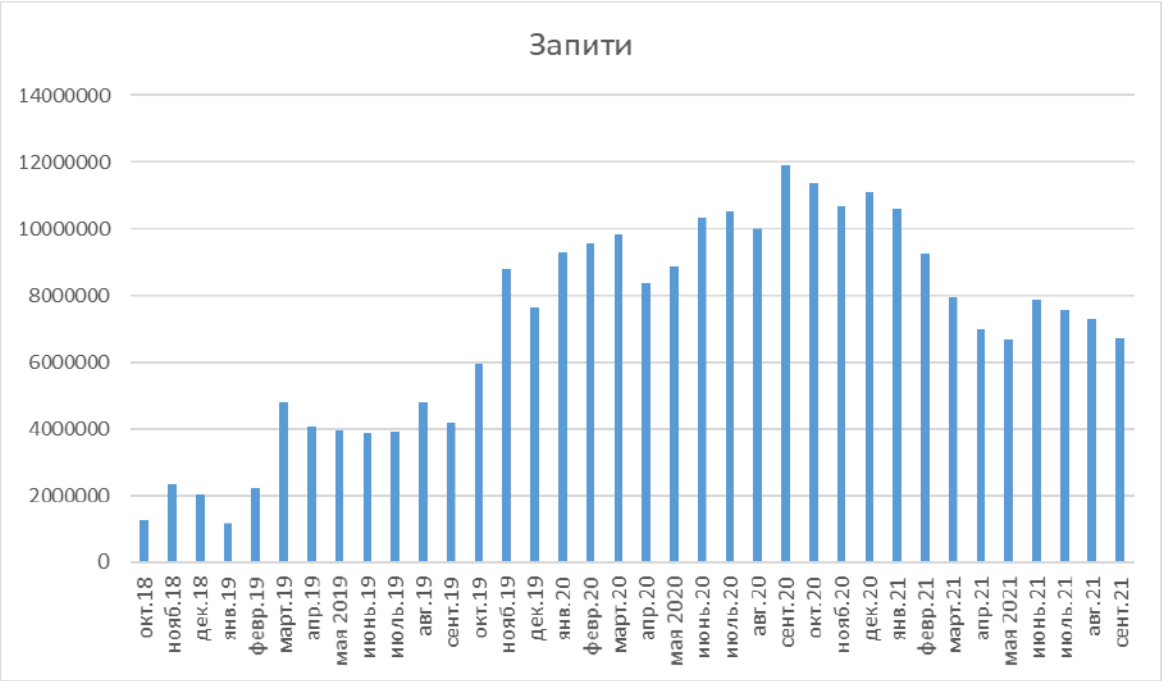

Рис. 2.11. Статистика по місяцях кількості запитів користувачів до аналізованого вебсервера

Таким чином, постійний моніторинг параметрів вебтрафіку є критично важливою складовою сучасної стратегії управління навантаженням на вебсервер (рис. 2.11). Він передбачає постійний збір даних про роботу вебресурсу та стан мережевих з'єднань, що забезпечує своєчасне виявлення змін у навантаженні, потенційних загроз чи відхилень від норми.

Зібрана під час моніторингу інформація підлягає аналізу, який включає глибоку обробку та інтерпретацію даних, виявлення закономірностей, прогнозування тенденцій та формування рекомендацій щодо оптимізації роботи вебсервера.

Основні особливості такого аналізу включають:

– використання спеціалізованих інструментів – застосування професійного програмного забезпечення, зокрема Google Analytics, New Relic або рішень з відкритим кодом, дає змогу ефективно збирати, обробляти й аналізувати дані вебтрафіку;

– реалізація моніторингу – оперативне відстеження змін у трафіку дозволяє своєчасно реагувати на аномалії, пікові навантаження або потенційні інциденти;

– фокус на ключових метриках – до основних показників слід віднести кількість HTTP-запитів, обсяг переданих даних, час відповіді сервера, а також частоту виникнення помилок і збоїв;

– оцінка типів запитів – аналіз розподілу запитів за HTTP-методами (GET, POST тощо) допомагає визначити характер користувацьких взаємодій і навантаження на окремі елементи системи;

– спостереження за користувацькими сесіями – вивчення активних сесій дозволяє відстежувати поведінку користувачів, виявляти найбільш популярні маршрути навігації та оптимізувати інтерфейс і функціонал;

– ідентифікація джерел трафіку – аналіз каналів надходження користувачів (пошукові системи, прямі заходи, зовнішні посилання) сприяє покращенню маркетингових стратегій і формуванню цільових кампаній;

– аналіз пристроїв та браузерів – врахування типів пристроїв, операційних систем і браузерів забезпечує адаптацію вебресурсу під різні платформи й поліпшує доступність для широкого кола користувачів;

– моніторинг часу відповіді – регулярне вимірювання затримок у відповіді сервера дозволяє своєчасно виявляти "вузькі місця" у продуктивності та вживати заходів для прискорення завантаження сторінок;

– виявлення аномалій – налаштування автоматичних сповіщень та систем алертів сприяє швидкому реагуванню на нетипову поведінку трафіку, помилки або перевантаження;

– захист від DDoS-атак – моніторингові засоби можна використовувати для ідентифікації підозрілої активності, пов'язаної з розподіленими атаками на

відмову в обслуговуванні, і своєчасного застосування контрзаходів;

– дотримання вимог щодо конфіденційності – системи моніторингу повинні відповідати чинному законодавству та стандартам безпеки щодо обробки персональних даних користувачів.

Таким чином, ефективна система моніторингу вебтрафіку повинна забезпечувати повноцінний контроль, підтримувати стабільність роботи вебресурсу та сприяти своєчасному прийняттю рішень в умовах динамічного цифрового середовища.

2.3. Критерії оцінки ефективності типів базисного вейвлету

Розглянута вище концептуальна модель, сформована на основі принципів багаторівневого аналізу та адаптивної обробки даних, закладає фундамент для побудови інструментів, здатних працювати в умовах високої динаміки вебтрафіку. Перехід від теоретичного обґрунтування до практичної реалізації передбачає вибір таких методів обробки сигналів, які забезпечують глибоку локалізацію ознак у часовій та частотній площинах. У цьому контексті вейвлет-перетворення виступають ключовим елементом, оскільки дають змогу не лише фільтрувати та згладжувати дані, але й виявляти короткочасні, складноструктуровані зміни у навантаженні вебсервера. Така властивість робить їх особливо придатними для створення гібридних інтелектуальних систем прогнозування, здатних адаптуватися до непередбачуваних змін у роботі інфраструктури.

Поєднання високої точності прогнозу з обчислювальною ефективністю робить вейвлет-аналіз придатним для обробки великих масивів даних. Таким чином, вейвлет-перетворення виступає потужним інструментом для аналізу та прогнозування динаміки вебтрафіку, забезпечуючи гнучкий та ефективний підхід до управління навантаженням на вебсервер.

Вейвлет-перетворення є потужним інструментом аналізу часових рядів та сигналів, дозволяючи ефективно виявляти локальні особливості та багатошкальні структури. Вибір базисного вейвлету суттєво впливає на точність аналізу та обчислювальну ефективність алгоритмів. Вейвлет-перетворення застосовується в

багатьох галузях науки та техніки, зокрема у цифровій обробці сигналів, стисненні зображень, аналізі біомедичних даних та прогнозуванні часових рядів. Одним із ключових етапів при використанні вейвлет-перетворення є вибір відповідного базисного вейвлету.

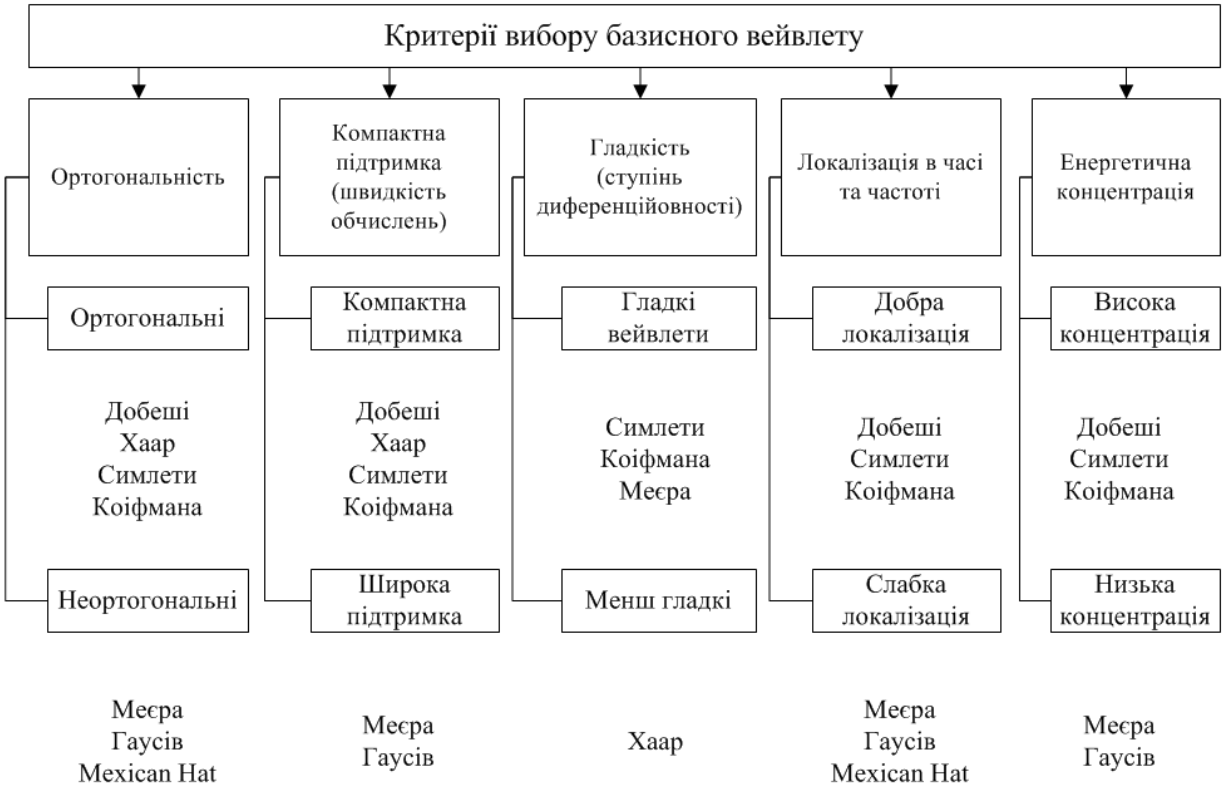

Рис. 2.12. Схема класифікації базисних вейвлетів за критеріями ефективності

Від ефективності вибору базисного вейвлету залежить якість представлення сигналу, точність обробки даних та швидкість обчислень. Тому необхідно визначити чіткі критерії для оцінки ефективності різних типів вейвлетів, що дозволить зробити ефективний вибір для конкретного завдання.

Основні критерії оцінки базисних вейвлетів включають в себе (рис. 2.12):

– ортогональність – дозволяє відтворювати сигнал без втрати інформації, що особливо важливо для задач стискання даних та шумопригнічення;

– компактна підтримка – чим більше компактність, тим швидше відбувається обчислення;

– згладжувальні властивості – вейвлети з високим ступенем гладкості

краще підходять для аналізу складних сигналів;

– локалізацію в часі та частоті – висока локалізація добре відображає локальні особливості сигналу без значних розмиттів;

– енергетична концентрація – для зменшення втрат при стисненні важливо, щоб більшість енергії сигналу містилася в малому числі коефіцієнтів після вейвлет-перетворення.

Таблиця 2.2

Порівняння базисних вейвлетів

Вейвлет	Ортогональність	Компактна підтримка	Гладкість	Локалізація	Енергетична концентрація
Хаар	+	+	-	+	-
Добеші	+	+	+	+	+
Симлети	+	+	+	+	+
Коіфмана	+	+	+	+	+
Меєра	-	-	+	-	-
Гаусів	-	-	+	-	-
Mexican Hat	-	-	+	-	-

Вейвлети з високою локалізацією в часі та частоті (Добеші, Симлети, Коіфмана) точно відображають локальні зміни сигналу. Нарешті, ефективна енергетична концентрація, характерна для Добеші та Симлетів, дозволяє зберегти основну частину інформації в обмеженій кількості коефіцієнтів, що підвищує ефективність стиснення (табл. 2.2).

Ортогональні вейвлети, такі як Добеші, Хаар і Симлети, забезпечують точне відтворення сигналу без втрат, що критично для стискання даних і шумопригнічення.

Компактна підтримка, властива вейвлетам Хаара та Добеші, сприяє швидшому обчисленню. Гладкі вейвлети, наприклад, Симлети та Коіфмана, краще

підходять для обробки складних сигналів.

Отже, залежно від поставленої задачі вибір базисного вейвлету може суттєво впливати на результат обробки даних: для стискання сигналів та шумопригнічення ефективним є Добеші або Симлети, для швидкого аналізу найкраще підходить Хаар через простоту розрахунків, для обробки складних даних найбільш підходять Коіфмана та Симлети, які мають хорошу гладкість і локалізацію, для аналізу плавних функцій корисними можуть бути Меєра або Гаусів вейвлет, незважаючи на їхню неортогональність.

Таким чином, правильний вибір базисного вейвлету є критичним для досягнення більш точного вейвлет-аналізу. Подальші дослідження можуть включати адаптивний вибір вейвлетів у залежності від характеристик вхідного сигналу.

Коротко зупинимось на особливостях застосування дискретного вейвлет-перетворення, звернемо увагу на три основні алгоритмів ДВП (табл. 2.3), які застосовується в часових рядах для виявлення трендів (низькочастотні компоненти) та аномалій (високочастотні).

Алгоритм Малла (Mallat, FWT) – це ефективна реалізація багаторівневого дискретного вейвлет-перетворення яка ґрунтується на фільтрації сигналу за допомогою низько- та високочастотного фільтрів; передбачає видалення слабкоенергетичних коефіцієнтів (downsampling) на кожному рівні, після фільтрації сигнал зменшується вдвічі; будує вейвлет-розклад сигналу за шкалами, що зростають із кожним рівнем (де масштаб j – це рівень декомпозиції).

На кожному рівні вхідний сигнал $x[n]$ проходить через: низькочастотний фільтр $h[n]$ (для отримання апроксимації A_j) і високочастотний фільтр $g[n]$ (для отримання деталей A_j).

Має перевагою високу обчислювальну ефективність ($O(N)$ для сигналу довжиною N), чітке розділення на низько- та високочастотні компоненти.

Недоліками можуть бути відсутність інваріантності до зсуву – малі зміни у вхідних даних можуть суттєво вплинути на коефіцієнти, втрата інформації через субдискретизацію, що ускладнює аналіз нестаціонарних сигналів.

Таблиця 2.3

Порівняння основних алгоритмів ДВП

Критерій	Mallat (FWT)	à trous (SWT)	Lifting Scheme
Downsampling (зменшення розмірності)	Так	Ні	Так (але можливо уникнути у деяких варіаціях)
Редундантність (надлишковість)	Ні	Так	Частково
Інваріантність до зсуву	Ні	Так	Частково
Часова складність	O(N)	O(NlogN)	O(N)
Пам’яттєва складність	Низька	Висока	Дуже низька
Використання фільтрів	Конволюційна фільтрація + декітування	Конволюція з фільтрами з вставками нулів	Факторизація фільтрів, реалізована через прості кроки

Алгоритм à trous (стаціонарне вейвлет-перетворення, Stationary Wavelet Transform, SWT) (фр. *"з дірками"*) відмовляється від субдискретизації – на кожному рівні використовується одна кількість відліків, що досягається заповненням "дірок" у фільтрах.

Особливість цього методу полягає у поступовому розширенні фільтра шляхом вставлення нульових коефіцієнтів між елементами фільтра $h(t)$ на кожному рівні згладжування, що забезпечує збільшення масштабу аналізу без зменшення кількості відліків.

Як фільтр береться материнський вейвлет, який був визначений в рамках попереднього етапу. Замість того, щоб просто застосовувати фільтрацію до сигналу, алгоритм пропускає певні елементи в сигналу на кожному етапі. На

кожному рівні розкладу пропускається певний крок фільтрації, що дозволяє зберігати розмірність сигналу та зберігати більше інформації. Після кожного етапу фільтрації отримуються вейвлет-коефіцієнти d_j, які використовуються для аналізу локальних особливостей сигналу, таких як різкі зміни або піки. Нульовий коефіцієнт d_0 представляє вхідний ряд $f(t)$. На кожному рівні зберігається інтервал або «дірка», що дозволяє зберегти більшу кількість інформації для наступних рівнів. У стандартному вейвлет-аналізі (Mallat algorithm), материнський вейвлет і масштабуюча функція формально пов'язані. Але в алгоритмі *à trous*, застосовується тільки масштабуюча функція (тобто фільтр згладжування) a_j, а вейвлет-функція d_j формується як різниця згладжених сигналів між рівнями: Після декомпозиції сигналу на різні рівні, алгоритм дозволяє відновити (реконструювати) сигнал шляхом поєднання вейвлет-коефіцієнтів.

Алгоритм *à trous* визначається через послідовність фільтрацій, де кожна фільтрація має вигляд:

$$a_j = f(t) * h_j(t) \text{ для } j = 1,2,\dots,p \ , \tag{2.15}$$

де * позначає згортку, $f(t)$ – це дані, що аналізуються, $h(t)$ – це вибраний фільтр для аналізу, $h_j(t)$ – масштабований фільтр на j-му рівні (отриманий через вставлення $2^{j-1}-1$ нулів між елементами $h(t)$).

На кожному рівні обчислюється деталізуючий коефіцієнт як різниця між згладженими сигналами:

$$d_j = a_{j-1}(t) - a_j(t), j = 1,2,\dots,p \ . \tag{2.16}$$

Ці коефіцієнти $d_j(t)$ відповідають деталям сигналу в межах відповідного масштабу j і містять локальну інформацію про зміни сигналу (піки, імпульси, флуктуації тощо).

Після завершення розкладу сигнал представляється у вигляді адитивної суми:

$$x(t) = \sum_{j=1}^{p} d_j(t) + a_p(t), \tag{2.17}$$

де $a_p(t)$ – згладжений залишковий компонент, що описує фонову тенденцію сигналу. Для запобігання спотворень на краях сигналу застосовується симетричне

віддзеркалення:

$$a(N+k) = a(N-k) \text{ для } k > 0. \quad (2.18)$$

Обчислювальна складність алгоритму становить $O(n)$, де n – довжина вхідного ряду, а обсяг необхідної пам'яті – $O(n \cdot p)$, де p – кількість рівнів декомпозиції.

Має перевагою інваріантність до зсуву – зручний для аналізу нестаціонарних сигналів (напр., часові ряди), зберігає часову роздільну здатність на всіх рівнях.

Недоліками можуть бути вища обчислювальна складність $O(N\ log\ N)$ через збільшення обсягу даних, кореляція між рівнями через надлишковість.

Алгоритм ліфтингу (Lifting Scheme) будує вейвлети через три кроки: split, predict, update. Він особливо зручний для цілочисельних або апаратно реалізованих перетворень. Алгоритм будується з операцій розщеплення (split, розділяємо сигнал на парні й непарні зразки), прогнозу (predict, використовуючи парні значення, прогнозуємо непарні), оновлення (update, оновлюємо парні значення, використовуючи помилки прогнозу) та за потреби нормалізації (масштабуємо коефіцієнти). Має перевагою адаптивність – можливість створення користувацьких вейвлетів під конкретні задачі (напр., для сигналів зі складеними паттернами), низьку обчислювальну складність $O(N)$ та можливість роботи в реальному часі.

Недоліками можуть бути складність підбору оптимальних предикторів та апдейт-операторів, обмежена інваріантність до зсуву (залежить від реалізації).

Таким чином, Mallat – ефективний для швидкого аналізу з обмеженими ресурсами, але не підходить для точного виявлення локалізованих подій, а*лгоритм à trous* – кращий вибір для нестаціонарних рядів (напр., дані навантаження), де важлива інваріантність до зсуву, Lifting – ефективний для адаптивних систем, де потрібно гнучко налаштовувати вейвлети під специфічні сигнали.

2.4. Класифікація та підходи до проєктування системи прогнозування навантаження на вебсервер

Прогнозування навантаження на вебсервери є важливим завданням для забезпечення їх стабільної роботи, оптимізації використання ресурсів і зниження

часу відгуку. Сучасні вебсервери обслуговують мільйони користувачів і піддаються швидким змінам навантаження. Непередбачувані пікові навантаження можуть призвести до зниження ефективності або навіть відмови системи. Для уникнення таких ситуацій розробляються алгоритми прогнозування навантаження, що дозволяють адаптивно управляти ресурсами серверів.

Таблиця 2.4

Порівняльний аналіз методів (5-бальна шкала)

Підхід	Точність	Обчислювальна складність	Адаптивність до змін	Застосування
ARIMA	3	1	2	Дані зі стабільним трендом
Holt-Winters	2	1	1	Короткостроковий прогноз
LSTM	4	3	4	Нелінійні часові ряди
XGBoost	3	2	3	Багатофакторний аналіз
Transformers	5	4	5	Довготривалий прогноз
Wavelet-ARIMA	4	2	2	Сезонні тренди з шумом
Wavelet-LSTM	5	3	4	Динамічне навантаження
Wavelet-Transformer	5	4	5	Довготривалі залежності

Системи прогнозування навантаження можна класифікувати за типами використовуваних підходів:

– статистичні методи (ARIMA, експоненційне згладжування);

– методи машинного навчання (LSTM, XGBoost);

– гібридні моделі (Wavelet-ARIMA, Wavelet-LSTM, Wavelet-Transformer).

Останнім часом активний розвиток отримали методи (табл. 2.4), що

поєднують часово-частотний аналіз (вейвлет-перетворення) з машинним навчанням, що дозволяє досягти вищої точності прогнозів.

Статистичні методи включають підходи, як-от ARIMA та експоненційне згладжування (Holt-Winters), які ефективні для прогнозування даних із сезонними коливаннями або трендами. Вони мають високу інтерпретованість та низькі обчислювальні витрати, однак демонструють слабку адаптацію до складних, нелінійних патернів і чутливі до різких змін у даних.

Методи машинного навчання, такі як LSTM, XGBoost і Transformers, дозволяють моделювати складні залежності у часових рядах. Вони забезпечують високу точність прогнозів та здатні адаптуватися до нелінійної динаміки, проте потребують значних обчислювальних ресурсів і великих обсягів навчальних даних.

Гібридні моделі, що поєднують вейвлет-перетворення з методами ML або DL (наприклад, Wavelet-LSTM чи Wavelet-Transformer), забезпечують попереднє фільтрування шуму та покращують прогнозування за рахунок поділу сигналу на різні частотні компоненти. Вони особливо ефективні при обробці складних і високочастотних навантажень, але потребують додаткової обробки, що збільшує обчислювальні витрати.

Вейвлет-перетворення очищає та трансформує дані перед передачею в гібридні моделі ML/DL. Розглянемо ряд таких методів детальніше (табл. 2.5).

1. *Wavelet-ARIMA* – для стабільних сезонних трендів. ARIMA (AutoRegressive Integrated Moving Average) є стандартним методом для прогнозування часового ряду, особливо при наявності трендів та сезонності. Однак стандартна ARIMA має обмеження в обробці шуму та високочастотних змін у даних. Вейвлет-перетворення DWT дозволяє виділити високочастотні компоненти, що є шумом (D1, D2, D3), залишаючи чистий тренд (A4, A5), який можна використовувати з ARIMA. Такий підхід покращує точність прогнозу, коли навантаження змінюється поступово і має періодичні цикли, наприклад, денні чи тижневі коливання. Для сезонних змін навантаження на сервері (наприклад, більше запитів вранці та менше вночі), Wavelet-ARIMA дозволить спрогнозувати навантаження. Але метод не підходить для випадків із різкими стрибками або аномальними змінами в трафіку.

Таблиця 2.5

Порівняльна таблиця гібридних методів

Метод	Підходить для	Основні переваги	Обмеження
Wavelet-ARIMA	Дані з чітким трендом, сезонність	Фільтрація шуму, точний прогноз	Не працює для різких змін
Wavelet-LSTM	Складні нелінійні залежності	Висока точність, адаптивність	Високі витрати ресурсів
Wavelet-CNN	Аналіз локальних змін, аномалії	Добре знаходить пікові зони	Менше підходить для прогнозу
Wavelet-Transformer	Довгострокове прогнозування	Глобальний аналіз залежностей	Вимагає багато обчислювальних потужностей

2. *Wavelet-LSTM* – для складних, нелінійних патернів. LSTM (Long Short-Term Memory) є потужним методом для моделювання довготривалих залежностей у часових рядах. Однак LSTM може бути чутливим до шуму, що знижує точність прогнозу. Використання вейвлет-перетворення DWT до LSTM дозволяє відфільтрувати шумові компоненти та передавати лише трендові частини для подальшого аналізу. Це дозволяє точніше передбачати складні нелінійні патерни, розділяючи високочастотні та низькочастотні компоненти навантаження. Якщо трафік вебсервера має складну, нелінійну структуру (наприклад, несподівані піки через зовнішні події), Wavelet-LSTM зможе ефективно навчити модель на цих патернах. Метод потребує значних обчислювальних ресурсів, таких як GPU або TPU, що може бути обмеженням для серверів із низьким обладнанням.

3. *Wavelet-CNN* – для виявлення аномалій у трафіку. CNN (Convolutional Neural Networks) зазвичай використовуються для роботи з зображеннями, але також ефективно застосовуються до часових рядів, якщо їх представити у вигляді вейвлет-спектрограм (CWT). Поєднання CWT з CNN дозволяє виявляти локальні закономірності в даних, що є корисним для виявлення аномалій, таких як різкі сплески в навантаженні через атаки чи флешмоби. Для виявлення DDoS-атак, де

нормальний трафік має певні частотні патерни, а під час атаки частотний спектр змінюється, CNN може помітити ці відмінності. Цей метод більше підходить для класифікації, ніж для прогнозування.

4. *Wavelet-Transformer* – для довгострокового прогнозування. Трансформери (наприклад, Temporal Fusion Transformer, T5) ефективно обробляють часові ряди, вивчаючи віддалені залежності між подіями. Застосування DWT до трансформерів дозволяє зменшити розмірність даних, видалити шумові компоненти, що покращує ефективність навчання. Цей підхід особливо підходить для довгострокового прогнозування, де потрібно враховувати зміну навантаження на сервер протягом кількох тижнів або місяців. Прогнозування піків трафіку на основі даних про навантаження за останні місяці, щоб підготувати сервери до майбутніх змін. Високі вимоги до обчислювальних ресурсів, що може бути проблемою для деяких систем.

Рис. 2.13. Класифікація систем прогнозування навантаження на вебсервер

Ефективний вибір моделі прогнозування вебнавантаження залежить від характеру трафіку: для стабільних трендів ефективним є Wavelet-ARIMA, при наявності нелінійних залежностей – Wavelet-LSTM або XGBoost, для довготривалих прогнозів – Wavelet-Transformer, а для виявлення аномалій, зокрема DDoS-атак, – Wavelet-CNN. Гібридні моделі, що поєднують вейвлет-перетворення з ML або DL-підходами, забезпечують вищу точність завдяки розділенню частотних компонентів сигналу.

Існує кілька основних категорій методів прогнозування навантаження на вебсервер (рис. 2.13), кожна з яких має власні переваги залежно від характеру даних і цілей аналізу.

1. За підходом до прогнозування (рис. 2.14). Методи прогнозування навантаження на вебсервери можна класифікувати за використовуваним підходом до обробки даних.

Рис. 2.14. Методи прогнозування навантаження за підходом до прогнозування

1.1. Статистичні методи. Цей підхід базується на математичному моделюванні часових рядів. Найпоширенішими є лінійна регресія, авторегресійні моделі (AR, MA, ARMA, ARIMA, SARIMA), а також метод експоненційного згладжування (Хольта-Вінтерса). Такі моделі добре підходять для даних із сезонністю та трендами.

1.2. Методи машинного навчання (ML). Алгоритми машинного навчання здатні моделювати складні й нелінійні залежності. До них належать дерева рішень, випадкові ліси (Random Forest), методи градієнтного бустингу (XGBoost, LightGBM, CatBoost), метод опорних векторів (SVM) і алгоритм k-найближчих сусідів (k-NN).

1.3. Глибоке навчання (DL). Методи глибинного навчання передбачають використання багатошарових нейронних мереж для роботи з великими обсягами даних. Найпоширеніші архітектури – це багатошарові перцептрони (MLP), рекурентні мережі (LSTM, GRU), згорткові нейронні мережі (CNN), а також трансформери (Temporal Fusion Transformer, GPT), що забезпечують високу точність прогнозування.

1.4. Гібридні методи. Цей підхід поєднує переваги кількох моделей. Наприклад, комбінування статистичних методів і глибинного навчання або створення ансамблів із різних моделей дозволяє підвищити точність і стійкість прогнозу.

1.5. Методи на основі вейвлет-перетворення. Вейвлет-перетворення дозволяє аналізувати сигнали з урахуванням їх частотних характеристик. Дискретне вейвлет-перетворення (DWT) дає змогу виокремлювати тренди й коливання, а неперервне (CWT) – виявляти аномалії та короткочасні зміни в запитах. Вейвлет-розкладання допомагає фільтрувати шум і згладжувати дані, що особливо корисно при довгостроковому прогнозуванні. Гібридні моделі, які комбінують вейвлети з нейромережами (наприклад, LSTM або CNN), використовуються для попередньої обробки даних і підвищення ефективності прогнозу.

2. За джерелом даних. Прогнозування навантаження на вебсервери може ґрунтуватися на різних типах даних, які розрізняються за рівнем охоплення, достовірністю та способом отримання.

2.1. Локальні дані. Це інформація, зібрана безпосередньо з одного сервера. Вона включає лог-файли, системні метрики процесора (CPU), оперативної пам’яті (RAM), дискової підсистеми та мережевих інтерфейсів. Такі дані дозволяють аналізувати роботу конкретного вузла.

2.2. Глобальні дані. До цієї категорії належать агреговані метрики, що надходять від балансувальників навантаження або кластерів вебсерверів. Вони забезпечують ширший огляд стану всієї інфраструктури, дозволяючи виявляти загальні тенденції в навантаженні.

2.3. Реальні дані. Це фактична інформація про взаємодію користувачів із системою: лог-файли HTTP-запитів, телеметрія поточної активності, а також мережеві показники трафіку. Такі дані мають найвищу практичну цінність для моделювання реальних сценаріїв навантаження.

2.4. Синтетичні дані. Цей тип даних генерується штучно за допомогою спеціалізованих інструментів, таких як Apache JMeter або Gatling тощо. Вони використовуються для створення тестового навантаження та імітаційного моделювання, що допомагає перевірити роботу системи в різних умовах.

3. За рівнем адаптивності. Підходи до прогнозування навантаження можуть відрізнятися за здатністю моделі адаптуватися до змін у даних у режимі наближеному до реального часу.

3.1. Статичні моделі. Такі моделі будуються на основі історичних даних та залишаються незмінними після початкового навчання. Вони не оновлюються при появі нових вхідних даних, тому можуть втрачати актуальність у разі змін у шаблонах навантаження.

3.2. Динамічні моделі. Ці підходи забезпечують безперервне оновлення моделі з урахуванням нових даних. Онлайн-навчання (Online Learning) дозволяє системі поступово вдосконалювати свої прогнози, адаптуючись до поточних умов і змін у поведінці користувачів.

Таблиця 2.6

Вибір підходу залежно від горизонту прогнозування

Горизонт прогнозу	Ефективні методи
Оперативне прогнозування (хвилини, секунди)	LSTM, GRU, 1D-CNN, Wavelet + ML, Kalman Filter, Online Learning Models
Короткострокове прогнозування (до 1 доби)	ARIMA, SARIMA, XGBoost, Random Forest, Wavelet + ARIMA, LightGBM, Prophet
Середньострокове прогнозування (до 1 тижня)	Трансформери (Informer, Temporal Fusion Transformer), SARIMA, BiLSTM, Wavelet + LSTM
Довгострокове прогнозування (до 1 місяця)	Prophet, LSTM Encoder–Decoder, Wavelet smoothing + ARIMA, Hybrid WNN + SVR
Трендове прогнозування (більше 1 місяця)	Holt-Winters (ETS), Prophet, Поліноміальна регресія, Wavelet low-frequency components

4. За горизонтом прогнозування. Методи прогнозування класифікуються також за тривалістю періоду, на який здійснюється передбачення (табл. 2.6). Горизонт прогнозу визначає не лише глибину майбутніх значень, але й тип моделей, що максимально ефективно працюють для певного часового інтервалу. Це пов’язано зі зміною природи часових рядів: чим більший інтервал передбачення, тим менш важливими стають локальні коливання й тим більшою є роль трендів та

сезонності.

4.1. Оперативне прогнозування (секунди або хвилини вперед). Для цього горизонту характерні швидкі флуктуації та висока динаміка. Застосовуються моделі, здатні працювати у режимі наближеному до реального часу й вловлювати дрібні зміни: рекурентні нейронні мережі (LSTM, GRU), згорткові моделі (1D-CNN), гібридні підходи на основі вейвлет-аналізу. Ці методи ефективні при моніторингу навантаження на вебсервери, мережеві трафіки, системи IoT.

4.2. Короткострокове прогнозування (до однієї доби). На цьому інтервалі можна виявити регулярні добові цикли, тому ефективними є класичні статистичні моделі – ARIMA, SARIMA, а також сучасні ML-алгоритми (XGBoost, Random Forest, LightGBM). Комбінація вейвлет-декомпозиції з ARIMA дозволяє зменшити вплив шуму, покращуючи точність прогнозу.

4.3. Середньострокове прогнозування (до одного тижня). У цьому діапазоні переважають сезонні патерни (наприклад, тижнева циклічність навантаження). Використовуються більш складні моделі, такі як трансформери (Informer, TFT), нейронні мережі глибинної архітектури (BiLSTM, Encoder-Decoder LSTM) та сезонні модифікації SARIMA. Поєднання вейвлет-аналізу з LSTM дозволяє виділяти низькочастотні компоненти, що підвищує стійкість моделі.

4.4. Довгострокове прогнозування (до одного місяця). Завдання цього рівня передбачення переважно орієнтовані на трендові та сезонні коливання. Часто застосовуються моделі типу Prophet, Holt–Winters, а також гібридні підходи (WNN + SVR або Wavelet smoothing + ARIMA). Нейронні мережі типу Encoder–Decoder теж непогано працюють із довгими часовими залежностями.

4.5. Трендове прогнозування (понад один місяць). Основна мета — не передбачити конкретні значення, а оцінити загальний напрямок розвитку системи. Застосовується для стратегічного планування ресурсів, оновлення інфраструктури та фінансового планування. Для довгих горизонтів ефективним є експоненційне згладжування (Holt–Winters), поліноміальна регресія, моделі Prophet, а також аналіз низькочастотних компонент, отриманих за допомогою вейвлет-перетворення. Такі моделі дозволяють відокремити тренд від шуму та оцінити довготривалі тенденції.

5. За обчислювальними ресурсами. Прогнозування також класифікується за способом обчислення та розміщення ресурсів, що використовуються для аналітики.

5.1. Локальне виконання. Прогнозування виконується безпосередньо на сервері, де збираються дані, що дозволяє зменшити затримки, але обмежене апаратними можливостями.

5.2. Хмарне виконання. Передбачає використання хмарних платформ, таких як AWS, Google Cloud або Azure, для здійснення прогнозів. Це забезпечує гнучкість і масштабованість.

5.3. Розподілені обчислення. Застосовується в системах із великим обсягом даних. Інструменти на кшталт Apache Spark або Kubernetes дозволяють проводити прогнозування паралельно на кількох вузлах кластера, підвищуючи продуктивність.

Таблиця 2.7

Вибір підходу залежно від обчислювальних ресурсів

Ресурси	Ефективні методи
Обмежені ресурси (невеликий рівень CPU/RAM сервера)	ARIMA, SARIMA, експоненційне згладжування
Середній рівень ресурсів	Random Forest, XGBoost, SVM
Високий рівень ресурсів (GPU, хмара)	LSTM, GRU, трансформери, Wavelet + LSTM

Навантаження на вебсервер залежить від багатьох факторів: трафіку користувачів, сезонності, раптових піків активності тощо. Тому вибір методу прогнозування залежить від вимог до точності, обчислювальних ресурсів та специфіки поведінки навантаження (табл. 2.7).

Статистичні підходи є ефективними, коли навантаження на вебсервер має чітко виражені тренди або сезонність, а також у випадках, коли система не зазнає різких змін. Їх доцільно застосовувати, якщо важлива прозорість і пояснюваність моделі, або коли обсяг даних обмежений і недостатній для тренування складніших алгоритмів. Наприклад, лінійна регресія підійде для передбачень із приблизно

лінійною динамікою навантаження. ARIMA чи SARIMA будуть доречними для даних із циклічними коливаннями, тоді як метод експоненційного згладжування Хольта-Вінтерса ефективний при поступових змінах без різких стрибків.

Методи ML варто використовувати тоді, коли дані демонструють складну та нелінійну структуру, містять аномалії або відчутні стрибки в навантаженні. Вони особливо корисні у випадках, коли модель потребує постійного оновлення. Серед ефективних інструментів: Random Forest і XGBoost, які підходять для середньострокових прогнозів на основі великих обсягів історичних даних; SVM – для невеликих, але складних наборів даних; і k-NN – якщо навантаження повторюється за схожими шаблонами в окремі періоди.

Глибокі нейронні мережі застосовуються, коли доступні великі обсяги даних, а також за наявності складних залежностей у часових рядах. Вони здатні автоматично виявляти приховані закономірності без необхідності ручного інженерного аналізу. LSTM і GRU добре моделюють послідовності подій і корисні для довготривалого аналізу тимчасових змін. CNN ефективні у виявленні локальних шаблонів у трафіку, а трансформери демонструють високу точність у складних сценаріях довгострокового прогнозування.

Підходи з використанням вейвлетів особливо корисні тоді, коли навантаження є нерівномірним, містить короткочасні імпульси або шум, що заважає точному прогнозуванню. Вейвлет-перетворення дозволяє відокремити короткотермінові коливання від довгострокових трендів. Дискретне вейвлет-перетворення (DWT) використовується для багаторівневої обробки сигналу – воно допомагає прибрати шум і готує дані до подальшого аналізу, зокрема в ARIMA чи LSTM. Неперервне вейвлет-перетворення (CWT) краще підходить для аналізу частотних компонентів і виявлення аномалій. Гібридні рішення на основі Wavelet + ML/DL дозволяють скоротити розмірність даних і підвищити точність прогнозів. У задачах фільтрації шуму DWT розкладає сигнал навантаження, після чого відкидаються високочастотні компоненти, і очищені дані передаються до моделі, після чого машинне навчання прогнозує потенційні аномалії.

Отже, для прогнозування стабільного та передбачуваного навантаження

ефективно використовувати прості моделі, такі як ARIMA чи XGBoost. У випадку складних і динамічних змін доцільно застосовувати глибокі нейронні мережі (LSTM, GRU, трансформери), тоді як вейвлет-перетворення виявляються корисними для детального аналізу короткочасних флуктуацій і зменшення шуму в даних. Найвищу ефективність демонструють гібридні підходи, що поєднують вейвлет-аналіз із машинним або глибоким навчанням, особливо в умовах нестабільного навантаження. Завдяки здатності вейвлет-перетворення аналізувати сигнали одночасно в часовій та частотній областях, воно дозволяє окремо виділяти та обробляти як довготривалі тренди (низькочастотні компоненти), так і різкі короткострокові коливання (високочастотні компоненти), що виникають, наприклад, під час флешмобів або DDoS-атак. Дискретне вейвлет-перетворення (DWT) дозволяє структурувати дані за частотними рівнями та очищувати їх від шуму, що є особливо корисним перед подальшим використанням моделей ML чи DL. У свою чергу, безперервне вейвлет-перетворення (CWT) застосовується для виявлення аномалій та аналізу швидких змін у навантаженні завдяки формуванню спектру частот, що дозволяє глибше зрозуміти структуру трафіку.

Гібридні моделі, які поєднують вейвлет-перетворення з методами машинного або глибокого навчання, дають змогу суттєво підвищити точність прогнозування навантаження на вебсервери. Завдяки попередньому розкладанню часових рядів на високочастотні та низькочастотні компоненти, такі підходи дозволяють ефективно фільтрувати шум, виявляти тренди й адаптувати модель до різних типів змін. Наприклад, компоненти з низькою частотою можуть передаватися до моделей типу ARIMA для аналізу глобальних трендів, тоді як високочастотні складові – до LSTM або CNN для обробки короткострокових флуктуацій. Таким чином, гібридні схеми Wavelet + ML/DL поєднують переваги частотного аналізу з можливостями сучасних алгоритмів прогнозування, що робить їх надзвичайно ефективними для роботи з динамічними та шумними даними.

Крім усього вище зазначеного у прогнозуванні часто застосовують вейвлетні нейронні мережі (Wavelet Neural Networks, WNN), які працюють саме з DWT. Вони використовуються для аналізу та прогнозування нелінійних динамічних систем,

зокрема часових рядів, обробки сигналів, зображень та інших складних даних.

Внаслідок підходів до реалізації такого типу нейронної мережі він має цілий ряд переваг порівняно з традиційною нейронною мережею:

- виділення значущих характеристик – вейвлети дозволяють ефективно розкладати сигнали на різних масштабах, що покращує розпізнавання закономірностей;
- стійкість до шуму – завдяки можливості усунення несуттєвих компонентів сигналу;
- адаптивність – можуть працювати з нелінійними залежностями краще, ніж стандартні нейромережі;
- менше нейронів у прихованому шарі – WNN часто потребує менше параметрів, ніж багатошарові персептрони (MLP).

WNN має схожу архітектуру з класичними MLP, але з відмінностями в активаційній функції:

- вхідний шар – отримує часовий ряд або інший сигнал [86];
- прихований шар – замість класичних функцій активації (ReLU, Sigmoid) використовуються вейвлетні функції (Морле, Добеші, Хаара тощо);
- вихідний шар – зазвичай лінійна комбінація вихідних значень прихованого шару.

Формально, вихідний сигнал WNN можна записати як:

$$y(x) = \sum_{i=1}^{N} w_i \psi\left(\frac{x - b_i}{a_i}\right), \tag{2.19}$$

де ψ – вейвлетна функція, a_i і b_i – параметри масштабу і зсуву, w_i – вагові коефіцієнти, N – кількість нейронів у прихованому шарі.

Алгоритм навчання нейронної мережі з вейвлетами (WNN) поєднує в собі переваги вейвлет-перетворення та нейронних мереж, що дозволяє ефективно обробляти та аналізувати складні сигнали. Основні етапи навчання WNN включають:

- *Передобробка даних*: Вхідні дані (сигнали) піддаються вейвлет-перетворенню для виділення важливих характеристик, таких як частотні

компоненти та часові особливості. Вибір відповідної вейвлет-функції (наприклад, Морле, Добеші) та параметрів перетворення є критичним для досягнення бажаних результатів.

- *Формування навчального набору даних*: З перетворених сигналів формуються вектори ознак, які використовуються як вхідні дані для нейронної мережі. Мітки або цільові значення (наприклад, класифікація, регресія) асоціюються з кожним вектором ознак.

- *Проєктування та ініціалізація нейронної мережі*: Вибір архітектури мережі (наприклад, кількість шарів, кількість нейронів у кожному шарі). Ініціалізація ваг мережі випадковими значеннями або за допомогою методів, які сприяють швидшому збіжності.

- *Навчання мережі*: Використання алгоритму зворотного поширення помилки (backpropagation) для коригування ваг мережі. Оптимізація функції втрат (наприклад, середньоквадратична помилка для задач регресії або крос-ентропія для задач класифікації) за допомогою методів градієнтного спуску. Розподіл даних на навчальну, валідаційну та тестову вибірки для оцінки та запобігання перенавчанню.

- *Оцінка та вдосконалення моделі*: Аналіз продуктивності мережі на валідаційних та тестових даних. Налаштування гіперпараметрів (наприклад, швидкість навчання, кількість епох) для покращення результатів. Використання методів регуляризації (наприклад, dropout, L2-регуляризація) для запобігання перенавчанню.

- *Інтерпретація та використання моделі*: Аналіз важливості окремих вейвлет-коефіцієнтів для прийняття рішень моделлю. Застосування навченого WNN для прогнозування, класифікації або інших завдань на нових даних.

Поєднання ефективності вейвлет-перетворення з гнучкістю нейронних мереж дозволяє обробляти дані з різними частотними та часовими характеристиками, що робить WNN потужним інструментом для аналізу складних сигналів.

Вимірювання точності прогнозування є однією з ключових метрик

ефективності концептуальної моделі. Вона визначає, наскільки передбачувані значення навантаження співпадають із фактичними показниками вебсервера. В рамках сучасних моделей прогнозування застосовуються кількісні метрики, що дозволяють формалізувати цей процес: MAE, MAPE, RMSE.

Автоматизація вимірювання визначає здатність концептуальної моделі самостійно виконувати цикл збору, обробки та прогнозування даних без участі оператора. Це забезпечує стабільність роботи системи, зменшує ймовірність людських помилок та дозволяє підтримувати режим наближений до реального часу. Високий рівень автоматизації забезпечує не лише швидкість і ефективність обробки, а й можливість масштабування системи на декілька вузлів, що особливо важливо у великих вебінфраструктурах.

Невизначеність вимірювання прогнозів є природним наслідком стохастичності трафіку, шуму телеметрії, пропусків даних та модельних обмежень. Вона визначає надійність прогнозів і дозволяє виявляти критичні моменти, коли рішення можуть базуватися на неповній або потенційно неточній інформації. Основні підходи до оцінки невизначеності вимірювання включають: дисперсію та стандартне відхилення прогнозів, що дозволяє оцінити розкид значень навантаження та стабільність моделі; інтервали довіри прогнозів, що показують ймовірний діапазон фактичного навантаження для кожного прогнозного значення вимірювання.

Контроль невизначеності вимірювання тісно пов’язаний із механізмами адаптивного оновлення моделей та контролю помилок, що дозволяє підтримувати високий рівень точності прогнозування навіть при значних коливаннях навантаження.

Для концептуальної моделі прогнозування доцільно використовувати комбіновану метрику ефективності вимірювання, що враховує точність, автоматизацію та невизначеність. Формально її можна подати як функцію:

$$E = f(MAE, RMSE, MAPE, Automation\ Level, Uncertainty), \qquad (2.20)$$

де E — комплексний показник ефективності, $Automation\ Level$ — рівень автоматизації процесів, $Uncertainty$ — міра невизначеності прогнозів.

Такий підхід дозволяє порівнювати різні конфігурації моделі, оптимізувати параметри концептуальної моделі та нейромереж, а також приймати управлінські рішення щодо ресурсів вебінфраструктури.

Дуже стисло, загальний підхід до прогнозування навантаження на вебсервер можна розділити на наступні етапи: *декомпозиція сигналу навантаження* за допомогою обраного вейвлета; *аналіз низькочастотних компонент* для виявлення довготривалих трендів; *обробка високочастотних компонент* для виявлення різких змін та аномалій; *реконструкція сигналу* для отримання прогнозованого значення; *комбінація з нейромережами* (вейвлет-нейронні мережі) для покращення точності прогнозу.

Висновки до розділу 2

У межах другого розділу було виконано поглиблений аналіз та формалізацію завдання прогнозування навантаження на вебсервери загального призначення з урахуванням особливостей їхньої роботи в комп'ютерних мережах. Основний акцент зроблено на розробці концептуальної моделі прогнозування, яка стала логічним продовженням результатів, отриманих у першому розділі цієї дисертаційної роботи, і конкретизувала складові та параметри, необхідні для практичної реалізації прогнозної системи.

У процесі дослідження було уточнено предметну область та визначено комплекс ключових експлуатаційних характеристик вебсервера, що мають безпосередній вплив на рівень його навантаження. До таких параметрів належать кількість запитів у черзі, середній час відповіді, швидкість обробки запитів та інші метрики, що відображають стан і продуктивність системи. Для формалізації цих характеристик запропоновано опис області працездатності вебсервера через множину контрольованих параметрів $\{\tau_i\}$ з чітко встановленими граничними значеннями $[x_{min}, x_{max}]$. Вихід будь-якого параметра за межі допустимого діапазону розглядається як ознака перевантаження та потенційного погіршення якості обслуговування.

На основі отриманих даних розроблено концептуальну модель

прогнозування навантаження, яка описує взаємозв’язок між динамікою параметрів вебтрафіку та прогнозними оцінками майбутніх станів системи. Ключовою особливістю моделі є автоматичний вибір типу базисного вейвлету залежно від характеру вхідного сигналу, що дозволяє підвищити ефективність аналізу. Обґрунтовано застосування вейвлет-перетворень для багатомасштабного аналізу даних, який дає змогу одночасно виявляти довгострокові тенденції та короткочасні пікові зміни у вебтрафіку, що є критично важливим для своєчасного реагування на зміни навантаження.

Методологія моделі передбачає використання процедур фільтрації та попередньої обробки даних перед виконанням прогнозування, що зменшує вплив шумів та випадкових коливань на точність результатів. Визначено критерії ефективності вибору типу базисного вейвлету: точність відтворення сигналу, швидкість обчислень та стійкість до завад. Показано, що некоректний вибір параметрів материнського вейвлету здатен істотно знизити якість прогнозування, що обумовлює необхідність реалізації процедур їх адаптивного налаштування.

Отримані результати мають значну цінність для подальших досліджень і практичного впровадження. Запропонована концептуальна модель формує основу для побудови гібридних систем прогнозування, зокрема рішень на базі Wavelet+LSTM, здатних зменшити час аналізу вебтрафіку та підвищити точність прогнозів завдяки адаптивному вибору базису. Гнучкість моделі забезпечує її сумісність із наявними системами моніторингу та управління вебресурсами без необхідності суттєвих змін в архітектурі.

РОЗДІЛ 3. ОСОБЛИВОСТІ МОДЕЛІ ТА МЕТОДІВ ПРОГНОЗУВАННЯ НАВАНТАЖЕННЯ НА ВЕБСЕРВЕР ЗА ДОПОМОГОЮ ВЕЙВЛЕТ ПЕРЕТВОРЕНЬ

3.1. Базисні принципи розробки моделі прогнозування навантаження на вебсервер

Вебсервер можна розглядати як масштабну розподілену інформаційну систему, яка функціонує в умовах просторової та часової локалізації. Потік вебзапитів має виражену динаміку, що залежить від факторів, таких як кешування, попереднє завантаження чи система розповсюдження документів (рис. 3.1).

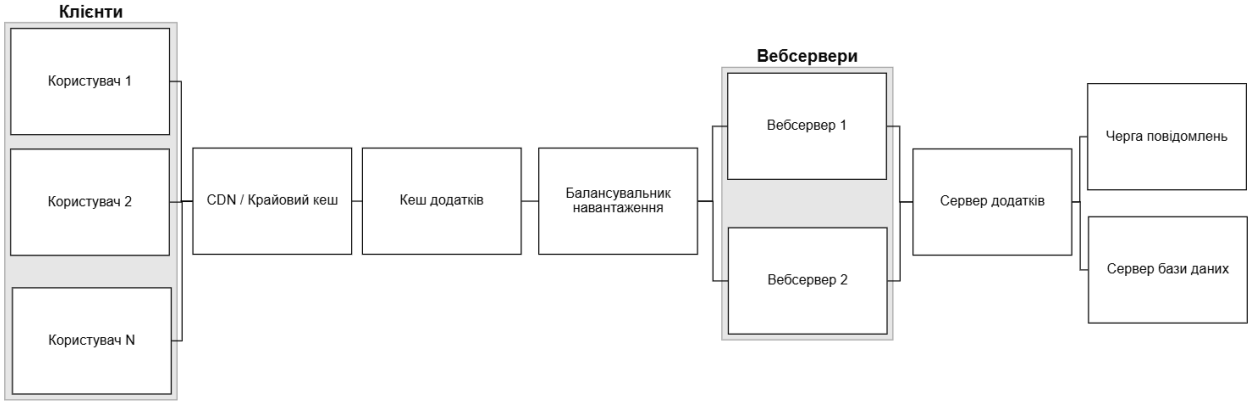

Рис. 3.1. Архітектурна схема вебсервера та потоків навантаження

Часова локалізація означає, що документи, до яких нещодавно звертались, з великою ймовірністю знову будуть затребувані найближчим часом. Це стало підґрунтям для ідеї організації серверного стеку об’єктів, впорядкованих за часом останнього використання [88]. Просторова локалізація, у свою чергу, відображає той факт, що кількість унікальних послідовностей посилань значно менша, ніж очікувалось би при випадковому доступі [58].

Такі властивості безпосередньо впливають на характер вебтрафіку, який відзначається великою варіативністю в різних часових масштабах [62]. Він має властивість самоподібності: локальні флуктуації повторюють глобальні закономірності, а масштабні кореляції призводять до ефекту «далекодії». У

результаті, сплески активності відтворюються на багатьох часових рівнях, незалежно від масштабу [94].

Самоподібність вебтрафіку пояснюється низкою чинників: механізмами кешування, поведінкою користувачів під час прийняття рішень, а також специфікою розподілу розмірів переданих документів [5]. Це відрізняє реальний трафік від класичних пуасонівських чи марковських моделей, які характеризуються короткою кореляцією та з часом згладжуються. У роботах [5, 6, 94] доведено, що моделювання з урахуванням самоподібності дає більш адекватні результати щодо затримок і втрат пакетів, ніж традиційні методи.

Таким чином, розробка моделі прогнозування навантаження на вебсервер має базуватися на урахуванні складної природи трафіку. Серед ключових принципів варто виділити: використання часових рядів і сезонних закономірностей; обробку шумів та аномалій; роботу з великими обсягами даних; кластеризацію запитів; застосування імовірнісних методів для оцінки невизначеності тощо.

Сформулюємо ряд принципів, які дозволять побудувати масштабовану, швидкодіючу уточнену модель прогнозування навантаження на вебсервер, що буде враховувати часові залежності, циклічність, стохастичні процеси, зовнішні фактори та адаптивність тощо.

Принцип стохастичності та врахування зовнішніх факторів полягає в тому, що навантаження на вебсервер є стохастичним процесом, і модель має враховувати цю невизначеність виходячи з концепції, що майбутнє значення навантаження X_{t+h} умовно на попередні значення $X_t, X_{t-1}, \ldots, X_1$ має нормальний (гауссів) розподіл N з математичним сподіванням μ та дисперсією σ^2:

$$P(X_{t+h}|X_t, X_{t-1}, \ldots, X_1) \sim N(\mu, \sigma^2). \quad (3.1)$$

Навантаження на вебсервер X_t залежить від різних факторів: удень воно зазвичай вище, ніж уночі, а по буднях інтенсивність трафіку більша, ніж у вихідні. У періоди свят активність користувачів може зростати, а в спекотну погоду частіше використовується мобільний інтернет. Значний вплив має також кількість переходів із пошукових систем – чим їх більше, тим вищий рівень навантаження. Крім того, зі збільшенням кількості одночасно активних користувачів

підвищується ймовірність пікових навантажень.

$$X_t = \beta_0 + \sum_{i=1}^{n} \beta_i Y_{i,t} + \epsilon_t, \quad (3.2)$$

де $Y_{i,t}$ – зовнішні фактори, що впливають на навантаження (наприклад, час доби, день тижня, температура повітря), β_0 – вільний член (константа), β_i – коефіцієнти моделі, що показують вплив кожного фактора, ϵ_t – випадковий шум.

Принцип часових залежностей та сезонності полягає в тому, що навантаження на вебсервер змінюється з часом, тому важливо використовувати методи часових рядів. Для часового ряду показників навантаження використовується використовується автокореляційна функція для виявлення залежностей між поточними та попередніми значеннями випадкового процесу X_t навантаження на сервер. Вона допомагає визначити, наскільки поточне значення пов'язане з його попередніми значеннями через певний лаг *k*:

$$\rho(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}, \quad (3.3)$$

де $\rho(k)$ – автокореляція на лагу k, E – математичне сподівання, $\mu = E[X_t]$ – середнє значення, $\sigma^2 = E[(X_t - \mu)^2]$ – дисперсія, X_t і X_{t+k}– значення навантаження на вебсервер у моменти часу *t* та *t+k*. Автокореляційна функція допомагає визначити домінуючі частотні компоненти в часовому ряді перед застосуванням вейвлет-перетворення.

Крім того, навантаження на вебсервер має добові, тижневі або місячні цикли. Ці особливості можна виявити використовуючи Facebook Prophet – спеціалізована модель для прогнозування часових рядів, з вираженими сезонними компонентами, святами та трендами. Prophet може працювати з відсутніми даними і забезпечує стабільні результати навіть у випадку сильних коливань у навантаженні:

$$X_t = T_t + S_t + H_t + R_t, \quad (3.4)$$

де T_t – тренд (глобальні зміни навантаження на сервер), S_t – сезонна компонента (циклічні коливання трафіку), H_t – святкові ефекти (наприклад, чорна п’ятниця), R_t – залишкова компонента (непередбачувані зміни, аномалії).

Принцип попередньої обробки та нормалізації даних полягає в тому, щоб

підготувати вхідне навантаження на вебсервер до аналізу, моделювання та прогнозування таким чином, щоб забезпечити ефективні результати та уникнути проблем, пов'язаних з неправильним інтерпретуванням або обчисленням. Попередня обробка та нормалізація даних важливі для поліпшення ефективності алгоритмів, забезпечення коректності та стабільності моделі.

Визначимо ключові етапи попередньої обробки та нормалізації:

– пропущені значення, які можуть виникати через помилки при зборі даних або через технічні проблеми заповнюються медіанним значенням для числових даних;

– викиди (значення, які значно відрізняються від інших даних) які можуть виникати теж через помилки в зборі даних або бути результатом рідкісних, але важливих подій. Викиди замінюються на середнє значення і перевіряються на наявність аномалій для більш точної ідентифікації на предмет значущості;

– Min-Max нормалізація масштабує дані в межах діапазону [0,1], що дозволяє зробити дані однорідними по всіх ознаках:

$$X_{norm} = \frac{X - min(X)}{max(X) - min(X)}. \quad (3.5)$$

Принцип вибору ефективного типу базового вейвлету полягає в тому, щоб підібрати тип базового вейвлету, який забезпечить максимальну ефективність прогнозування навантаження на вебсервер (наприклад, Хаара, Добеші, Коіфмана тощо) залежно від характеру навантаження, його дискретності та вимог до роздільної здатності. Дискретне вейвлет-перетворення (DWT) виконується за допомогою згортки із вейвлет-фільтрами з апроксимаційним коефіцієнтом $A_j[n]$ на рівні j для індексу n:

$$A_j[n] = \sum_k h[k]\, A_{j-1}[n - 2k], \quad (3.6)$$

і детальним коефіцієнтом $D_j[n]$ на рівні j:

$$D_j[n] = \sum_k g[k]\, A_{j-1}[n - 2k], \quad (3.7)$$

де $h[k]$ – це вейвлет -фільтр для апроксимаційних коефіцієнтів, $g[k]$ – це фільтр

для детальних коефіцієнтів, $A_{j-1}[n-2k]$ – це апроксимаційний коефіцієнт на попередньому рівні $j-1$, зсунутий на $2k$, щоб створити новий коефіцієнт на рівні j.

Для зменшення обчислювальної складності і пришвидшення вибору відповідного типу вейвлету використовуємо швидке вейвлет-перетворення (Fast Wavelet Transform, FWT) – це оптимізований алгоритм обчислення дискретного вейвлет-перетворення (DWT), який використовує каскадну схему фільтрації та декідування (зменшення вибірки) для обробки сигналу за O(N) замість O(N^2), як у класичному підході.

Серед усіх допустимих типів вейвлетів найбільш ефективним вважається той, для якого значення функції ефективності E_i є максимальним. Інакше кажучи, вибір ефективного типу вейвлету n_i здійснюється на основі максимізації ефективності:

$$E_i = \sum_{k=1}^{K} \tau_k T_k(n_i), \tag{3.8}$$

де $T_k(n_i)$ – значення k-го критерію для вейвлету n_i, τ_k – вага k-го критерію, K – загальна кількість критеріїв оцінювання.

Підвищення ефективності обчислень відбувається за рахунок каскадного застосування фільтрів; зменшення вибірки вдвічі (downsampling) – після згортки з фільтрами залишаємо лише парні індекси; ітеративної обробки – отримана апроксимація $A_j[n]$ стає вхідним сигналом для наступного рівня декомпозиції.

Апроксимаційні коефіцієнти (низькочастотна складова) обчислюються за формулою:

$$A_j[n] = \sum_k h[k]\, A_{j-1}[2n-k], \tag{3.9}$$

Детальні коефіцієнти (високочастотна складова) обчислюються за формулою:

$$D_j[n] = \sum_k g[k]\, A_{j-1}[2n-k], \tag{3.10}$$

де $h[k]$ – це коефіцієнти масштабуючого фільтра, $g[k]$ – це коефіцієнти вейвлет-фільтра, $A_{j-1}[n]$ – результат попереднього рівня декомпозиції.

Крім того, детальні компоненти з високою енергією можуть вказувати на ключові моменти змін у навантаженні:

$$U_j = \sum_t D_j^2(t). \tag{3.11}$$

Відновлення сигналу здійснюється через інтерполяцію та згортку:

$$A_{j-1}[n] = \sum_k h'[k]\, A_j[2n-k] + \sum_k g'[k]\, A_j[2n-k], \tag{3.12}$$

де $h'[k]$та $g'[k]$ – реконструкційні фільтри.

Принцип стійкості до шуму полягає в тому, що дані навантаження на вебсервер містять випадкові флуктуації, сплески трафіку, DDoS-атаки тощо:

$$X_t = FX_{t-1} + w_t, \tag{3.13}$$

де X_t – вектор стану системи в момент t (поточне навантаження на сервер), F – матриця переходу стану, що описує, як стан змінюється з часом, w_t – шум процесу (випадкові зміни у трафіку).

Перед застосуванням прогнозної моделі виконується вейвлет-фільтрація:

$$X(t) = A_J(t) + \sum_{j=j_{min}}^{J} D_j(t), \tag{3.14}$$

де вибирається j_{min}, що забезпечує мінімізацію шуму.

Аномалії та викиди визначаються за допомогою високочастотних коефіцієнтів вейвлет-розкладу

$$D_j(t) > \lambda\sigma_{D_j}, \tag{3.15}$$

де $D_j(t)$ – детальні коефіцієнти на масштабі j, σ_{D_j} – стандартне відхилення детальних коефіцієнтів на масштабі j, λ – поріг виявлення. Наприклад, λ=3 може вказувати, що аномалія визначається, коли детальний коефіцієнт перевищує три стандартних відхилення від середнього значення.

Для згладжування аномальних змін використовується локальний регресійний аналіз для прогнозованих даних:

$$\hat{y}_i = \sum_j w_j y_j ,, \tag{3.16}$$

де w_j – ваги, що залежать від відстані $|x_i - x_j|$.

Принцип гнучкості та адаптивності моделі полягає в тому, що розроблена модель має оновлювати свої параметри відповідно до нових даних при зміні трафіку. Цього можна досягти використовуючи метод ковзного середнього з динамічним оновленням параметрів моделі у відповідь на зміну характеристик навантаження.

Короткострокова адаптація виконується за допомогою методу ковзного вікна ширини *d* (кількість попередніх значень) для зменшення впливу застарілих значень статистичних даних $\{y_1, y_2, \dots, y_t\}$. Поточне прогнозоване навантаження розраховується як:

$$\hat{y}_t = \frac{1}{d} \sum_{i=t-d+1}^{t} y_i, \tag{3.17}$$

де $\hat{y}_t$ – прогноз у момент часу t, а y_i – попередні вимірювання навантаження.

Довгострокова адаптація забезпечується адаптивним оновленням ваг вейвлет-моделі за допомогою градієнтного спуску з регуляризацією:

$$w^{(t+1)} = w^{(t)} - \eta \left(\frac{\partial L}{\partial w} + \alpha \|w\|^2 \right), \tag{3.18}$$

де w – вагові коефіцієнти моделі, η – швидкість навчання, $\alpha\|w\|^2$ – регуляризаційний член для зменшення ризику перенавчання.

Модель не повинна бути перенавченою, а повинна зберігати здатність робити прогнози на нових, раніше невідомих даних, тобто ефективно узагальнювати закономірності без запам’ятовування конкретних прикладів з навчальної вибірки. Цього можна досягти використовуючи метод розширюваного вікна для зібраних статистичних даних $\{y_1, y_2, \dots, y_t\}$ з використанням усіх попередніх спостережень y_i до моменту *t*:

$$\hat{y}_t = \frac{1}{t} \sum_{i=1}^{t} y_i. \tag{3.19}$$

Уникнути перенавчання у нейронних мережах дозволяє Dropout – це популярна техніка регуляризації, особливо в глибоких моделях. Під час навчання нейромережі випадковим чином "вимикається" частина нейронів (разом з їх

вагами) на кожній ітерації. Тобто, у кожному проході використовується інша "підмережа" – це змушує модель не залежати від окремих нейронів, а навчатися більш стійким узагальненням. Нехай $z^{(l)}$ – вихід з шару l, тоді:

$$\tilde{z}^{(l)} = r^{(l)} \cdot z^{(l)}, \tag{3.20}$$

де $r^{(l)}$ – маска (вектор з 0 або 1), з випадковими значеннями, які вибирається за допомогою розподілу Бернуллі з параметром p (ймовірність того, що нейрон буде активним, тобто, не буде вимкнений), $\cdot$ – покомпонентне множення, $\tilde{z}^{(l)}$ – активовані виходи після застосування маски.

Принцип продуктивності та оптимізації полягає в тому, що розроблена модель має працювати ефективно (без втрати продуктивності) незалежно від обсягу вхідних даних. Тобто вона повинна масштабуватися як для невеликого вебсайту, так і для високонавантажених серверів із мільйонами запитів. Вимоги до масштабованої моделі полягають в тому щоб забезпечити можливість обробки великих обсягів даних без зниження швидкодії, гнучкість архітектури – з можливістю паралельного чи розподіленого обчислення, розподіленим зберіганням даних (якщо обсяг статистичнних даних великий) тощо. Цього можна досягти використовуючи квантильну регресію для оцінки поведінки навантаження в пікові моменти:

$$Q_\tau(y|x) = x^T \beta_\tau, \tag{3.21}$$

де $Q_\tau(y|x)$ – умовний квантиль τ-го порядку для y за умови x, β_τ – вектор коефіцієнтів регресії для квантиля τ, x^T – транспонований вектор досліджуваних змінних.

Для оцінки коефіцієнтів β_τ можна використати методи оптимізації, такі як мінімізація функції втрат за допомогою критеріїв квантильної регресії:

$$\hat{\beta}_\tau = \arg\min_\beta \sum_{i=1}^{n} \rho_\tau(y_i - {x_i}^T \beta), \tag{3.22}$$

де функція втрат $\rho_\tau(u) = u(\tau - 1(u < 0))$ для квантильного порядку τ. Це означає, що для кожного квантиля ми мінімізуємо асиметричну функцію втрат, фокусуючись на певному рівні навантаження.

Для ефективності обчислень модель повинна працювати в режимі близькому до режиму реального часу. Необхідно вибирати алгоритми, які мають лінійну O(n) або квазілінійну складність O(n log n), наприклад стохастичний градієнтний спуск, який оновлює параметри моделі після кожного окремого зразка даних, а не після всього набору, що робить метод швидшим та ефективнішим для великих обсягів даних:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t, x_t), \tag{3.23}$$

де θ_t – параметри моделі, η – швидкість навчання, $L(\theta)$ – функція втрат, x_t – поточний вхідний зразок.

Доцільно використовувати тільки найбільш значущі характеристики для зменшення ресурсоємності і, відповідно, підвищення точності прогнозування навантаження на вебсервер. Застосування аналізу основних компонентів (PCA, Principal Component Analysis) знижує розмірність матриці вхідних даних $X \in \mathbb{R}^{n \times p}$ з кількістю спостережень n і кількістю змінних p. Це дозволяє перетворити високовимірні дані у простір меншої кількості вимірів із мінімальною втратою інформації. Його суть – знайти k нових ортогональних осей (головні компоненти, власні вектори коваріаційної матриці $\Sigma = \frac{1}{n} X^T X$) $w_k \in \mathbb{R}^p$, з найбільшими власними значеннями, які максимізують дисперсію відповідних компонент (власні значення):

$$w_k = \arg \max_{w, \|w\|=1} Var(Xw). \tag{3.24}$$

В результаті ми отримуємо проекцію даних на нові осі:

$$Z = XW_k. \tag{3.25}$$

Модель має бути не тільки точною, але й ефективною у використанні ресурсів, особливо якщо йдеться про прогнозування навантаження на вебсервер. Необхідно мінімізувати функцію втрат з урахуванням обмежень на обчислювальні ресурси:

$$\min_{\theta} \mathcal{L}(X, y; \theta) + \lambda \cdot \Omega(\theta), \tag{3.26}$$

де $\mathcal{L}(X, y; \theta)$ – функція втрат (наприклад, RMSE, MAE), θ – параметри моделі, $\Omega(\theta)$ – навантаження на обчислення (кількість параметрів), λ – коефіцієнт балансу між

точністю та ресурсами.

Для компромісу між точністю та швидкодією введемо цільову функцію у вигляді співвідношення точності до часу:

$$max \frac{A(\theta)}{T_1(\theta) + T_2(\theta)}, \tag{3.27}$$

де $A(\theta)$ – точність (або 1/RMSE), T_1 і T_2 – час на навчання та прогноз.

Вибір ефективних ознак, що формулюються попереднім принципом, спрямований на якість і простоту моделі, а оптимізація продуктивності – на ефективність та швидкість. Разом вони забезпечують баланс між точністю та реалістичністю використання моделі в практичних умовах.

Принцип аналізу та комбінування прогнозів полягає в тому, що навантаження на вебсервер має різні часові масштаби з короткотривалими та довготривалими змінами. Це забезпечується використанням стаціонарного вейвлет-перетворення (Stationary Wavelet Transform, SWT) – яке усуває проблему втрати даних при розкладі сигналу X_t з виділенням особливостей трафіку та аномалій:

$$W_j(t) = \sum_k h_k W_{j-1}\left(t - 2^{j-1}k\right), \tag{3.28}$$

де W_j – вейвлет-коефіцієнти на рівні j, h_k – фільтр згладжування, 2^{j-1} – масштаб.

SWT реалізовується за допомогою алгоритму à trous (фр. *"з дірками"*), який не використовує піддискретизацію (downsampling). На відміну від дискретного вейвлет-перетворення, яке включає піддискретизацію, SWT зберігає всі точки сигналу та уникає втрати даних, забезпечує стабільність часової локалізації сигналу і відображає аномальні сплески навантаження (наприклад, DDoS-атаки) у високочастотних коефіцієнтах. Кожен рівень розкладу має той самий розмір, що і вихідний сигнал.

У багатомасштабному аналізі можна виділити глобальні тренди A_J (що відповідають великомасштабним змінам) і швидкі зміни D_j (які можуть бути зумовлені аномаліями або атакою) на різних масштабах. Навантаження розкладається у вигляді суми апроксимаційних і детальних коефіцієнтів:

$$X(t) = A_J(t) + \sum_{j=1}^{J} D_j(t), \tag{3.29}$$

де $A_J(t)$ – апроксимаційні коефіцієнти на масштабі J, $D_j(t)$ – детальні коефіцієнти на масштабах j.

Остаточне прогнозоване значення навантаження на вебсервер отримується через інтеграцію короткострокових та довгострокових компонентів, побудованих на прогнозах із різних часових масштабів.

Короткострокове прогнозування ґрунтується на детальних коефіцієнтах $D_j(t)$, що відображають швидкі зміни у трафіку (наприклад, сплески навантаження), тоді як довгострокове – на апроксимаційних коефіцієнтах $A_J(t)$, які репрезентують загальні тренди.

Підсумковий прогноз $\hat{X}(t)$ розраховується як зважена сума прогнозів з різних рівнів масштабування:

$$\hat{X}(t) = \sum_{j} w_j \hat{X}_j(t), \tag{3.30}$$

де w_j – вагові коефіцієнти, що налаштовуються за точністю кожного прогнозу.

Принцип класифікації та оцінки полягає в тому, що навантаження на вебсервер можна групувати за подібними характеристиками (тип запиту, регіон, пристрій тощо), таким чином зменшуючи складність обробки запитів та дозволяючи ефективніше розподіляти ресурси, оптимізуючи час відповіді та зменшуючи ймовірність перевантаження серверів в пікові моменти:

$$\hat{C} = \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2, \tag{3.31}$$

де k – кількість кластерів, C_i – точки в кластері, μ_i – центр кластера C_i.

Важливо оцінювати точність прогнозу моделі навантаження на вебсервер для вибору ефективного набору гіперпараметрів шляхом порівняння передбачених значень із фактичними, з урахуванням специфіки системи, критичності пікових навантажень та прийнятного рівня відхилення, використовуючи відповідні

метрики похибки, такі як RMSE – чутлива до великих відхилень (викидів), MAE – менш чутлива до викидів, MAPE – неможна застосовувати при нульовому значенні навантаження $Y_i = 0$, чутлива до малих значень навантаження у знаменнику:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}, \tag{3.32}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|, \tag{3.33}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|, \tag{3.34}$$

де Y_i – реальне навантаження, $\hat{Y}_i$ – прогнозоване.

Використання згрупованих принципів для прогнозування навантаження на вебсервер дозволяє створити комплексну та ефективну модель, яка враховує різноманітні аспекти, що впливають на серверне навантаження. Принципи часових залежностей та сезонності допомагають моделі адаптуватися до циклічних змін навантаження, а стохастичні підходи враховують невизначеність процесів. Важливість обробки даних і правильного вибору вейвлет-перетворення дозволяє витягувати ключову інформацію з сигналів, що мають різні часові масштаби, і здійснювати глибший аналіз. Адаптивність моделі, її здатність до узагальнення та оптимізації продуктивності дозволяють зберігати точність прогнозів навіть при зміні умов або в умовах великих обсягів даних. Всі ці принципи разом дозволяють створити потужну, гнучку та ефективну модель, яка може враховувати як короткострокові, так і довгострокові зміни в навантаженні, забезпечуючи тим самим стабільну та надійну роботу вебсервера.

Таким чином, у результаті проведеного дослідження було сформовано низку принципів, що слугують підґрунтям для побудови ефективної моделі прогнозування навантаження на вебсервер. Зазначені принципи становлять основу розвитку методологічної бази моделі та визначають ключові аспекти її реалізації. Нижче наведено перелік цих принципів:

– принцип стохастичності та врахування зовнішніх факторів;
– принцип часових залежностей та сезонності;
– принцип попередньої обробки та нормалізації даних;
– принцип вибору ефективного типу базового вейвлету;
– принцип стійкості до шуму;
– принцип гнучкості та адаптивності моделі;
– принцип продуктивності та оптимізації;
– принцип аналізу та комбінування прогнозів;
– принцип класифікації та оцінки.

У дослідженнях, спрямованих на підвищення точності прогнозів вебтрафіку, активно застосовуються методи на базі штучних нейронних мереж – зокрема рекурентних моделей, а також лінійні й нелінійні методи оптимізації [73, 76, 78]. У роботі [79] показано, як нейромережеві (конекціоністські) підходи можуть ефективно виявляти значущі закономірності у часових рядах на різних масштабах. Особливо значний внесок нейронні мережі зробили в задачах класифікації та прогнозування послідовностей [84].

Проте класичні алгоритми навчання на основі градієнтного спуску часто виявляються неефективними при роботі з даними, що мають довготривалу залежність [92], тобто коли поточний результат залежить від інформації, отриманої значно раніше. Цю проблему частково вирішують рекурентні нейронні мережі (RNN), зокрема архітектура LSTM, що мають здатність до збереження довгострокового контексту. Такі мережі інтегрують авторегресійні фільтри у рекурентну структуру, що дозволяє більш ефективно навчати модель навіть за наявності прихованих станів.

Окрім того, [93] показав, що багатомасштабна архітектура мережі допомагає подолати проблему зниження точності прогнозу. Для цього використовується дискретне вейвлет-перетворення, яке допомагає виявити характерні коливання вебтрафіку на різних рівнях декомпозицію даних. Замість встановлення причин часової локалізації або самоподібності вебтрафіку можна зосередити увагу на прогнозуванні трафіку [98] в масштабі годин і хвилин, після попереднього

підтвердження наявності довгострокових залежностей у часовому ряді.

3.2. Інтегрована модель процесів прогнозування навантаження на вебсервер

Процес прогнозування навантаження на вебсервер – це складна багаторівнева система, що охоплює збір, обробку, аналіз та інтерпретацію великого обсягу даних (рис. 3.2).

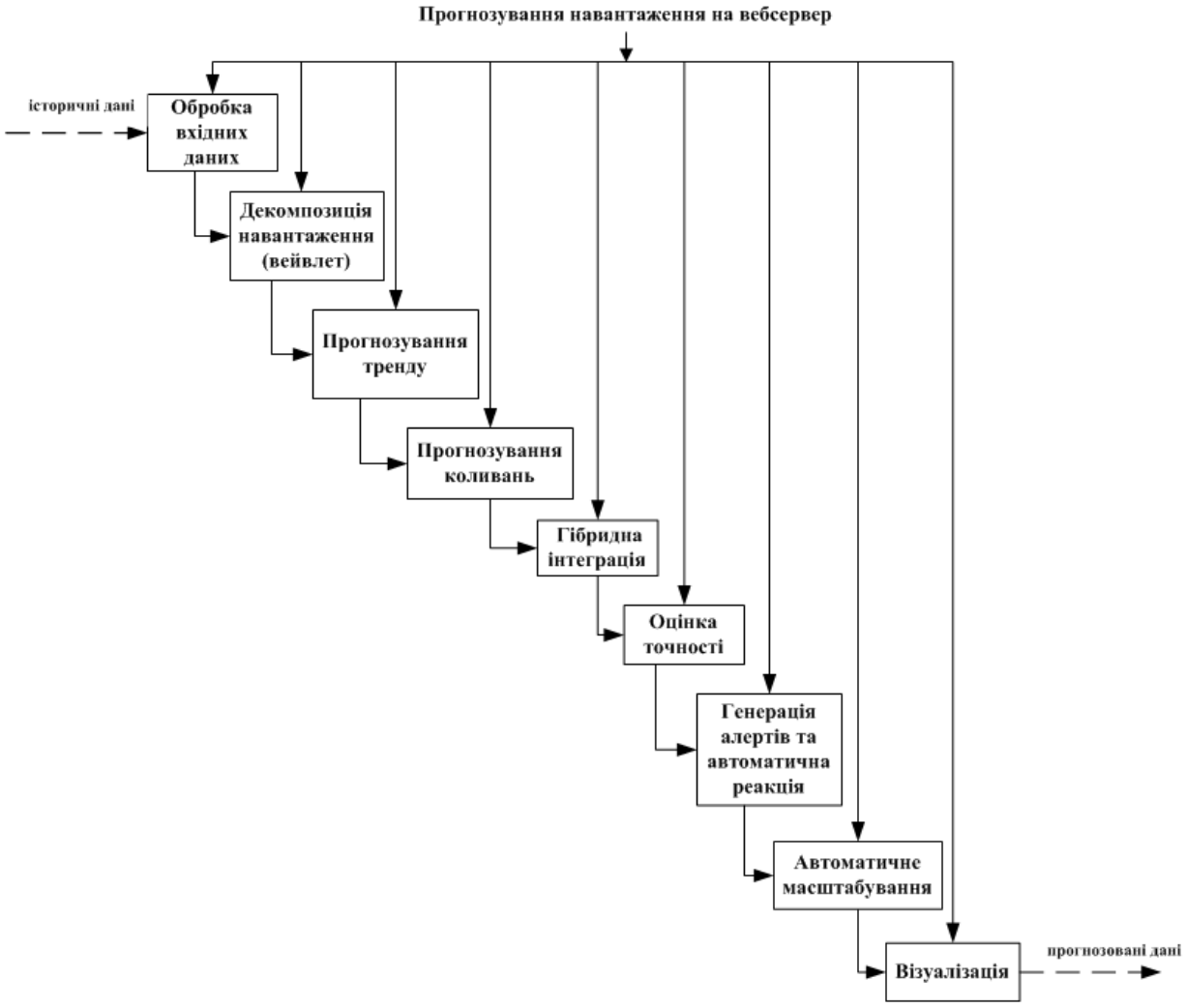

Рис. 3.2. Діаграма декомпозиції процесів прогнозування навантаження

Такий процес є критично важливим для забезпечення стабільної роботи серверної інфраструктури, зменшення часу відповіді, оптимізації розподілу ресурсів та запобігання перевантаженням, що можуть призвести до відмов у

обслуговуванні (DoS).

Інтегрована модель процесів прогнозування навантаження на вебсервер (ІМППНВ) являє собою об’єднання функціональних, аналітичних і алгоритмічних компонентів у взаємопов’язаний повний цикл прогнозування – від збору даних до прийняття рішень. Інтегрована модель реалізує взаємодію модулів моніторингу, вейвлет-аналізу, нейромережевого прогнозування, оптимізації параметрів та візуалізації результатів. Вона забезпечує цілісну інтеграцію інформаційних, аналітичних і керувальних процесів у єдиному середовищі, що дає змогу точніше формувати прогнози вебтрафіку й автоматично адаптувати конфігурацію серверних ресурсів під змінні умови роботи вебсистеми.

Прогнозування навантаження на вебсервер зазвичай розглядається як задача часових рядів, де майбутні значення метрики навантаження (кількість запитів, використання процесора, затримка відповіді тощо) передбачаються на основі історичних даних. Особливість цієї задачі полягає у високій динамічності середовища, сезонності запитів, наявності пікових навантажень та неочікуваних стрибків, зумовлених зовнішніми факторами (наприклад, маркетингові кампанії, оновлення системи тощо).

Сучасні підходи значною мірою базуються на методах машинного навчання, які дозволяють виявляти складні, нелінійні залежності між вхідними параметрами. Ці моделі потребують великої кількості навчальних даних, складного налаштування гіперпараметрів і часто погано пояснювані з точки зору внутрішньої логіки.

Сучасна тенденція – використання гібридних підходів, які поєднують переваги декількох методів. Одним із перспективних напрямів є поєднання вейвлет-перетворення для попередньої обробки сигналу з нейронними мережами для прогнозування. Вейвлет-декомпозиція дозволяє виділити коротко- та довгострокові компоненти часового ряду, зменшити шум і полегшити навчання моделей.

Іншим підходом є модульні системи, де кожен модуль відповідає за окремий етап: збір даних, фільтрацію, прогнозування, оптимізацію ресурсів. Це підвищує

адаптивність та спрощує масштабування системи.

При реалізації систем прогнозування навантаження необхідно враховувати: затримки в надходженні даних; наявність шумів і викидів; динамічну зміну поведінки користувачів; обмеження обчислювальних ресурсів; потребу в реальному або квазіреальному часі.

З урахуванням виявлених тенденцій, доступних можливостей та існуючих обмежень, було сформульовано завдання – створення інтегрованої моделі прогнозування навантаження на вебсервер. Така модель повинна поєднувати переваги різних підходів, компенсуючи їхні недоліки та водночас зберігаючи структурну простоту. Доцільним у цьому контексті є модульний підхід, за якого кожен компонент системи відповідає за окремий етап – збір, обробку або прогнозування даних. Це дозволяє легко адаптувати модель до специфічних вимог конкретного середовища.

Попередню обробку даних доцільно реалізувати за допомогою вейвлет-аналізу, що забезпечує ефективне виділення як довготривалих, так і короткострокових трендів у часовому ряді, а також дозволяє зменшити вплив шуму. Такий підхід суттєво покращує якість вхідних даних для подальшого аналізу. Це особливо актуально, оскільки у системах моніторингу може спостерігатися обмеження точності через зашумленість даних. Застосування вейвлетів на цьому етапі дозволяє ефективно виділяти значущі характеристики та покращити підготовку даних до прогнозування.

Наступним етапом є застосування нейронних мереж, зокрема таких архітектур, як LSTM або TDNN (Time Delay Neural Network), які мають здатність до навчання на часових рядах і врахування довготривалих залежностей. Це дозволяє моделі виявляти складні взаємозв’язки між параметрами навантаження, які можуть залишитися непоміченими у традиційних системах аналізу. Інтеграція нейромережевого модуля дозволить покращити точність прогнозів за рахунок виявлення складних патернів у поведінці навантаження.

У подальшому перспективним напрямом розвитку є впровадження механізмів обробки потокових даних у реальному часі. Це забезпечить безперервне

оновлення вхідної інформації про навантаження та оперативне коригування прогнозної моделі, що дозволить швидко реагувати на динамічні зміни. Додатково, модель може бути доповнена модулем динамічного програмування для ефективного розподілу обчислювальних ресурсів на основі прогнозованого навантаження. Такий підхід дозволить автоматизувати управління інфраструктурою, зменшити витрати та забезпечити високу продуктивність у пікові періоди.

Окремим напрямом є впровадження механізмів автоматичного виявлення аномалій у трафіку на основі алгоритмів, таких як Isolation Forest або AutoEncoder. Це забезпечить своєчасне виявлення відхилень, які можуть призвести до зниження продуктивності системи. Поєднання такої аналітики з вейвлет-преобробкою та нейромережевим прогнозуванням створює цілісну інтегровану систему, що не лише передбачає навантаження, а й виявляє потенційні проблеми на ранньому етапі.

У сучасних високонавантажених вебсистемах точність прогнозування трафіку визначає ефективність механізмів масштабування, балансування навантаження та забезпечення дотримання SLA. Однак будь-яка модель прогнозування – незалежно від її алгоритмічної складності – невідворотно стикається з проблемою шумів, спотворень і помилок у даних. Джерела таких помилок варіюються від нестабільності мережі та некоректного логування до аномалій у поведінці користувачів і побічних ефектів внутрішніх компонентів інфраструктури. У цьому контексті методи контролю і корекції помилок набувають значення не лише як класичний механізм забезпечення цілісності даних, але і як фундаментальна частина інтелектуальної моделі прогнозування навантаження.

У рамках інтегрованої моделі прогнозування навантаження контроль помилок формує основу достовірності усіх наступних аналітичних процесів, включно з плануванням ресурсів, автоскейлінгом, оптимізацією маршрутизації та побудовою моделі вебсервера.

Навантаження вебсерверів за своєю природою хаотичне та стохастичне. Частина нерегулярності є корисною (вказує на реальну зміну поведінки

користувачів), але значна частина є шумом, що спотворює інтерпретацію трендів.

Дискретизація, спрощення архітектури, неточні параметри та неповна імітація реальних умов породжують додаткові відхилення. Контроль помилок передбачає не лише їх виявлення, а й активну компенсацію, адаптацію моделей, самокоригування прогнозів та стабілізацію результатів.

Коректно реалізована система контролю помилок підвищує точність прогнозів на 20–40%, залежно від складності сигналу та інтенсивності шумів. Завдяки контролю помилок забезпечується передбачуване виконання сервісних зобов’язань. Контроль помилок у режимі, наближеному до реального часу, дозволяє: враховувати флеш-трафік, реагувати на DoS-/DDoS-подібні аномалії, адаптувати модель до нової поведінки користувачів. Без цього система стає надто "жорсткою" та втрачає ефективність в умовах змінності.

В інтегрованій архітектурі прогнозування вебнавантаження корекція помилок виконує роль ключового стабілізуючого компонента. На відміну від телекомуникаційних систем, де помилки зазвичай мають фізичну природу, у контексті вебсерверів вони є змішаними: структурними, семантичними та часо-залежними. Помилки вхідних даних можуть проявлятися як пропущені значення, спотворені сплески навантаження, дублікати логів, затримки часу отримання телеметрії або фрагментація events через черги повідомлень. Якщо такі деформації не виявляти й не компенсувати, вони призводять до систематичного зміщення прогнозу та спотворення результатів роботи інтегрованої моделі.

Процедури контролю помилок у моделі прогнозування виконують первинний аналіз, спрямований на виявлення нетипових, неконсистентних або статистично неможливих записів. Вейвлет-трансформація, зокрема алгоритм à trous, відіграє тут роль не тільки інструмента декомпозиції, але і механізму виявлення локальних аномалій. Оскільки вейвлети чутливі до різких змін і високочастотних шумів, вони дозволяють відокремити реальні піки навантаження від спотворень, спричинених помилками збору даних. На цьому етапі формується клас «аномальних підсигналів», які або усуваються через компенсуючі фільтри, або маркуються для подальшої обробки.

На відміну від контролю помилок, який лише ідентифікує проблеми, корекція помилок спрямована на реконструкцію часових даних таким чином, щоб вони максимально відображали реальну поведінку вебсервера. Це дозволяє відновити дані, щоб зберегти природну структуру вебнавантаження.

Етап корекції помилок у поведінковій моделі інтегрованої моделі виконує важливу адаптивну функцію. Нейромережеві модулі WNN та LSTM здатні не лише обробляти очищені часові ряди, але й навчатися компенсувати систематичні помилки, що виникають при реальному запису логів. Наприклад, LSTM-модуль може виконувати реконструкцію відсутніх ділянок, використовуючи контекст минулих станів. У свою чергу, WNN забезпечує «м’яке коригування» структури сигналу, відновлюючи його багаторівневі особливості після застосування фільтрації. Цей комбінований підхід створює ефект внутрішнього коду виправлення помилок, аналогічний до FEC-кодів у комунікаційних системах, але адаптований під динаміку вебнавантаження.

Особливої ролі корекція помилок набуває у режимах, наближених до реального часу. Навантаження вебсерверів може змінюватися зі швидкістю, яка унеможливлює глибоку пакетно-орієнтовану обробку. Тому інтегрована модель процесів прогнозування використовує ітеративні механізми «м’якого оновлення»: прогноз і фактичні значення порівнюються на кожному кроці, а відхилення, що виходять за межі очікуваної похибки, інтерпретуються як потенційні помилки телеметрії. Таким чином формується цикл адаптивного зворотного зв’язку, у якому модель сама «коригує» помилки шляхом реконфігурації ваг, параметрів вейвлет-фільтрів та інтервалів прогнозування.

У результаті корекція та контроль помилок перестають бути лише допоміжними функціями; вони стають головним механізмом підтримання достовірності даних, що надходять до інтегрованої моделі. Саме якість цих механізмів визначає здатність моделі зберігати точність передбачення в умовах реального, нестабільного трафіку вебсервера. Інтеграція таких процедур суттєво підвищує надійність системи прогнозування й забезпечує можливість реалізації випереджувального масштабування, адаптивного балансування навантаження та

відповідності вимогам SLA.

Інтегрована модель прогнозування навантаження на вебсервери може здатися занадто складною для реалізації, тому важливо дотримуватися певних принципів, які сприятимуть зниженню загальної складності системи. Насамперед, це модульність: модель доцільно розділити на окремі функціональні блоки (збір даних, попередня обробка, прогнозування, оптимізація), що дозволить спростити її розгортання, забезпечити можливість незалежної модифікації окремих модулів та зменшити витрати на підтримку. У таких умовах модульність стає ключовою архітектурною властивістю інтегрованої моделі прогнозування. Вона забезпечує її структурну впорядкованість, адаптивність, можливість незалежного оновлення компонентів, масштабованість і стійкість до помилок.

Модульна побудова моделі дозволяє розглядати прогнозування не як монолітний алгоритм, а як сукупність взаємодіючих, але автономних блоків— кожен із чітко визначеною функцією, входами, виходами та внутрішніми процедурами. В контексті інтегрованої моделі вебсервера модульність формує чітку архітектуру “компонентних рівнів”, що відтворюють різні аспекти фізичного, поведінкового та інформаційного середовища.

Модульність передбачає, що складна система розбивається на окремі модулі, які виконують окремі спеціалізовані задачі. Таке структурування дозволяє оптимізувати кожний компонент незалежно, не руйнуючи цілісності системи.

Модульність забезпечує слабке зчеплення між компонентами. Кожний модуль реалізує власну логіку, використовуючи стандартизовані протоколи обміну даними. Завдяки цьому можна змінювати внутрішні алгоритми (наприклад, замінити LSTM на Transformer) без зміни загальної структури. У складних інфраструктурах, де навантаження та поведінка користувачів змінюються, модульність дозволяє розвивати окремі частини системи поступово та без ризику втрати узгодженості всього комплексу.

Модульність дозволяє проводити порівняльне тестування моделей, легко змінювати архітектури, навчати моделі паралельно та інтегрувати зовнішні алгоритми.

При збої одного модуля система зберігає роботу в деградованому режимі, автоматично перезапускає модуль, ідентифікує джерело помилки без ефекту каскаду. До того ж, кожен модуль може бути оптимізований під свій клас задач: вейвлет-модуль – для фільтрації, нейромережевий – для прогнозування, цифровий двійник – для поведінкового аналізу, модуль контролю помилок – для стабільності моделі. Цифровий двійник можна розглядати як надбудовний модуль, що агрегує роботу інших модулів і надає системі можливість відтворювати поведінку вебсервера з підвищеною точністю.

У модульній архітектурі кожен компонент може розвиватися незалежно, виконуючи чітко визначену функцію та взаємодіючи з іншими через стандартизовані інтерфейси. Це робить систему життєздатною в умовах динамічних навантажень, високої мінливості даних у режимі, наближеному до реального часу.

Другим ключовим принципом є автоматизоване налаштування гіперпараметрів, що забезпечить скорочення часу на конфігурування моделей машинного навчання та покращить їхню точність.

Запропонована інтегрована модель поєднує в собі такі інструменти аналізу, як вейвлет-перетворення та нейронні мережі, що дозволяє підвищити якість прогнозування. Вейвлет-аналіз застосовується на етапі попередньої обробки даних для виділення короткострокових та довгострокових компонентів, усунення шуму та декомпозиції сигналу за часовими масштабами. Це покращує вхідні дані для подальшої обробки. Нейронні мережі, зокрема архітектури типу LSTM, дозволяють моделювати складні нелінійні залежності та враховувати часові контексти в даних, що особливо актуально для задач прогнозування навантаження.

Незважаючи на переваги, слід враховувати і низку обмежень. Зокрема, використання нейронних мереж супроводжується потребою у великих обсягах даних, високою чутливістю до зашумленості та необхідністю ретельного підбору гіперпараметрів, аби уникнути перенавчання. Крім того, поєднання вейвлет-перетворення та нейромереж збільшує загальну обчислювальну складність моделі.

Втім, така система має ряд суттєвих переваг: поєднання вейвлет-аналізу з

нейромережами дозволяє моделі враховувати як локальні, так і глобальні характеристики даних; вейвлети зменшують вплив шуму, підвищуючи якість навчання нейронної мережі; модель демонструє гнучкість у роботі з нестабільними часовими рядами, типовими для вебнавантаження; потенційна інтеграція потокового аналізу та динамічного програмування відкриває нові можливості для адаптивного управління ресурсами.

Рис. 3.3. Схема інтегрованої моделі прогнозування навантаження на вебсервер

Таким чином, запропонована модель здатна подолати обмеження

традиційних методів прогнозування, забезпечуючи точніші результати та ефективніше управління інфраструктурними ресурсами вебсерверів.

Після визначення в попередньому розділі основних параметрів, що описують навантаження вебсервера (кількість запитів за одиницю часу, затримки, використання CPU/RAM, мережевий трафік тощо) – переходимо до розбудови самої моделі.

На основі сформульованих раніше принципів розробимо узагальнену інтегровану модель прогнозування навантаження на вебсервер (рис. 3.3).

Модель оперує часовими рядами $X(t)$, які включають у себе кількість запитів $R(t)$, використання CPU $C(t)$, споживану пам'ять $M(t)$, мережевий трафік $B(t)$. Це формально можна записати як:

$$X(t) = \{R(t), C(t), M(t), B(t)\}. \tag{3.35}$$

Підсумковим результатом моделювання повинно стати знаходження апроксимаційної функції $\hat{X}(t)$, яка передбачає значення навантаження на вебсервер.

З формальної точки зору $X(t)$ – це випадковий процес, що описує навантаження на вебсервер у момент часу t. Завдання полягає в тому, щоб побудувати функцію прогнозування

$$\hat{X}(t+k) = f(X(t), X(t-1), \dots, X(t-n), \theta), \tag{3.36}$$

де k – горизонт прогнозування, n – кількість попередніх значень (довжина вікна аналізу), θ – параметри моделі.

Наступним кроком буде вейвлет-декомпозиція вхідного ряду. Для багатомасштабного аналізу застосовуємо дискретне вейвлет-перетворення (DWT), що розкладає $X(t)$ на апроксимаційні $A_J(t)$ та детальні коефіцієнти $D_j(t)$ (3.29).

Після цього здійснюємо вибір значущих вейвлет-компонентів. Виконуємо оцінку енергії E_j для кожного рівня (3.11) і відбираємо тільки ті рівні, де E_j перевищує порогове значення λ. Це дозволяє зменшити розмірність задачі, залишаючи лише значущі компоненти.

Наступним кроком видалимо наявний шум. Для згладжування шуму застосовується вейвлет-фільтрація за адаптивним порогом. Детальні коефіцієнти

фільтруються за наступним правилом:

$$D_j^{filtered}(t) = \begin{cases} D_j, \text{якщо } |D_j| > \theta_j \sigma_{D_j}, \\ 0, \quad \text{інакше} \end{cases} \tag{3.37}$$

де σ_{D_j} – стандартне відхилення коефіцієнтів рівня j, а θ_j – відповідний пороговий коефіцієнт.

Після цього виконуємо інтеграцію прогнозних моделей. Прогнозування здійснюється за допомогою гібридної моделі, яка комбінує метод Prophet (3.4) для апроксимаційної складової $A_J(t)$, щоб отримати базовий прогноз сезонної та трендової компоненти, та LSTM-мережу для моделювання детальних компонент $D_j(t)$ для прогнозу залишкової (residual) компоненти, яку Prophet не може змоделювати:

Загальна математична модель для нейронної мережі з довгою короткочасною пам'яттю має вигляд:

$$h_t = \sigma\big(W_h h_{t-1} + W_x D_j(t) + b\big), \tag{3.38}$$

$$\widehat{D}_j(t + \Delta t) = W_0 h_t + b_0, \tag{3.39}$$

де h_t – вектор прихованого стану на момент часу t, h_{t-1} – прихований стан на попередньому кроці $t - 1$, $D_j(t)$ – значення детальної компоненти рівня j у час t, $\widehat{D}_j(t + \Delta t)$ – прогнозоване значення цієї компоненти через крок Δt, W_h – вага, яка застосовується до попереднього стану h_{t-1}, W_x – вага для вхідного сигналу $D_j(t)$, b – зсув (bias) для входу в прихований стан, σ – сигмоїдна активаційна функція, W_0 – вага для генерації виходу з h_t, b_0 – зсув для вихідного шару.

LSTM вчиться на residual $r(t)$, тобто на різниці між реальним навантаженням $x(t)$ і базовим прогнозом $\hat{x}_{Prophet}(t)$, для моделювання короткострокових залежностей, пікоподібних змін і нелінійностей:

$$r(t) = x(t) - \hat{x}_{Prophet}(t). \tag{3.40}$$

Остаточне прогнозне значення обчислюється як сума прогнозів:

$$\hat{X}(t + \Delta t) = \hat{A}_J(t + \Delta t) + \sum_{j=1}^{J} \widehat{D}_j(t + \Delta t). \tag{3.41}$$

Формалізовано математичне забезпечення загальної гібридної моделі

записується як:

$$\hat{x}(t) = \underbrace{\hat{g}(t) + \hat{s}(t) + \hat{h}(t)}_{Prophet} + f_{LSTM}\big(r(t-1), \dots, r(t-n)\big), \quad (3.42)$$

де $\hat{g}(t)$ – прогноз трендової компоненти, $\hat{s}(t)$ – прогноз сезонної компоненти, $\hat{h}(t)$ – прогноз ефекту святкових подій, f_{LSTM} – функція, реалізована через LSTM-мережу, n – кількість лагів.

Додатково для підвищення точності прогнозу використовується адаптивне зважування компонентів:

$$\hat{X}(t + \Delta t) = w_1 \hat{A}_J(t + \Delta t) + \sum_{j=1}^{J} w_j \widehat{D}_j(t + \Delta t). \quad (3.43)$$

Вагові коефіцієнти w оновлюються методом градієнтного спуску (3.22).

Пікові значення навантаження визначаються на основі детальних компонент. Якщо для певного моменту часу:

$$D_j(t) > \gamma \sigma_{D_j}, \quad (3.44)$$

то така точка класифікується як потенційна аномалія. Параметр γ задає поріг чутливості.

Для оцінювання точності прогнозу використовуються метрики RMSE (3.32) і MAPE (3.34).

Модель регулярно оновлюється за допомогою підходу ковзного вікна. Оновлення моделі виконується кожні T_{update} секунд, що забезпечує адаптацію до змін у вебтрафіку в режимі наближеному до режима реального часу.

Часові вікна – це концепція, яка широко використовується в задачах прогнозування часових рядів, в тому числі для моделей, таких як LSTM. У цьому контексті модель "запам’ятовує" певну кількість кроків у минуле, що дозволяє ефективно виявляти тренди, сезонні коливання та інші закономірності у даних. Вибір довжини вікна суттєво впливає на якість прогнозу: надто коротке вікно може не охопити важливу інформацію, тоді як надто довге може ускладнити навчання моделі. Наприклад, якщо вікно має довжину W, для передбачення значення на кроці t використовуються дані з моментів часу від $t - W + 1$ до t.

Ефективний вибір розміру часового вікна W залежить від кількох чинників, таких як частота даних, сезонність, складність патернів та характер прогнозу. Для даних з високою частотою доцільно використовувати менші вікна через локальні кореляції, тоді як у випадку сезонних або складних закономірностей потрібні ширші вікна для врахування довгострокових залежностей. Існують різні підходи до формування вікон: фіксовані (із сталим розміром), рухомі (які постійно зміщуються вперед у часі), а також варіативні – з адаптивною довжиною або мульти-віконною структурою для виявлення залежностей на кількох масштабах одночасно. Крім того, вибір вікна залежить від горизонту прогнозування: для короткострокових завдань краще використовувати вузьке вікно, що зосереджене на нещодавніх змінах, а для довгострокових – ширше, що охоплює тренди й сезонні коливання.

Наприклад, якщо ви прогнозуєте навантаження на вебсервер і використовуєте часові дані з інтервалом в 5 хвилин (наприклад, кількість запитів за кожні 5 хвилин), то вікно довжиною 12 (тобто 1 година) може бути ефективним для короткострокових прогнозів. Таким чином, для кожного прогнозу ви будете використовувати 12 попередніх значень, щоб передбачити навантаження на сервер на наступні 5 хвилин.

Якщо є великий набір даних з різними часовими вікнами та ви хочете зберігати в часі всі важливі закономірності, доцільно використати гібридний підхід, поєднавши LSTM з нейронною мережею з часовими затримками TDNN для отримання гнучкості обробки як короткострокових, так і довгострокових залежностей.

LSTM добре підходить для довгострокових залежностей. Ця нейронна мережа має здатність зберігати інформацію про попередні стани в пам'яті протягом довгих періодів часу, що є важливим для захоплення складних і довгострокових трендів, які можуть бути присутніми в даних.

Основна особливість LSTM полягає в здатності контролювати потік інформації за допомогою спеціальних механізмів, званих воротами: вхідного, забування (forget gate) та вихідного. У поєднанні з елементом довготривалої пам’яті (cell state), це забезпечує мережі здатність до навчання складних часових

залежностей.

На кожному кроці часу LSTM приймає вхідний вектор x_t і попередній прихований стан h_{t-1}, після чого виконує кілька послідовних операцій. Шар забування визначає, яку частину збереженої інформації з попереднього стану потрібно відкинути. Це реалізується за допомогою сигмоїдальної функції активації, яка обчислює вагу забування для кожного елементу. Далі працює вхідний шар, який формує нову інформацію для запису в пам'ять, обчислюючи як кандидатів на нові значення, так і маску того, що саме потрібно оновити.

Оновлення стану пам’яті полягає в поєднанні частково збереженої старої пам’яті з новими кандидатами, зваженими через відповідні ворота. Нарешті, вихідний шар визначає, яка частина оновленого стану пам’яті буде використана для генерації виходу мережі на поточному кроці. Цей вихід знову ж таки модулюється сигмоїдальними "вихідними воротами" і нелінійною активацією $tanh$.

Математично всі ці етапи описуються системою рівнянь, де кожне з воріт має свою систему ваг (W, U) та зміщення (b), а також власну функцію активації σ. Наприклад, forget gate обчислюється як:

$$f_t = \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big). \tag{3.45}$$

Аналогічно обчислюються інші ворота та вихідний стан h_t, що в результаті дозволяє LSTM зберігати, відкидати або оновлювати інформацію на кожному часовому кроці.

TDNN більше орієнтовані на короткострокові залежності, оскільки вони працюють з фіксованими вікнами часу і використовують затримки для збереження історичних даних протягом коротших проміжків часу.

Поєднання моделей LSTM і TDNN забезпечує гнучкість в роботі з часовими рядами різної природи. LSTM виявляє довгострокові залежності, наприклад, сезонні коливання або стійкі тренди навантаження на сервер, тоді як TDNN досліджує короткострокові зміни, включаючи різкі піки навантаження. Завдяки цьому гібридна модель здатна одночасно враховувати як глобальні закономірності, так і локальні зміни. Такий підхід особливо корисний при обробці даних, зібраних з різною частотою, оскільки дозволяє зберігати важливі залежності на різних

часових рівнях деталізації.

Реалізація такої моделі починається з попередньої обробки даних, де кожне часове вікно формує окремі послідовності для відповідних моделей: короткі інтервали подаються на вхід TDNN, а довші – до LSTM. У моделі TDNN виконує першу обробку, зосереджену на короткотермінових коливаннях, після чого LSTM аналізує більш тривалі залежності. На завершальному етапі виходи обох моделей об’єднуються, наприклад, за допомогою повнозв’язаного шару або вагового механізму, що дозволяє отримати інтегрований прогноз з урахуванням різних часових характеристик даних.

Опробована архітектура гібридної моделі передбачає використання TDNN на початковому етапі для виділення короткострокових залежностей у часових вікнах шляхом застосування затримок (delay lines), що дає змогу виявити локальні патерни. Після цього оброблені дані передаються до шару LSTM, який аналізує довгострокові залежності й забезпечує прогнозування на більші періоди. На виході результати обох моделей поєднуються – через конкатенацію або зважене комбінування – та подаються на остаточний повнозв’язаний шар, який формує фінальний прогноз (рис. 3.4).

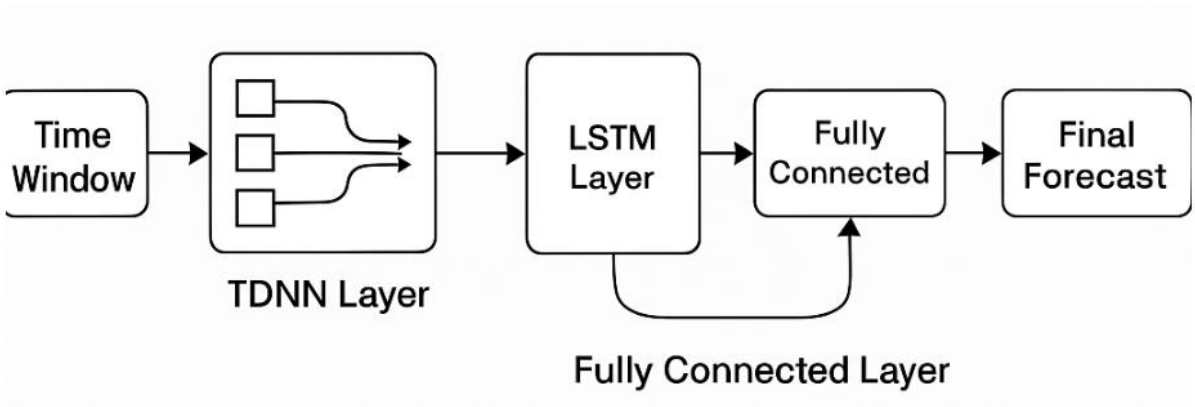

Рис. 3.4. Архітектура гібридної моделі поєднання прогнозів на різних часових горизонтах

Такий підхід забезпечує високу гнучкість у моделюванні часових залежностей на різних рівнях, сприяє зменшенню впливу шуму завдяки розподілу

обробки між коротко-, середньо- та довгостроковими компонентами, а також підвищує точність прогнозів на різних часових горизонтах.

У архітектурі TDNN з одним прихованим шаром, для кожного часу t, значення прихованого шару $h_j^{(t)}$ розраховується за допомогою активаційної функції σ, яка приймає як аргумент лінійне комбінаційне значення вхідного вектора $\mathbf{x}^{(t)} = [x(t), x(t-1), \dots, x(t-\tau)]^\top \in \mathbb{R}^{\tau+1}$, зваженого вагами $w_{ji}^{(1)}$ для кожного з попередніх моментів часу $t-i$ (де i змінюється від 0 до τ), з додаванням зсуву $b_j^{(1)}$. Тобто, кожен нейрон прихованого шару отримує інформацію з декількох попередніх моментів часу. Останнє передбачення $\hat{y}(t+\Delta t)$ формується як лінійна комбінація виходів з прихованого шару, де ваги $w_j^{(2)}$ коригують значення кожного прихованого нейрону, а $b^{(2)}$ є зсувом на виході. У цій моделі використовуються два набори ваг: перші $w_{ji}^{(1)}$ і зсуви $b_j^{(1)}$ для прихованого шару, та другі $w_j^{(2)}$ і $b^{(2)}$ для вихідного шару. Така архітектура дозволяє моделювати залежності між вхідними даними через час та робити прогнози для майбутніх значень:

$$h_j^{(t)} = \sigma\left(\sum_{i=0}^{\tau} w_{ji}^{(1)} x(t-i) + b_j^{(1)}\right), \tag{3.46}$$

$$\hat{y}(t+\Delta t) = \sum_j w_j^{(2)} h_j^{(t)} + b^{(2)}, \tag{3.47}$$

У випадку багатошарової TDNN для L прихованих шарів, цей процес повторюється на кожному з шарів, де вихід одного шару стає вхідним для наступного:

$$\mathbf{h}^{(1)} = \sigma^{(1)}(W^{(1)}\mathbf{x}^{(t)} + \mathbf{b}^{(1)}, \tag{3.48}$$

$$\mathbf{h}^{(l)} = \sigma^{(l)}(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}, l = 2, \dots, L, \tag{3.49}$$

$$\hat{y}(t+\Delta t) = W^{(L+1)}\mathbf{h}^{(L)} + b^{(L+1)}. \tag{3.50}$$

Інтегрована модель процесів прогнозування навантаження на вебсервер відображає узгоджену взаємодію етапів збору, попередньої обробки, декомпозиції, прогнозування та адаптивного управління навантаженням. У межах такої моделі особливе значення має етап вейвлет-аналізу, який забезпечує багаторівневе

представлення часових рядів вебтрафіку та дозволяє виділити як трендові, так і високочастотні компоненти, що характеризують миттєві коливання навантаження.

Однак ефективність цього етапу істотно залежить від вибору базисної (материнської) вейвлет-функції, яка визначає характер фільтрації сигналу, ступінь втрати інформації та здатність моделі відтворювати динаміку трафіку з мінімальною похибкою. Тому наступним логічним кроком після формування інтегрованої моделі є розроблення методу визначення найефективнішого типу материнського вейвлету, що дозволяє на основі багатокритеріальної оцінки (з урахуванням ортогональності, асиметрії, регулярності, компактності, геометричної подібності тощо) обрати оптимальну вейвлет-функцію для аналізу та прогнозування навантаження вебсерверів.

3.3. Метод визначення найефективнішого типу материнського вейвлету

Суть методу визначення найефективнішого типу материнського вейвлету полягає в систематичному багатокритеріальному оцінюванні вейвлет-функцій з урахуванням їхніх математичних, спектральних і структурних властивостей щодо характеру досліджуваного сигналу навантаження вебсервера. Метод передбачає порівняння вейвлетів-кандидатів (Daubechies, Symlet, Coiflet, Biorthogonal, Meyer, Haar тощо) за рядом кількісних і якісних критеріїв, що відображають придатність вейвлету для часово-частотного аналізу трафіку.

Метод визначення найефективнішого типу материнського вейвлету для задачі прогнозування навантаження на вебсервер базується на порівнянні декількох вейвлетів за чітко визначеними критеріями. Оскільки вибір вейвлету впливає на якість декомпозиції, точність моделі та обчислювальну ефективність, потрібно враховувати множину критеріїв ефективності:

Мінімізація похибки реконструкції забезпечує якісне відновлення вихідного сигналу після зворотного перетворення. В якості критерію виступає нормалізована похибка реконструкції:

$$\varepsilon_{rec} = \frac{\|x(t) - \hat{x}(t)\|_2}{\|x(t)\|_2}, \tag{3.51}$$

де $x(t)$– оригінальний сигнал, $\hat{x}(t)$ – реконструйований сигнал, отриманий після зворотного вейвлет-перетворення.

Енергетична концентрація в деталях (ефективність стиснення) максимізує кількість корисної інформації, сконцентрованої в малій кількості вейвлет-коефіцієнтів:

$$\gamma = \frac{\sum_{i=1}^{k}\left|D_i^j\right|^2}{\sum_{i=1}^{n}|x_i|^2}, k \ll n, \tag{3.52}$$

де D_i^j – вейвлет-коефіцієнти на j-му рівні декомпозиції, x_i – елементи вхідного сигналу. Чим вище значення γ, тим ефективніше сигнал подано через деталі з меншим числом коефіцієнтів.

Кореляція коефіцієнтів із трендами навантаження виявляє інформативні вейвлет-компоненти для прогнозування майбутніх значень навантаження:

$$\rho_j = corr\left(D^j(t), y(t+\Delta t)\right), \tag{3.53}$$

де $D^j(t)$ – детальна складова на рівні j, $y(t+\Delta t)$ – прогнозоване значення навантаження через інтервал Δt.

Високе значення ρ_j вказує на сильну кореляцію компоненти з майбутнім станом системи.

Стійкість до шуму дозволяє оцінити здатність вейвлет-декомпозиції ефективно відокремлювати шумові компоненти від сигналу:

$$SNR_{out} = 10\,log_{10}\left(\frac{Var\left(x_{signal}\right)}{Var(x_{noise})}\right), \tag{3.54}$$

де x_{signal} – відфільтрований корисний сигнал, x_{noise} – оцінка шуму.

Порівнюється до і після застосування вейвлету. Підвищення значення SNR_{out} після застосування вейвлет-фільтрації свідчить про ефективне приглушення шумів.

Кореляція з вихідним сигналом (Information Preservation) показує наскільки добре апроксимація та деталі зберігають характеристики вихідного часового ряду. Аналізується коефіцієнтом кореляції r між оригінальним і реконструйованим сигналом:

$$r = \frac{Cov(x, \hat{x}\text{^})}{\sigma_x \sigma_{\hat{x}}}. \quad (3.55)$$

Середньоквадратична похибка реконструкції (RMSE реконструкція) показує міру втрати інформації після прямого та зворотного перетворення (див. 3.32).

Компактність представлення (Sparsity) визначає кількість ненульових або домінантних коефіцієнтів після декомпозиції. Чим їх менше, тим вища стисліть подання. Ентропія у контексті аналізу вейвлет-декомпозиції визначається як:

$$H = -\sum_{i=1}^{N} p_i log_2(p_i), p_i = -\frac{|c_i|}{\sum_j |c_j|}, \quad (3.56)$$

де p_i – ймовірність (або відносна енергія) i-го коефіцієнта c_i вейвлет-перетворення, N – кількість коефіцієнтів.

Локалізація у часі та частоті (Time-Frequency Resolution) визначає здатність вейвлету виявляти короткотривалі пікові події. Вимірюється здатність вейвлету локалізувати сигнали в обох просторах – часу і частоти. Використаємо стандартний критерій невизначеності Гейзенберга:

$$\Delta t_j = \sqrt{\sum_i (t_i - \mu_t)^2 |D^j(t_i)|^2}, \Delta f_j = \sqrt{\sum_i (f_i - \mu_f)^2 |D^j(f_i)|^2}, \quad (3.57)$$

де Δt_j – дисперсія по часу для масштабу j, Δf_j– дисперсія по частоті, $D^j(t)$ – вейвлет-коефіцієнти для масштабу j, μ_t, μ_f– центри мас енергії по часу та частоті.

Інтегральна метрика:

$$\Delta_j = \Delta t_j \cdot \Delta f_j \quad (3.58)$$

Чим менше Δ_j – тим краща локалізація.

Адаптивність до країв (Edge effects) визначає наскільки вейвлет стабільний поблизу початку та кінця сигналу. Щоб оцінити стабільність вейвлет-перетворення біля початку і кінця сигналу та знайти амплітуду артефактів на краях при реконструкції використаємо відношення максимуму:

$$A_{edge} = \max_{t_i \in edges} |x(t_i) - \hat{x}(t_i)|, \quad (3.59)$$

де $edges = \{1, \ldots, k\} \cup \{n - k + 1, \ldots, n\}$ – краєві зони довжиною k, $\hat{x}(t_i)$ – реконструйований сигнал, $n = 2k$.

Прогнозна цінність (Predictive Contribution) дозволяє оцінити внесок вейвлет-перетворення у точність прогнозу. Порівнюються стандартні метрики прогнозування RMSE (3.32), MAE (3.33) і MAPE (3.34) до та після вейвлет-декомпозиції або між різними типами вейвлетів

Швидкість обчислення (Computational Efficiency) оцінює час виконання прямого та зворотного перетворення (особливо для потокових даних). Середній час обробки одного сигналу:

$$T_{avg} = \frac{T_{forward} - T_{inverse}}{n} \tag{3.60}$$

або ж в залежності від розміру batch:

$$T_{batch} = \frac{T_{total}}{B}, \tag{3.61}$$

де $T_{forward}$ – час декомпозиції, $T_{inverse}$ – час реконструкції, B – розмір батчу (кількість сигналів у пакеті).

Після розрахунку метрик для кожного кандидата, треба обрати один з методів верифікації для завершення аналізу:

– побудувати зважену мета-функцію якості;

– застосувати методи ранжування або багатокритеріальної оптимізації;

– обрати вейвлет за максимальним зростанням точності прогнозу.

Розглянемо кожен з методів більш детально.

1. *Побудова зваженої мета-функції якості (Weighted Quality Meta-Function)*. Нехай маємо m критеріїв оцінки якості вейвлета (наприклад, реконструкційна похибка, енергонасиченість, SNR, час обробки тощо).

Позначимо $Q_i(w)$ – значення i-го критерію для вейвлета w, $\omega_i \in [0,1]$ – вага i-го критерію (відображає його важливість), $\sum_{i=1}^{m} \omega_i = 1$.

Тоді мета-функція якості:

$$M(w) = \sum_{i=1}^{m} \omega_i Q_i(w). \tag{3.62}$$

Обираємо вейвлет із максимальним значенням $M(w)$.

2. *Багатокритеріальна оптимізація (Multi-Objective Optimization)*. Задачу

можна сформулювати як:

$$maximize_{w \in W} = [Q_1(w), Q_2(w), \dots, Q_m(w)], \quad (3.63)$$

де W – множина кандидатів (Daubechies, Haar тощо).

Застосуємо метод Pareto-оптимальності: шукаємо такі w, для яких немає іншого вейвлета, який би був кращим одночасно за всіма критеріями.

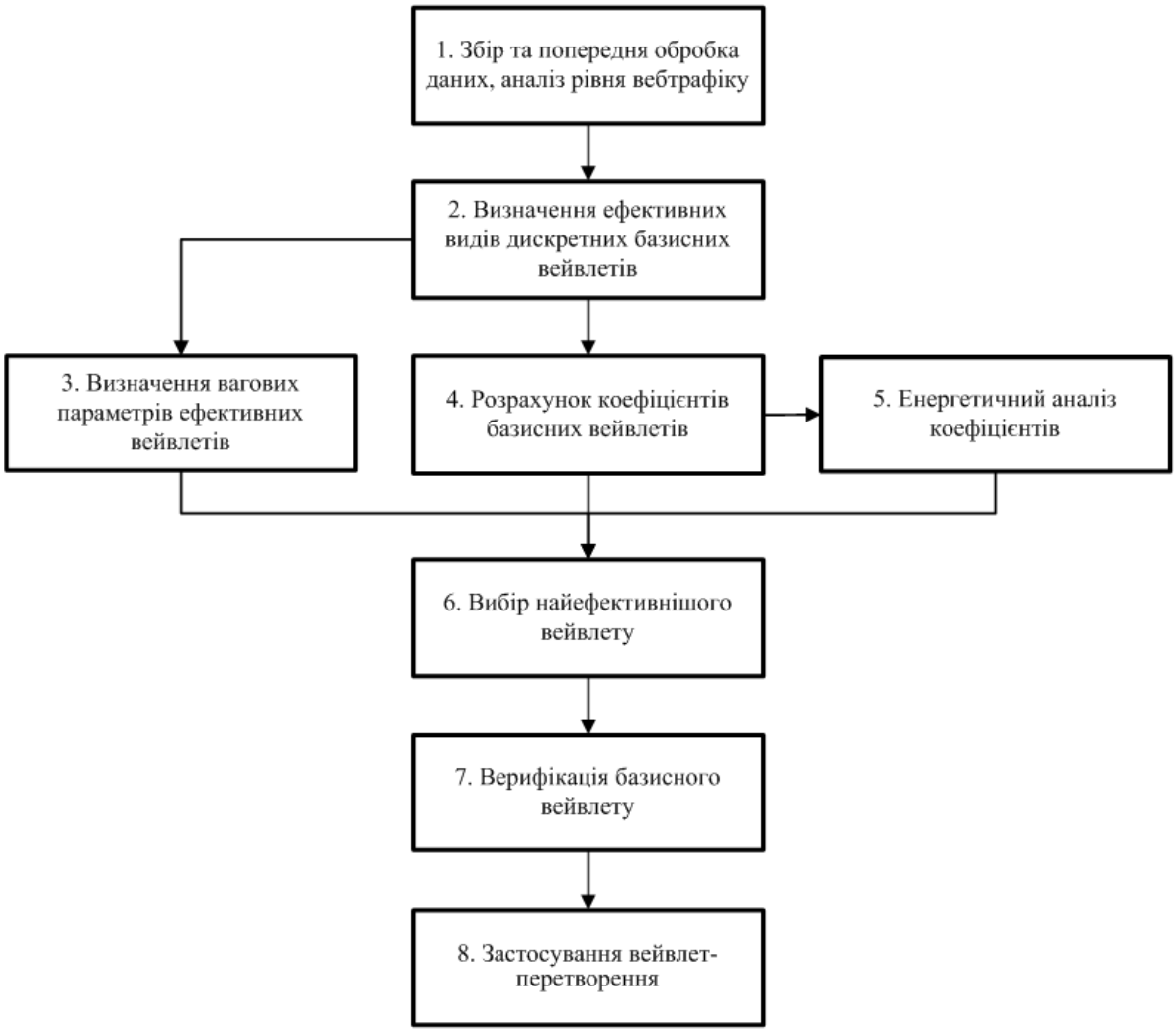

Рис. 3.5. Структурна схема методу визначення найефективнішого типу материнського вейвлету

3. *Вибір за приростом точності прогнозу*. Нехай $RMSE_{base}$ – похибка моделі без вейвлет-декомпозиції, а $RMSE_w$ – похибка з вейвлетом w.

Розрахунок приросту точності:

$$\Delta_{acc}(w) = \frac{RMSE_{base} - RMSE_w}{RMSE_{base}}. \quad (3.64)$$

Обирається вейвлет:

$$w^* = arg \max_{w \in W} \Delta_{acc}(w). \quad (3.65)$$

Оцінка вейвлетів за об'єктивними метриками дає змогу кількісно визначити ефективність кожного типу материнського вейвлету для обробки та аналізу навантаження на вебсервер.

Представлений метод для визначення найефективнішого типу материнського вейвлету залежить від конкретного застосування та властивостей сигналу чи функції, яку потрібно обробити. Основою для розробки методу вибору найефективнішого типу материнського вейвлету стали теоретичні дослідження [1, 3, 8, 33, 57]. Структурна схема запропонованого методу у вигляді послідовності з 8 етапів наведена на рис. 3.5.

Аналіз здійснювався в контексті вейвлет-перетворення дискретних даних, заданих на обмеженому часовому інтервалі. Конкретизовані етапи для реалізації такого методу з урахуванням множини критеріїв ефективності детально описано нижче.

Етап 1: Збір та попередня обробка даних, аналіз рівня вебтрафіку

Як вхідні дані $D_{in} = \{L, X, M\}$ розглянемо множину наступних компонентів: сирі логи вебсервера (Web Server Logs) L; часові ряди запитів (Time Series of Requests) X; метрики трафіку $M = \{IP, \tau, \Theta, \Sigma\}$, які деталізуються як: IP – множина унікальних IP-адрес користувачів, τ – часові мітки запитів (Timestamp series), Θ – типи HTTP-запитів (GET, POST, PUT тощо), Σ – коди статусів відповіді сервера (200, 404, 500 тощо).

На даному етапі для аналізу вибирається множина часових рядів запитів X навантаження на вебсервер. Для зменшення впливу різних масштабів даних на модель застосуємо Min-Max нормалізацію базуючись на принципі попередньої обробки та нормалізації даних. Нормалізація даних здійснюється за формулою (3.5). Процес нормалізації поділяється на кроки:

Крок 1 – визначення екстремальних значень. Проводиться аналіз усіх вхідних

даних для визначення $min(X)$ та $max(X)$.

Крок 2 – виконання нормалізації. Для кожного параметра виконується обчислення нормалізованого значення за формулою (3.5).

У результаті цього формується множина X_{norm}, в які значення даних приведені до масштабу, придатного для подальшого використання в ІМППНВ.

Крок 3 – аналіз рівня вебтрафіку. Перш ніж вибрати тип материнського вейвлету, необхідно оцінити властивості сигналу: частотний спектр сигналу (низькочастотні чи високочастотні компоненти), лінійність та неперервність сигналу, наявність різких змін (перехідних процесів), зашумленість сигналу.

На цьому кроці виконуємо очищення даних від шуму, заповнення пропусків (інтерполяція), визначення статистичних характеристик трафіку (середнє значення, дисперсія, розподіл), визначення типів і рівнів трафіку (пік, низька активність, сплески), агрегація трафіку, аналіз піків, сезонності, трендів; фільтрація шуму

Для оцінки середнього значення та дисперсії використаємо формули:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \, , \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 . \qquad (3.66)$$

Вихідними даними етапу є множина $D_{out} = \{\Gamma, \Phi, \Delta, \tilde{X}\}$, де Γ – статистичні характеристики трафіку (Traffic Statistics), Φ – оброблені дані для аналізу (Processed Feature Vectors), Δ – виявлені аномалії (Anomaly Sets), $\tilde{X}$ – нормалізовані та очищені часові ряди (Preprocessed Time Series).

Етап 2: Визначення ефективних видів дискретних базисних вейвлетів

В якості вхідних даних $D_{in} = \{\Phi, \tilde{X}, \Lambda\}$ для цього етапу використовуються оброблені дані трафіку Φ, очищений сигнал навантаження $\tilde{X}$, загальні характеристики навантаження на вебсервер $\Lambda = \{\tau_s, F_s, S_p\}$, а саме: τ_s – часовий формат сигналу (Time Format), F_s – частота вибірки (Sampling Frequency), S_p – показники сезонності (Seasonality Parameters).

Відповідно до результатів розділу 1, до переліку апробованих типів материнських вейвлетів віднесено:

- Хаар (Haar) – добре підходить для дискретних сигналів із різкими змінами,

але має обмежену гладкість.

– Добеші (Daubechies) – популярні для обробки сигналів, де потрібна компромісна гладкість і підтримка.

– Мейєр (Meyer) – використовуються в більш складних задачах, де важлива чіткість у частотному розкладі.

– Симлети (Symlets) – варіанти вейвлетів, схожі на Добеші, але з більшими симетріями.

– Коіфлети (Coiflets) – мають ще більш високі рівні гладкості і менше обчислювальне навантаження.

– Біортогональні (Biorthogonal) – застосовуються у випадках, коли потрібна висока точність відновлення сигналу.

В рамках даного етапу аналізуємо тип даних, виявляємо характер змін (стохастичність, циклічність), порівнюємо різні типи вейвлетів (Daubechies, Coiflet, Biorthogonal, Symlets тощо) за метриками: кореляція, RMSE, SNR, ентропія.

Таблиця 3.1

Оцінка вейвлетів за об’єктивними метриками

Вейвлет	Кореляція	RMSE	MAE	Ентропія	SNR (dB)	E_{1-3}	τ (мс)
Haar	0.91	0.153	0.112	2.10	16.5	0.78	1.2
Daubechies-4	0.97	0.097	0.075	1.53	21.2	0.85	2.8
Symlet-4	0.96	0.089	0.068	1.60	20.3	0.84	3.1
Coiflet-2	0.95	0.092	0.070	1.70	19.4	0.80	3.4
Biorthogonal 3.5	0.94	0.095	0.073	1.65	18.6	0.82	3.0
Mexican Hat	0.90	0.101	0.080	2.40	14.8	0.72	5.5

Основними критеріями виступають (табл. 3.1): коефіцієнт кореляції r між оригінальним і реконструйованим сигналом, що показує ступінь збереження структури даних; середньоквадратична помилка $RMSE$ як показник точності реконструкції; ентропія H, яка відображає ступінь інформаційного вмісту після

декомпозиції; відношення сигнал/шум SNR – для вимірювання рівня шумозахищеності; енергонасиченість на перших рівнях E_{1-3}, що ілюструє локалізацію важливої інформації на низьких масштабах; та час виконання τ, який характеризує обчислювальну ефективність.

Вибираємо тип вейвлетів за критерієм мінімізації помилок апроксимації. Серед доступних типів вейвлетів вибирається той, який при фіксованій кількості рівнів декомпозиції забезпечує мінімальну помилку відновлення сигналу після зворотного перетворення.

Нехай $x(t)$ – початковий сигнал навантаження на вебсервер), $\tilde{x}(t)$ – сигнал, відновлений після DWT та інверсного DWT, Ψ – набір доступних вейвлет-базисів (типів вейвлетів), $Err(\psi)$ – помилка апроксимації при використанні вейвлету $\psi \in \Psi$.

Цільова функція:

$$\psi^* = arg \min_{\psi \in \Psi} Err(\psi). \quad (3.67)$$

У якості цільової функції візьмемо такі типові метрики помилки як корінь середньоквадратичної помилки RMSE (3.32) і середню абсолютну помилку MAE (3.33).

З набору кандидатів Ψ для кожного $\psi \in \Psi$ здійснюємо DWT → обрізка/збереження коефіцієнтів → інверсне DWT.

Обчислюємо $Err(\psi)$ за однією з метрик.

Обираємо вейвлет ψ^*, який дає найменшу помилку і забезпечує найкращу апроксимацію сигналу при заданому рівні декомпозиції.

Зокрема, для наведеного прикладу в таблиці 3.1 обраним типом вейвлету буде $\psi^* = Symlet - 4$.

Як вихідні дані $D_{out} = \{W_b, R_q\}$ маємо: обраний ефективний базис дискретного вейвлету (Optimal Wavelet Basis) з визначеною глибиною дискретизації W_b, рейтингові оцінки кандидатних вейвлетів за метриками (Wavelet Ranking Vector) R_q.

Етап 3: Визначення вагових параметрів ефективних вейвлетів

Як вхідні дані $D_{in} = \{W_b, R_q\}$ розглядаємо вибрані ефективні вейвлети W_b, їх критерії оцінки R_q.

Крок 1 – побудова матриці оцінювання. Матриця оцінювання $\boldsymbol{R} = [r_{ij}] \in R^{n \times m}$, де r_{ij} – значення критерію i для вейвлета j, m – кількість критеріїв, n – кількість вейвлетів-кандидатів.

Візьмемо для прикладу 5 типів вейвлетів: haar, db4, sym4, coif1, bior3.3 і B₃-сплайн. Побудуємо таблицю 3.2 оцінювання всіх критеріїв ефективності r_i для розглянутих типів базисних вейвлетів.

Таблиця 3.2

Матриця експертного оцінювання критеріїв ефективності

Вейвлет	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
Haar	0,4	1,0	0,2	0,1	1,0	0,7	1,0	1,0	0,2
db4	0,8	0,8	0,7	0,6	1,0	0,9	1,0	0,8	0,7
sym4	0,8	0,8	0,8	0,5	1,0	0,9	1,0	0,9	0,8
coif1	0,7	0,7	0,9	0,4	1,0	0,8	1,0	0,7	0,7
bior3.3	0,8	0,7	0,8	1,0	0,5	0,9	1,0	0,8	0,6
B₃-сплайн	0,9	0,7	0,9	1,0	0,2	1,0	0,9	0,7	0,9

Оцінювання значущості кожного з критеріїв ефективності для даної задачі проводилося за допомогою експертного методу парних порівнянь. Отримані вагові коефіцієнти w_i (які можна скоригувати при потребі) наведено в таблиці 3.4.

Крок 2 – нормалізація вагових коефіцієнтів для подальшого ранжування. Застосовуємо алгоритм AHP (аналітичного ієрархічного процесу, Analytic Hierarchy Process) для ранжування. AHP дозволяє побудувати ієрархію критеріїв r_i, зробити попарні порівняння, знайти ваги критеріїв w_i, і нарешті – пріоритет кожного варіанту.

Після побудови матриці попарних порівнянь було отримано ваги критеріїв

w_i. Далі за кожним критерієм нормалізуються значення альтернатив, і обчислюється зважена сума.

Для кожного стовпця (критерію) нормалізуються значення (поділимо кожне значення на максимум по цьому стовпцю).

Після нормалізації помножимо кожне значення на відповідну вагу критерію.

Загальну ефективність i-го типу базисного вейвлету можна визначити за формулою:

$$R_i = \sum_{k=1}^{K} w_k r_k(i), \tag{3.68}$$

де R_i – інтегральний показник ефективності i-го типу вейвлету, $r_k(i)$ – значення k-го критерію для i-го типу, w_k – ваговий коефіцієнт відповідного критерію, K – загальна кількість критеріїв.

Найефективніший тип вейвлету визначається за формулою:

$$R_{eff} = \max_i R_i. \tag{3.69}$$

Інтегральний показник ефективності R_i кожного з протестованих типів базисних вейвлетів було обчислено шляхом підстановки значень у формули (3.51, 3.52). Результати розрахунків алгоритмом AHP подано в таблиці 3.3.

Таблиця 3.3

Інтегральні значення ефективності R апробованих типів базисних вейвлетів

Ранг	Вейвлет	R (AHP-оцінка)
1	sym4	0.427278
2	db4	0.421787
3	coif1	0.403546
4	bior3.3	0.401461
5	B3-spline	0.388734
6	Haar	0.288181

Таким чином, було визначено, що найвищу інтегральну ефективність в

наведеному прикладі для дослідженого конкретного виду навантаження на вебсервер демонструє вейвлет *sym4*.

В результаті виконання поточного етапу вихідні дані $D_{out} = \{\boldsymbol{R}, w, R_i\}$ становлять собою матрицю оцінок $\boldsymbol{R} = [r_{ij}]$ кандидатних вейвлетів, вектор w вагових коефіцієнтів критеріїв w_i для кожного типу вейвлету і інтегральний показник R_i, який дає змогу визначити найефективніший тип вейвлету.

Етап 4: Розрахунок коефіцієнтів базисних вейвлетів

В якості вхідних даних $D_{in} = \{w^*, \tilde{X}\}$ розглянемо вибраний ефективний базисний вейвлет w^*, очищений та нормалізований часовий ряд навантаження $\tilde{X}$.

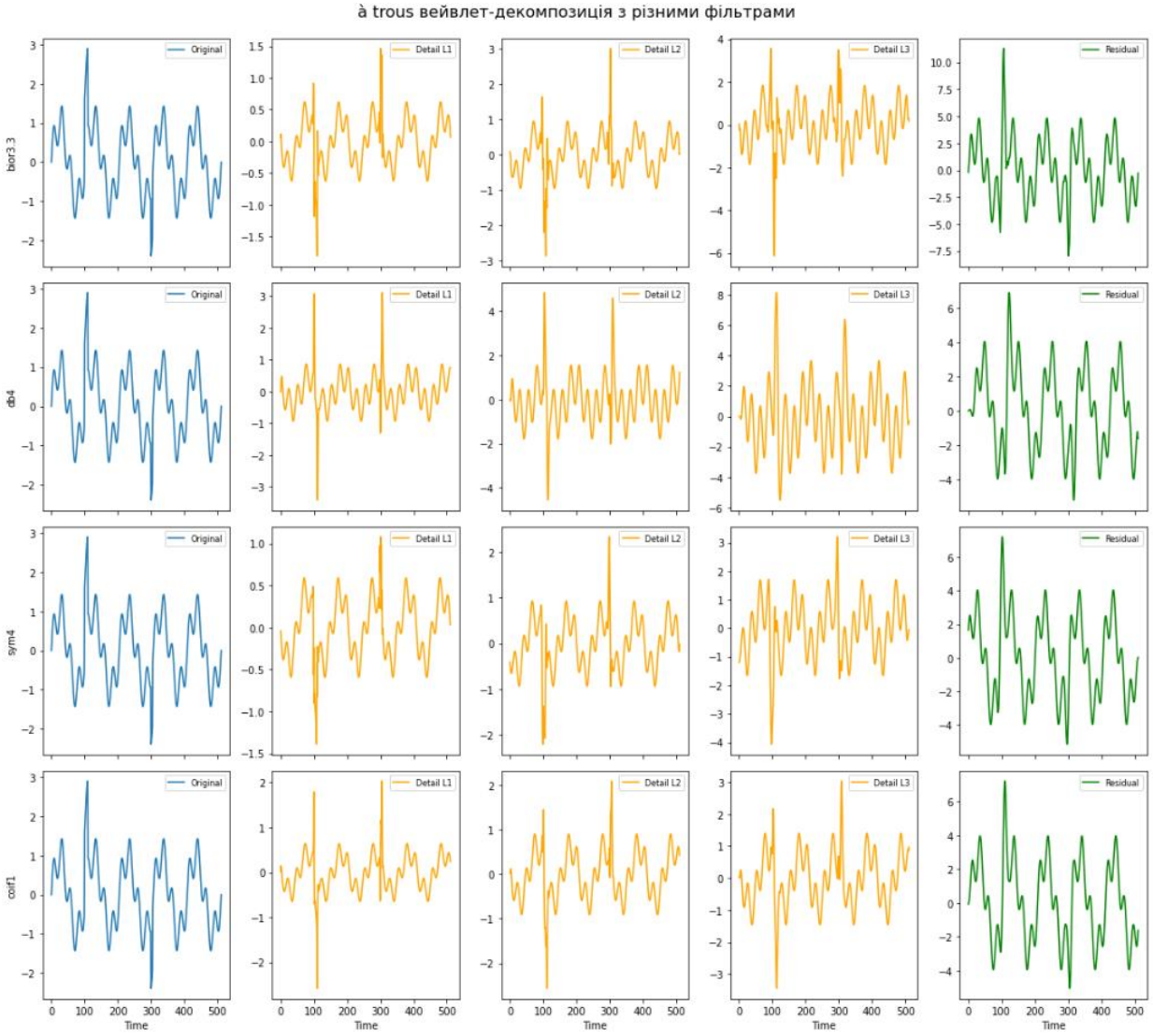

Рис. 3.6. *à trous* вейвлет-декомпозиція з різними фільтрами

В рамках цього етапу будемо виконувати дискретне вейвлет-перетворення для розкладу сигналу на базисні вейвлети, аналізуючи ентропію, та SNR на кожному рівні. Розглянемо задачу апроксимації часового ряду з використанням дискретного вейвлет-перетворення (ДВП), яке застосовується до вектора дискретних значень як вхідної вибірки. Вейвлети можна інтерпретувати як набір фільтрів низьких і високих частот, які здійснюють перетворення сигналу без зміни його довжини.

Використовується алгоритм *à trous* (з фр. «з дірками»), що характеризується збереженням розмірності сигналу при багаторівневій фільтрації (2.9).

На рис. 3.6 показано як змінюється вейвлет-декомпозиція одного й того самого сигналу при використанні різних материнських вейвлетів у алгоритмі *à trous*: перший стовпчик – це вихідний сигнал, наступні три стовпчики – деталізаційні компоненти на кожному рівні (від високочастотних до нижчих), останній стовпчик – залишкова апроксимація (згладжена версія сигналу).

Кожен рядок – це інший вейвлет: *bior3.3* (біортогональний, симетричний, підходить для точного локального згладжування); *db4* (Добеші, має хорошу компромісність між локалізацією у часі й частоті); *sym4* (симлет, подібний до Добеші, але більш симетричний); *coif1* (має високу згладжувальну здатність і підходить для виділення трендів).

В якості вихідних даних $D_{out} = \{\boldsymbol{C}, P_\psi\}$ маємо матриці вейвлет-коефіцієнтів $\boldsymbol{C} = [c_{jk}] \in R^{J \times K}$ з рівнем розкладу (декомпозиції) j і позиційним індексом вхідного ряду k; профілі розкладу $P_\psi = \{A_j, D_j\}_{j=1}^{J}$ з апроксимаційними коефіцієнтами (low-pass) A_j і детальними коефіцієнтами (high-pass) D_j.

Етап 5: Енергетичний аналіз коефіцієнтів

В якості вхідних даних маємо коефіцієнти базисних вейвлетів $D_{in} = \{\boldsymbol{C}\}$.

Крок 1 – обчислення енергії вейвлет-коефіцієнтів на кожному рівні розкладу базисних вейвлетів. Енергетичні коефіцієнти у контексті вейвлет-декомпозиції – це числові характеристики, які відображають внесок кожного рівня деталізації (вейвлет-коефіцієнтів) у загальну енергію сигналу. Вони дозволяють: кількісно

оцінити важливість кожного масштабу (деталей) у сигналі; порівнювати різні вейвлети щодо їх здатності концентрувати інформацію; вибирати ефективні рівні для скорочення, шумопригнічення або прогнозування.

Для кожного рівня j декомпозиції маємо вейвлет-коефіцієнти d_j. Енергія E_j обчислюється як сума квадратів значень коефіцієнтів:

$$E_j = \sum_{i=1}^{p} d_j(i)^2. \tag{3.70}$$

Це дозволяє оцінити, скільки інформації сигналу зберігається в кожному масштабі.

Аналогічно, для залишкового згладженого компонента (residual):

$$E_{res} = \sum_{i=1}^{p} a_p(i)^2. \tag{3.71}$$

Щоб порівняти внесок кожного рівня у відносних величинах, обчислюють нормовану енергію:

$$\varepsilon_j = \frac{E_j}{\sum_{k=1}^{p} E_k + E_{res}}. \tag{3.72}$$

Ці ε_j виражають частку енергії, яку несе кожен рівень. Їхня сума дорівнює 1.

Використовується критерій максимізації енергії важливих компонентів (в низьких рівнях коефіцієнтів) для кращої ефективності перетворення.

В результаті ми можемо обрати ефективний вейвлет, який концентрує більше енергії в малу кількість масштабів і має кращу здатність до аналізу сигналу, спрогнозувати навантаження (рівні з високою енергією найчастіше несуть найбільш релевантну інформацію), пригнітити шум (енергія допомагає виділити, де розташовані "слабкі" шумові компоненти).

Енергетична таблиця 3.4 показує, як розподіляється енергія сигналу по рівнях деталізації та залишковій апроксимації для різних вейвлетів у алгоритмі à trous: db4 має найбільшу енергію в деталізаціях (особливо на L3), що свідчить про здатність виявляти високочастотні флуктуації; bior3.3 залишає найбільше енергії в залишковій частині, що корисно, якщо потрібне збереження глобальної форми

сигналу; sym4 і coif1 – помірні, з більш м’яким балансом між деталями та трендом.

Таблиця 3.4

Розподіл енергії сигналу по рівнях деталізації та залишковій апроксимації

Вейвлет	Detail L1	Detail L2	Detail L3	Residual
bior3.3	81.26	190.26	709.24	4318.51
db4	187.27	627.82	2600.11	2965.64
sym4	68.50	173.36	530.96	3028.51
coif1	86.19	158.50	382.59	2966.23

Крок 2 – оцінка відновлення сигналу. Визначаємо, наскільки добре кожен вейвлет відновлює оригінальний сигнал за допомогою зворотного вейвлет-перетворення (Inverse Wavelet Transform).

Для кожного вейвлету обчислюється різниця між відновленим сигналом і оригінальним сигналом через корінь середньоквадратичної помилки RMSE (3.32). Використовується критерій мінімізації кореня середньої квадратичної помилки.

В якості додаткових критеріїв оцінювання ефективності використовуються такі показники як: коефіцієнт компресії – вимірює, наскільки ефективно вейвлет стисне сигнал без втрат; коефіцієнт відновлення – визначає, наскільки точно можна відновити сигнал після його розкладу і зворотного перетворення; евклідова норма L_2 – використовується для оцінки "похибок" або різниці між відновленим і оригінальним сигналом.

На основі енергетичного аналізу та оцінки помилки відновлення сигналу вибирається той вейвлет, який дає мінімальну помилку відновлення та зберігає найбільше енергії у важливих компонентах сигналу.

В якості вихідних даних $D_{out} = \{E_j, \varepsilon_{rec}\}$ маємо енергетичну характеристику коефіцієнтів, де E_j – компоненти енергії сигналу в базисі w^*, $\varepsilon_{rec} = \left\|\tilde{X} - \hat{X}\right\|_2^2$ – помилка відновлення сигналу після оберненого вейвлет-перетворення (квадрат евклідової норми як сума квадратів помилок на всіх дискретних кроках часової

шкали).

Етап 6: Вибір найефективнішого вейвлету

В якості вхідних даних $D_{in} = \{\varepsilon_i, \varepsilon_{rec}, w_i, r_{ij}, R_i\}$ маємо наступні метрики: енергетичні характеристики ε_i, вектор точності відновлення ε_{rec}, вагові коефіцієнти w_i, матрицю значень $\boldsymbol{R} = [r_{ij}]$, інтегральні показники R_i ефективності вейвлетів.

Цей етап використовується для можливої переоцінки початкового вибору вейвлету w^* – якщо кілька кандидатів перевіряються у розрахунках. Тоді вихідний w^* стає фінальним оптимумом базису із найбільш вигідним співвідношенням енергії та точності відновлення:

$$w^* = arg \max_{w_i \in W} \frac{\varepsilon_i}{\varepsilon_{rec\,i} + \delta}, \tag{3.73}$$

де $\delta \to 0$ – мале додатне число для уникнення ділення на нуль.

Алгоритм AHP базується на експертних оцінках, а енергетичний аналіз – це кількісна оцінка. Тому доцільно використати в рамках поточного етапу гібридний підхід, який поєднує: суб'єктивну експертну оцінку (AHP) і об'єктивну метрику – енергетичну ефективність вейвлет-декомпозиції.

Робоча формула для підсумкової ваги W_i кожного критерію r_i буде:

$$W_i = \alpha w_i^{AHP} + (1 - \alpha)\varepsilon_i, \tag{3.74}$$

де w_i^{AHP} – вага альтернативи з AHP; ε_i – нормалізоване значення енергії для альтернативи; $\alpha \in [0,1]$ – коефіцієнт довіри до експерта (наприклад, 0.7).

Таким чином, за підсумками поточного етапу будуть уточнені значення інтегрального показника R'_i, який дозволить визначити який тип вейвлету найефективніший за формулами:

$$R'_i = \sum_{k=1}^{K} W_k r_k(i), \tag{3.75}$$

$$R_{eff} = \max_i R'_i. \tag{3.76}$$

В якості вихідних даних $D_{out} = \{w^*\}$ маємо найефективніший тип вейвлету $w^* \in W$.

Етап 7: Верифікація базисного вейвлету

В якості вхідних даних $D_{in} = \{\tilde{X}, \hat{X}, w^*\}$ маємо фактичні нормалізовані та очищені $\tilde{X}$ часові ряди навантаження і прогнозовані $\hat{X}$ значення навантаження, отримані після застосування моделі з вейвлетом w^*. На цьому етапі визначаємо функцію втрат $L(w)$ для прогнозування, щоб оптимізувати вибір вейвлетів за критерієм мінімізації помилок. Це будуть стандартні метрики прогнозування RMSE (3.32), MAE (3.33) і MAPE (3.34).

Таблиця 3.5

Вплив вейвлет-декомпозиції на точність прогнозу (після LSTM)

Вейвлет	MAE	RMSE	MAPE (%)	Training Time (s)
Haar	0.103	0.145	12.3%	31.4
Daubechies-4	0.071	0.109	8.7%	38.2
Symlet-4	0.075	0.113	9.2%	40.1
Coiflet-2	0.080	0.120	10.0%	41.7
Biorthogonal 3.5	0.078	0.118	9.6%	39.6
Mexican Hat	0.115	0.162	14.1%	29.8

Таблиця 3.6

Зведена таблиця ранжування вейвлетів

Вейвлет	Якість декомпозиції (max 10)	Прогнозна точність (max 10)	Загальна оцінка (max 20)
Daubechies-4	9.5	9.2	18.7
Symlet-4	9.0	8.9	17.9
Biorthogonal 3.5	8.8	8.7	17.5
Coiflet-2	8.5	8.5	17.0
Haar	7.0	7.2	14.2
Mexican Hat	6.5	6.5	13.0

Після вибору вейвлету модель перевіряється на нових наборах даних для оцінки її здатності прогнозувати навантаження на вебсервер (табл. 3.5).

Якщо модель дає невідповідні прогнози, можливе коригування типу вейвлету або параметрів моделі для поліпшення результатів (табл. 3.6).

В якості вихідних даних $D_{out} = \{Rank(w_i), w^*, L(w)\}$ будуть остаточний рейтинг ефективності всіх досліджених вейвлетів $Rank(w_i)$, рекомендований тип материнського вейвлету w^*, функція втрат $L(w) = \{RMSE, MAE, MAPE\}$ прогнозування з показниками точності моделі RMSE, MAE і MAPE.

Етап 8: Застосування вейвлет-перетворення

В якості вхідних даних $D_{in} = \{A_J(t), D_j(t)\}$ маємо матриці апроксимаційних коефіцієнтів $A_J(t)$ і матриці детальних коефіцієнтів $D_j(t)$.

Після остаточного з'ясування, який тип вейвлет-перетворення буде для конкретного набору даних навантаження на веб-сервер найефективнішим, в рамках цього етапу зберігаємо матриці апроксимаційних коефіцієнтів $A_J(t)$, які репрезентують загальні тренди і матриці детальних коефіцієнтів $D_j(t)$, що відображають швидкі зміни у трафіку (сплески навантаження) в межах відповідного масштабу j.

В якості вихідних даних $D_{out} = \{\widehat{A_J}(t), \widehat{D_J}(t)\}$ маємо збережені матриці апроксимаційних коефіцієнтів $\widehat{A_J}(t)$ для довгострокового прогнозування і детальних коефіцієнтів $\widehat{D_J}(t)$, для короткострокового прогнозування.

Для наочності представимо у таблиці 3.7 структурований короткий опис всіх етапів реалізації методу визначення найефективнішого типу материнського вейвлету.

Розроблений метод визначення найефективнішого типу материнського вейвлету забезпечує обґрунтований вибір базисної функції, яка найкраще відповідає характеристикам досліджуваного сигналу вебнавантаження. Вибір оптимального вейвлету дає змогу підвищити точність декомпозиції часових рядів трафіку, мінімізувати інформаційні втрати та забезпечити адекватне відображення локальних і глобальних змін у структурі навантаження.

Таблиця 3.7

Структура етапів методу визначення найефективнішого типу материнського вейвлету

№	Назва етапу	Вхідні дані D_{in}	Вихідні дані D_{out}
1	Збір та попередня обробка даних, аналіз рівня вебтрафіку	$\{L, X, M\}$	$\{\Gamma, \Phi, \Delta, \tilde{X}\}$
2	Визначення ефективних видів дискретних базисних вейвлетів	$\{\Phi, \tilde{X}, \Lambda\}$	$\{W_b, R_q\}$
3	Визначення вагових параметрів ефективних вейвлетів	$\{W_b, R_q\}$	$\{\boldsymbol{R}, w, R_i\}$
4	Розрахунок коефіцієнтів базисних вейвлетів	$\{w^*, \tilde{X}\}$	$\{\boldsymbol{C}, P_\psi\}$
5	Енергетичний аналіз коефіцієнтів	$\{\boldsymbol{C}\}$	$\{E_j, \varepsilon_{rec}\}$
6	Вибір найефективнішого вейвлету	$\{\varepsilon_i, \varepsilon_{rec}, w_i, r_{ij}, R_i\}$	$\{w^*\}$
7	Верифікація базисного вейвлету	$\{\tilde{X}, \hat{X}, w^*\}$	$\{Rank(w_i), w^*, L(w)\}$
8	Застосування вейвлет-перетворення	$\{A_J(t), D_j(t)\}$	$\{\widehat{A_J}(t), \widehat{D_j}(t)\}$

На основі отриманих результатів стає можливим формування методу прогнозування навантаження на вебсервер, який використовує обраний вейвлет як основу для попередньої обробки даних. Такий підхід дозволяє здійснювати ефективне розділення сигналу на складові частоти, зменшувати вплив шумів та передавати очищений і структурований часовий ряд до нейронної мережі прогнозування. У результаті забезпечується підвищення достовірності прогнозу, швидкості адаптації моделі до динаміки трафіку та зменшення середньої похибки передбачення навантаження на сервер.

3.4. Метод прогнозування навантаження на вебсервер

Суть методу прогнозування навантаження на вебсервер полягає у використанні інтегрованого поєднання часово-частотного аналізу сигналу за допомогою вейвлет-перетворення з глибинним нейронним прогнозуванням, що дозволяє з високою точністю передбачати зміну інтенсивності вебзапитів і забезпечувати оптимальне використання обчислювальних ресурсів серверної інфраструктури.

Метод прогнозування навантаження на вебсервер базується на визначеній множині критеріїв ефективності, які використовуються для оцінки якості роботи моделі та її здатності прогнозувати навантаження:

Багатомасштабність обробки даних зазвичай включає застосування методів, які дозволяють працювати з даними на різних рівнях або з різною роздільною здатністю, наприклад, через використання дискретних вейвлет-перетворень (DWT).

В контексті прогнозування навантаження на вебсервер цей критерій грає вирішальну роль аналізуючи дані на різних часових або просторових масштабах, тобто: глобальний тренд навантаження (повільні зміни протягом годин/днів), локальні аномалії (раптові стрибки трафіку – спайки, DoS, акції), періодичність (денна, тижнева сезонність тощо). На великому масштабі (низькі частоти): видно загальну тенденцію росту або падіння навантаження – це допомагає довгостроковому прогнозу. На малому масштабі (високі частоти): видно піки, короткі викиди, зумовлені кампаніями, ботами, атаками – це допомагає миттєвому реагуванню та точному короткостроковому прогнозу.

Математично це можна представити через декомпозицію даних на кілька масштабів або рівнів:

$$x(t) = \sum_{j=1}^{J} \sum_{k=1}^{K} \psi_{j,k}(t) \cdot c_{j,k}, \tag{3.77}$$

де $x(t)$ – початковий сигнал або дані, J – кількість рівнів декомпозиції, K – кількість точок або частин на кожному рівні, $\psi_{j,k}(t)$ – вейвлет-функція для масштабу j, $c_{j,k}$

– коефіцієнти вейвлет-перетворення на рівні j.

Виділення деталей з даних реалізується через фільтрацію сигналу допомогою вейвлет-перетворення, з виділенням високочастотних компонентів, що відповідають за деталі. Це може бути виражено через різницю між вихідними та загальними тенденціями:

$$x_{details}(t) = x(t) - \hat{x}(t), \tag{3.78}$$

де $x(t)$ – оригінальний сигнал (дані), $\hat{x}(t)$ – наближене значення або тренд, отриманий шляхом фільтрації (наприклад, через низькочастотний фільтр або середнє значення).

В цьому випадку виділяються деталі, що характеризують малі коливання або аномалії в даних, які можуть бути корисними для прогнозування.

Збереження інформації забезпечує мінімізацію втрат при обробці даних, що досягається шляхом оптимального вибору коефіцієнтів при декомпозиції. Це можна виразити як:

$$E = \sum_{j=1}^{J} \sum_{k=1}^{K} \left| c_{j,k} \right|^2, \tag{3.79}$$

де E – збережена інформація (енергія сигналу), $c_{j,k}$ – коефіцієнти декомпозиції для рівня j і точки k.

Метою критерія є максимізація збереженої енергії сигналу при мінімальних втратах, що дозволяє зберегти основну структуру даних.

Локальне прогнозування передбачає застосування моделей до невеликих підмножин даних для отримання короткострокових прогнозів. Математично це можна виразити через локальну регресію фрагменту даних:

$$\hat{y}(t_0) = f(x(t_0), \theta), \tag{3.80}$$

де $\hat{y}(t_0)$ – прогноз на момент t_0, $x(t_0)$ – вхідні дані на момент t_0, $f(\cdot, \theta)$ – модель прогнозування (наприклад, LSTM), θ – параметри моделі.

Локальне прогнозування дозволяє фокусуватися на малих ділянках часу або простору, підвищуючи точність прогнозу для короткострокових змін.

Інтеграція прогнозів полягає в комбінуванні кількох моделей або підходів

для отримання єдиного прогнозу. Це можна зробити за допомогою методів ансамблю, наприклад, вагового середнього:

$$\hat{y}_{\sum} = \sum_{i=1}^{m} w_i \, \hat{y}_i, \tag{3.81}$$

де $\hat{y}_i$ – прогноз від i-ї моделі, w_i – вага i-ї моделі, яка може бути визначена через точність або інші критерії, m – кількість моделей або прогнозів.

Цей критерій дозволяє об'єднати прогнози різних моделей для покращення загальної точності прогнозу.

Точність прогнозу вимірюється як відсоток правильних прогнозів або як різниця між прогнозованим і фактичним значенням:

$$Acc = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i = y_i) * 100\%, \tag{3.82}$$

де $\hat{y}_i$ – прогнозоване значення на i-му кроці, y_i – реальне значення на i-му кроці, $I(\hat{y}_i = y_i)$ – індикаторна функція, що дорівнює 1, якщо прогноз правильний, і 0 в іншому випадку, n – кількість спостережень.

Об'єднаний критерій помилок (ОКП, Combined Error Metric, CEM)– комплексний критерій, який оцінює різні типи помилок прогнозу. Для цього ми можемо комбінувати середню квадратичну помилку RMSE (3.32), середню абсолютну помилку MAE (3.33) та середню абсолютну процентну помилку MAPE (3.34) в одну єдину метрику, що дасть змогу оцінювати як абсолютні помилки, так і відносні помилки:

$$CEM = w_1 \cdot RMSE + w_2 \cdot MAE + w_3 \cdot MAPE, \tag{3.83}$$

де w_1, w_2, w_3 – ваги для кожного з критеріїв, що можна налаштовувати в залежності від пріоритетів, $w_1 + w_2 + w_3 = 1$.

ОКП дозволяє комбінувати різні аспекти точності прогнозу в одному показнику, даючи змогу враховувати як абсолютні, так і відносні помилки. MAE дає загальну уяву про середню величину помилки. RMSE чутливий до великих помилок, що може бути важливим у контексті прогнозування навантаження (де пікові навантаження можуть мати критичне значення). MAPE дає відносну оцінку

точності прогнозу, що важливо для порівняння результатів на різних рівнях навантаження або між різними моделями.

Часова складність моделі описується через функцію складності, яка визначається кількістю операцій, що виконуються під час навчання або прогнозування:

$$T(n) = O\big(f(n)\big), \tag{3.84}$$

де $T(n)$ – часова складність, $f(n)$ – функція, що описує час, який витрачається на обробку n елементів.

Ресурсна ефективність RE може бути виміряна як співвідношення використаних ресурсів Res (наприклад, пам'яті, процесора) до кількості оброблених даних D:

$$RE = \frac{Res}{D}. \tag{3.85}$$

Стійкість до шуму як окремий критерій оцінюється через зміну точності при введенні шуму в дані:

$$Rob = \frac{Accuracy_{noise}}{Accuracy_{clear}} \times 100, \tag{3.86}$$

де $Accuracy_{noise}$ – точність прогнозів на зашумлених даних, $Accuracy_{clear}$ – точність прогнозів на чистих даних.

Спроможність до адаптації моделі може бути виміряна через швидкість, з якою модель здатна оновлювати свої параметри або адаптуватися до нових умов. Це виразимо через кількість ітерацій I_{conv}, необхідних для досягнення стабільного прогнозу:

$$Adapt = \frac{I_{conv}}{T_{avail}}. \tag{3.87}$$

де T_{avail} – доступний час або кількість нових даних для адаптації.

Можливість інтеграції з іншими системами оцінимо через час T_{depl}, необхідний для впровадження моделі в реальну систему та через кількість потрібних налаштувань R_{config} для інтеграції:

$$Integration\ Time = T_{depl} + R_{config}. \tag{3.88}$$

Загальна якість моделі (Overall Model Quality, OMQ) виражається через комбінацію різних критеріїв ефективності, наприклад, зважену середню величину всіх попередніх метрик:

$$OMQ = \sum_{i=1}^{m} w_i M_i, \tag{3.89}$$

де w_i – вага кожного критерію, M_i – значення кожного критерію ефективності, m – кількість критеріїв.

Об'єднана метрика для критеріїв прогнозування та обробки даних. Всі ці критерії можна комбінувати в одну загальну метрику, яка визначатиме ефективність прогнозу на основі багатомасштабної обробки даних, виділення деталей, збереження інформації, локального прогнозування та інтеграції прогнозів:

$$\hat{y}_f = \alpha \sum_{j=1}^{J} \sum_{k=1}^{K} \left|c_{j,k}\right|^2 + \beta\big(x(t) - \hat{x}(t)\big) + \gamma \sum_{i=1}^{m} w_i \hat{y}_i, \tag{3.90}$$

де $\hat{y}_f$ – кінцевий прогноз, w_i – вага кожної моделі для інтеграції прогнозів, $c_{j,k}$ – коефіцієнти декомпозиції для багатомасштабного аналізу, $x(t) - \hat{x}(t)$ – деталі, виділені з сигналу, α, β, γ – коефіцієнти, що регулюють вагу критеріїв багатомасштабної обробки, виділення деталей та інтеграції прогнозів.

Таким чином, багатомасштабність та виділення деталей – це важливі аспекти для обробки сигналу та забезпечення якості прогнозу через збереження та виділення значущої інформації. Інтеграція прогнозів дозволяє комбінувати прогнози від різних моделей або підходів для покращення точності.

Для поєднання всіх наборів критеріїв (ефективності прогнозу та критеріїв обробки даних) запишемо комплексну метрику, яка включає як показники точності та якості прогнозу, так і характеристики обробки даних на різних рівнях.

Це дозволить мати більш збалансовану оцінку моделі, що враховує як точність, так і здатність обробляти дані на багатьох масштабах, зберігати інформацію, виділяти деталі та здійснювати інтеграцію прогнозів:

$$\hat{y}_{final} = w_1 \cdot Acc + w_2 \cdot CEM + w_3 \cdot RE + w_4 \cdot Rob + w_5 \cdot Adapt + \hat{y}_f, \tag{3.91}$$

де Acc – точність прогнозу (як показано раніше), CEM – ОКП, RE – ресурсна

ефективність, $Adapt$ – здатність до адаптації моделі, Rob – стійкість до шуму, w_1, w_2, w_3, w_4, w_5 – ваги для критеріїв точності, помилок, ресурсної ефективності, стійкості та адаптації.

Рис. 3.7. Структурна схема методу прогнозування навантаження на вебсервер

Ресурсна ефективність визначає, наскільки економно модель використовує

ресурси для прогнозування. Адаптивність показує, як швидко модель може адаптуватися до змін у даних або умовах. Стійкість до шуму вимірює, наскільки добре модель працює при наявності шуму в даних.

Конкретизовані етапи (рис. 3.7) для реалізації такого методу з урахуванням розглянутої множини критеріїв ефективності детально описано нижче.

Етап 1: Оцінка стійкості до шуму та зміни даних

Вхідними даними $D_{in} = \{w_m, \omega, X_{train}, \mu_{sys}\}$ є материнський вейвлет w_m, вагові коефіцієнти якості вейвлетів $\omega = \{\omega_1, \dots, \omega_k\}$, тренувальні вибірки часових рядів X_{train}, метрики (CPU, RAM, disk I/O) системного навантаження $\mu_{sys} = \{\mu_{CPU}, \mu_{RAM}, \mu_{IO}\}$.

На даному етапі перевіримо чутливість вибору вейвлету до шуму в даних шляхом тестування моделі на синтезованих зашумлених даних.

Застосуємо попередню Z-нормалізацію:

$$x_{norm} = \frac{x - \mu}{\sigma}. \tag{3.92}$$

де μ – середнє значення часового ряду навантаження на вебсервер, σ – стандартне відхилення ряду.

Застосуємо медіанний фільтр для виявлення та згладжування викидів, проаналізуємо спектр шуму в сигналі через енергетичний розподіл вейвлет-коефіцієнтів:

$$E_j = \sum_t \left| w_j(t) \right|^2. \tag{3.93}$$

Нарешті розрахуємо стабільність прогнозу на доданому гаусовому шумі:

$$SNR = \frac{\sum_{i=1}^{N} f(x_i)^2}{\sum_{i=1}^{N} (f(x_i) - \hat{f}(x_i))^2}, \tag{3.94}$$

де SNR – відношення сигналу до шуму, $f(x_i)$ – реальні значення, $\hat{f}(x_i)$ – прогнозовані значення.

Вихідними даними $D_{out} = \{X, \chi(w_m)\}$ будуть фільтровані (стабілізовані) часові ряди X, показник чутливості моделі до шуму $\chi(w_m) \in [0,1]$.

Етап 2. Формування множини вхідних параметрів

В якості вхідних даних $D_{in} = \{X, \nabla X(t), Tr(X), A_J(t), D_j(t)\}$ розглянемо часовий ряд запитів X, вейвлет-декомпозовані компоненти $A_J(t), D_j(t)$, похідні ознаки (градієнт $\nabla X(t)$, тренд $Tr(X)$ тощо).

На даному етапі сформуємо множину вхідних параметрів моделі використовуючи ковзні середні (Simple Moving Average, SMA) у момент часу t для зменшення шуму в даних і виявлення локальних трендів:

$$SMA = \frac{1}{n} \sum_{i=\mathrm{t}-n+1}^{t} x_i, \tag{3.95}$$

де x_i – значення часового ряду на момент i, n – розмір вікна згладжування.

Ковзна дисперсія (Moving Variance) дозволяє оцінити локальну волатильність:

$$Var = \frac{1}{n-1} \sum_{i=\mathrm{t}-n+1}^{t} (x_i - SMA)^2. \tag{3.96}$$

Для захоплення часової залежності вводяться лагові змінні:

$$Lag_k(x_t) = x_{t-k}. \tag{3.97}$$

Таким чином можна створити набір ознак, що відображають попередні стани системи на k кроків назад для тимчасової пам’яті у моделі.

Застосоване в попередньому методі дискретне вейвлет-перетворення (алгоритм à trous) передає дані для декомпозиції сигналу на різнорівневі компоненти (3.29), де кожна з компонент може виступати як окрема ознака, яка виявляє коливання у різних масштабах часу (від пікових змін до глобального тренду).

Для кожної детальної компоненти D_j вейвлету враховується її енергія E_j (3.11), щоб мати змогу визначити, які частотні діапазони сигналу містять найбільшу інформацію.

На виході маємо $D_{out} = \{\boldsymbol{X}\}$ матрицю ознак $\boldsymbol{X} \in \mathbb{R}^{T \times m}$, де кожен рядок – це вектор вхідних параметрів у момент часу t, а кожен стовпчик – одна з ознак (оригінальний сигнал, лаги, SMA, вейвлети тощо):

$$\boldsymbol{X}_t = [x_t, Lag_1(x_t), SMA, Var, D_1, E_1, \dots]. \quad (3.98)$$

Етап 3. Вибір моделі та моделювання

В якості вхідних даних $D_{in} = \{\boldsymbol{X}_t, y_t, y_{t+k}, k\}$ маємо матрицю ознак $\boldsymbol{X}_t \in \mathbb{R}^m$ – вектор вхідних ознак у момент часу t, цільову змінну $y_t \in \boldsymbol{X}_t \in \mathbb{R}^m$, тобто значення навантаження на вебсервер, y_{t+k} – реальне значення навантаження у момент часу $t + k$ (де $k \in \mathbb{Z}^+$ – горизонт прогнозування).

На даному етапі, потрібно побудувати модель, яка наближує функціональну залежність:

$$y_t = f(\boldsymbol{X}_t) + \varepsilon_t, \quad (3.99)$$

де ε_t – випадкова похибка.

Будуємо LSTM (Long Short-Term Memory) для обробки часових послідовностей навантаження на вебсервер:

$$h_t = LSTM(x_t, h_{t-1}, c_{t-1}), \quad (3.100)$$

$$\hat{y}_{t+1} = \boldsymbol{W}_{out} \cdot h_t + b, \quad (3.101)$$

де x_t – вхідний вектор ознак у момент часу t (трафік, похідні, день тижня), h_t – прихований стан (short-term memory), c_t – комірка пам’яті (long-term memory), $\boldsymbol{W}_{out}$ і b – ваги й зсув вихідного шару.

Добре підходить для прогнозування на основі лагових послідовностей $[x_{t-n}, \dots, x_t]$, дозволяючи утримувати як короткострокові, так і довгострокові залежності у часових рядах шаблонів трафіку.

Використовуємо Wavelet Neural Network (WNN) як комбінацію класичної нейронної мережі з вейвлет-базисними функціями як функціями активації:

$$\hat{y}_t = \sum_{j=1}^{N} w_j \psi\left(\frac{x_t - a_j}{b_j}\right), \quad (3.102)$$

де ψ – материнська вейвлет-функція, a_j – параметр зсуву (center), b_j – параметр масштабування (ширина вейвлету), w_j – ваговий коефіцієнт

WNN дозволяє вбудовувати вейвлетне бачення в глибину мережі – особливо добре для нерегулярних, імпульсних сигналів. Базується на попередньо обраному типу материнського вейвлету та ініціалізованих параметрах a_j, b_j.

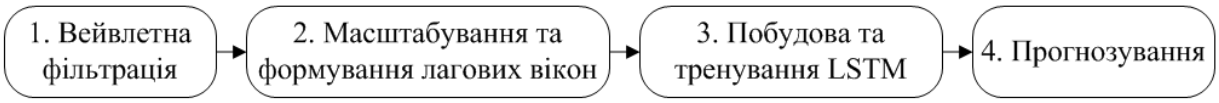

Рис. 3.8. Стадії реалізації гібридного підходу до прогнозування навантаження

WNN робить початкову фільтрацію (декомпозицію та реконструкцію), а потім LSTM прогнозує результати (рис. 3.8).

Параметри (кількість шарів, нейронів) підбираються за допомогою пошуку по сітці / випадкових стратегій. Узагальнено архітектура представлена як:

$$y_t = f\left(W_{(i,j)}\right), \tag{3.103}$$

де $W_{(i,j)}$ – параметри ваг нейронної мережі між шарами i та j.

Для навчання моделей використовуються класичний оптимізатор Adam (Adaptive Moment Estimation) – використовує середнє значення першого моменту (градієнтів) і другого моменту (квадратів градієнтів).

В якості функції втрат застосовуємо – корінь з середньої квадратичної похибки RMSE (3.32).

Для перехресної валідації розбиваємо дані на навчальну, валідаційну та тестову частини, щоб перевірити узагальнювальну здатність моделі.

Щоб оцінити якість моделі застосовуємо середню абсолютну похибку MAE (3.33), середню абсолютну процентну помилку MAPE (3.34) та коефіцієнт детермінації R^2:

$$R^2 = 1 - \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{N}(y_t - \bar{y}_t)^2}. \tag{3.104}$$

На завершення етапу отримується навчена модель, здатна прогнозувати навантаження вебсервера на основі сформованого набору ознак.

Вихідними даними $D_{out} = \{\hat{f}(\boldsymbol{X}_t|\theta), \theta, L(\theta)\}$ будуть навчена модель прогнозування $\hat{f}(\boldsymbol{X}_t|\theta)$, параметри θ моделі, які налаштовуються в процесі навчання, функція втрат (обрана відповідно до особливостей задачі) $L(\theta) = \{MAE, MAPE, RMSE, R^2\}$.

Етап 4. Прогнозування навантаження вебсерверу

В якості вхідних даних $D_{in} = \{Z_t\}$ розглянемо векторизовані вхідні ознаки, сформовані на попередніх етапах. Набір вхідних ознак для прогнозування на момент $t + k$ (горизонт передбачення) позначимо як:

$$Z_t = \left\{x_t, x_{t-1}, x_{t-2}, \dots, x_{t-k}, W_t^{(j)}, \nabla x_t, trend(x_t), \mu_t^{(w)}, \sigma_t^{(w)}, Lag_k(x_t)\right\}. \quad (3.105)$$

Це поточні значення ознак x_t, історичні значення $x_{t-1}, x_{t-2}, \dots, x_{t-k}$, вейвлет-декомпозовані компоненти $W_t^{(j)}$, похідні ознаки: градієнти, тренди, ковзні середні $\mu_t^{(w)}$, дисперсії $\sigma_t^{(w)}$ за вікно w, лагові змінні $Lag_k(x_t) = x_{t-k}$.

На даному етапі після подачі вхідних даних здійснюється оцінка майбутнього значення навантаження за допомогою навченого предиктора:

$$\hat{y}_{t+1} = f(x_t, x_{t-1}, x_{t-2}, \dots, x_{t-k}), \quad (3.106)$$

де $f(\cdot)$ – функція моделі (LSTM, WNN тощо), $\hat{y}_{t+1}$ – прогнозоване значення навантаження через одиницю часу.

Для моделей типу гібридних архітектур із вейвлет-декомпозицією, результат обчислюється через реконструкцію сигналу на основі усіх компонентів. У моделі прогнозування кожен детальний компонент $d_j(t)$ прогнозується окремо. Фінальний прогноз формується як:

$$\hat{x}(i) = \sum_{j=1}^{p+1} \widehat{W}(i,j), \quad (3.107)$$

де $\widehat{W}(i,j)$ – передбачене значення вейвлет-компоненти на j-му рівні для моменту часу i, а $p+1$ включає і залишкову складову a_p. Завдяки адитивній природі вейвлет-розкладу прогнозні компоненти комбінуються гібридно, з урахуванням ваг, що мінімізують глобальну помилку:

$$\hat{x}(i) = \sum_{j=1}^{p+1} \alpha_j \widehat{W}(i,j), \sum_{j=1}^{p+1} \alpha_j = 1, \quad (3.108)$$

де α_j – вагові коефіцієнти, визначені на основі зворотної помилки кожної компоненти.

Вихідними даними $D_{out} = \{\hat{Y}_t, plot(y_t, \hat{y}_t)\}$ будуть передбачене значення

навантаження $\hat{Y}_t = \{\hat{y}_{t+1}, \hat{y}_{t+2}, \dots, \hat{y}_{t+\mathrm{k}}\}$ на заданий горизонт k, графік відображення фактичних і прогнозованих значень у вигляді часового ряду.

Етап 5. Автоматизоване налаштування гіперпараметрів

В якості вхідних даних $D_{in} = \{\Theta, \boldsymbol{X}, \boldsymbol{y}\}$ розглянемо простір гіперпараметрів моделі $\Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ – кількість шарів, кількість нейронів, розмір вікна, learning rate, dropout rate тощо; раніше сформовану матрицю ознак $\boldsymbol{X}$ та відповідні мітки (об'єкти навчання) цільової змінної $\boldsymbol{y}$.

Автоматизований підбір гіперпараметрів є критично важливим етапом у побудові високоточних моделей прогнозування навантаження. Ефективно налаштовані параметри моделі забезпечують кращу узагальнювальну здатність, знижують помилки прогнозу та підвищують стабільність системи.

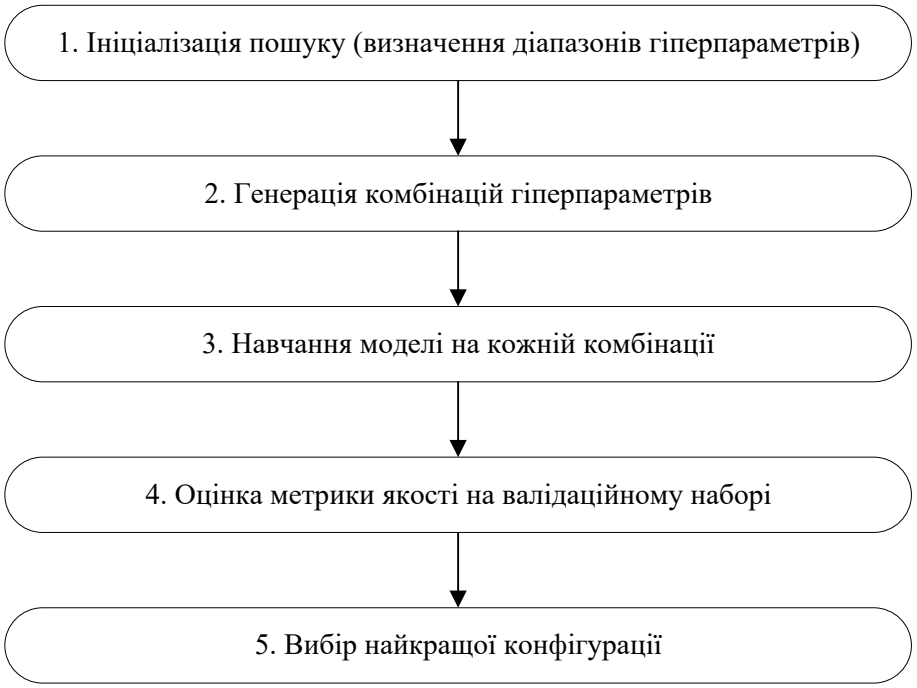

Рис. 3.9. Процес автоматизованого налаштування гіперпараметрів

Для ефективного дослідження простору гіперпараметрів використовуємо такі алгоритми автоматизованого пошуку, в залежності від поставленої мети і особливостей даних навантаження на вебсервер (рис. 3.9):

– Grid Search – повний перебір усіх можливих комбінацій гіперпараметрів при невеликій кількості параметрів.

– Random Search – випадковий вибір комбінацій з простору параметрів, часто забезпечує кращі результати з меншими витратами часу, ніж Grid Search.

– Bayesian Optimization (з бібліотекою Optuna) – моделює функцію втрат як стохастичний процес і адаптивно вибирає наступні точки на основі попередніх результатів. Особливо ефективна при складних та дорогих обчисленнях.

Формальним вираженням метафункції оптимізації буде мінімізація помилки прогнозу:

$$\Theta^* = arg \min_{\theta} L(y, \hat{y}^{(\theta)}), \tag{3.109}$$

де θ– вектор гіперпараметрів, $\hat{y}^{(\theta)}$ – прогноз моделі з параметрами θ, y – справжні значення, L – функція втрат (RMSE, MAE, R^2).

Інші альтернативні метрики оцінки якості моделі L для часових рядів: симетризована відносна похибка (SMAPE, Symmetric Mean Absolute Percentage Error):

$$SMAPE = \frac{100\%}{N} \sum_{t=1}^{N} \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \tag{3.110}$$

і середня абсолютна масштабована похибка (MASE, Mean Absolute Scaled Error):

$$MASE = \frac{\frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t|}{\frac{1}{N-1}\sum_{t=2}^{N}|y_t - y_{t-1}|}. \tag{3.111}$$

Вихідними даними $D_{out} = \{\Theta^*, \Theta_i \longrightarrow M_i, plot(\Theta_i, M_i)\}$ будуть ефективна конфігурація гіперпараметрів Θ^* моделі для подальшого прогнозування, таблиця відповідності гіперпараметрів $\Theta_i = \{\theta_{1i}, \theta_{2i}, \dots, \theta_{ni}\}$ і метрик точності $M_i = \{MAE_i, MAPE_i, RMSE_i, {R^2}_i\}$, візуалізація залежностей між гіперпараметрами та точністю.

Етап 6. Оцінка якості на різних наборах даних

В якості вхідних даних $D_{in} = \{S_{day}, S_{night}, S_{holiday}\}$ розглянемо множину тестових сценаріїв $S = \{S_{day}, S_{night}, S_{holiday}\}$, які відображають різні типи сценаріїв у реальному середовищі експлуатації вебсервера: робочі години S_{day} – висока інтенсивність запитів, типовий офісний трафік; нічний трафік S_{night} –

низьке навантаження, потенційно високий рівень шуму в даних; святкові дні $S_{holiday}$ – нерегулярні, сплескові або відсутні запити, непередбачувані шаблони.

На даному етапі для оцінювання ефективності моделі прогнозування вебнавантаження використовується множинна перевірка метрик (RMSE, SMAPE, R^2 тощо).

Для візуалізації розподілу помилок $(y_t - \hat{y}_t)$ у кожному наборі даних створюються гістограми. Це дозволяє: ідентифікувати зміщення моделі (систематичні відхилення), виявити "довгі хвости" у розподілі помилок (великі винятки), порівняти розсіяння похибок між піднаборами.

Модель також тестується на довші горизонти прогнозування $(t + 10, t + 20)$, щоб перевірити стійкість при зростанні відставання і визначити як погіршуються метрики зі збільшенням горизонту, чи є зсуви або тенденції до пере- чи недооцінювання.

Вихідними даними $D_{out} = \{S_i \longrightarrow M_i, Score\}$ будуть таблиця оцінок M_i для кожного набору S_i (табл. 3.8), узагальнені висновки $Score = f(Var(R^2{}_i), max(SMAPE_i))$ про здатність моделі працювати на гетерогенних даних, де потрібне додаткове донавчання чи тонке донавчання (fine-tuning), чи можлива адаптація до контексту (наприклад, вбудовані маркери робочих/вихідних днів).

Таблиця 3.8

Оцінки метрик для кожного типу даних:

Набір	RMSE	SMAPE (%)	R^2
Робочі години	12.5	7.3	0.91
Нічний трафік	4.2	15.7	0.68
Святкові дні	18.6	22.4	0.52

Етап 7. Візуалізація і генерація звіту

В якості вхідних даних $D_{in} = \{\hat{Y}_{WNN}, \hat{Y}_{LSTM}, T, M\}$ розглянемо результати

прогнозування для моделей (WNN та LSTM), часові мітки прогнозу T, метрики точності прогнозу $M = \{MAE, MAPE, SMAPE, RMSE, R^2\}$.

Даний етап призначений для ефективного аналізу результатів виконаного прогнозування навантаження на вебсервер. Тому ми будуємо лінійні графіки трафіку, щоб показати зміни трафіку (як реальні, так і прогнозовані значення) з часом. Генеруємо Boxplot помилок, щоб оцінити розподіл помилок між різними прогнозами, чи є певні аномалії чи сильні коливання в помилках прогнозування. Створюємо теплові карти для візуалізації кореляцій між різними змінними (наприклад, між прогнозами і реальними значеннями), виявлення патернів і зв'язків у даних.

Вихідними даними $D_{out} = \{R, V, D\}$ будуть автоматично згенерований звіт R у форматі PDF/HTML, що містить опис моделей, набір графіків V (лінійні графіки, heatmap, гістограми похибок), інтерактивна інформаційна панель-дашборд D для візуалізації результатів з використанням бібліотек Dash, Streamlit, Plotly. Де обирається діапазон дат $[t_i, t_j]$ для аналізу, порівнюються різні моделі та їх результати, відбувається фільтрація за метриками.

Етап 8. Інтеграція в систему моніторингу

В якості вхідних даних $D_{in} = \{M^*, A_{API}(t), U(t)\}$ розглянемо вибрану ефективну модель M^* для прогнозування вебнавантаження, API даних у момент часу t із реального вебсервера $A_{API}(t) = \{Q(t), V(t), \tau(t), \rho(t)\}$, які будуть використовуватися для моніторингу та прогнозування (запити $Q(t)$, обсяг трафіку $V(t)$, час відповіді $\tau(t)$, використання ресурсів $\rho(t) = \{CPU, RAM, I/O\}$ тощо), вектор $U(t)$ актуальних значень метрик моніторингу.

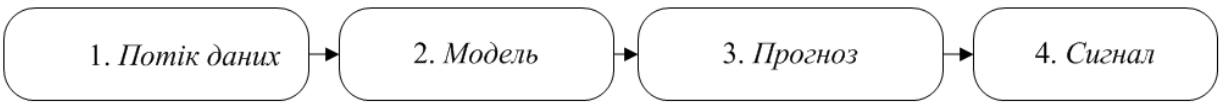

Рис. 3.10. Схема роботи модуля прогнозування

На даному підсумковому етапі реалізовується модуль прогнозування через

розгортання REST API для доступу до моделі прогнозування. Це дозволить іншій частині системи (наприклад, моніторинговій системі) звертатися до моделі для отримання прогнозів (рис. 3.10).

Таблиця 3.9

Структура етапів методу прогнозування навантаження на вебсервер

№	Назва етапу	Вхідні дані D_{in}	Вихідні дані D_{out}
1	Оцінка стійкості до шуму та зміни даних	$\{w_m, \omega, X_{train}, \mu_{sys}\}$	$\{X, \chi(w_m)\}$
2	Формування множини вхідних параметрів	$\{X, Tr(X), A_J(t), D_j(t)\}$	$\{\boldsymbol{X}\}$
3	Вибір моделі та моделювання за допомогою алгоритмів машинного навчання	$\{\boldsymbol{X}_t, y_t, y_{t+k}, k\}$	$\{\hat{f}(\boldsymbol{X}_t \mid \theta), \theta, L(\theta)\}$
4	Прогнозування навантаження вебсерверу	$\{Z_t\}$	$\{\hat{Y}_t, plot(y_t, \hat{y}_t)\}$
5	Автоматизоване налаштування гіперпараметрів	$\{\Theta, \boldsymbol{X}, \boldsymbol{y}\}$	$\{\Theta^*, \Theta_i \longrightarrow M_i, plot(\Theta_i, M_i)\}$
6	Оцінка якості на різних наборах даних	$\{S_{day}, S_{night}, S_{holiday}\}$	$\{S_i \longrightarrow M_i, Score\}$
7	Візуалізація і генерація звіту	$\{\hat{Y}_{WNN}, \hat{Y}_{LSTM}, T, M\}$	$\{R, V, D\}$
8	Інтеграція в систему моніторингу	$\{M^*, A_{API}(t), U(t)\}$	$\{\hat{Y}_{\text{time}}, F, E_{alert}\}$

Модуль приймає дані з вебсервера через HTTP запит, робить прогноз і повертає результат у вигляді JSON. Після отримання прогнозу система порівнює його з певним порогом і сповіщає адміністратора або систему балансування навантаження, якщо значення перевищує встановлений ліміт. Пороги можна

налаштувати вручну або автоматизувати їх визначення на основі попередніх даних.

Дані, що надходять від вебсервера (кількість запитів, CPU usage, тощо), передаються до API моделі. Модель отримує дані і на основі них робить прогноз і генерує інші метрики. Якщо прогнозоване навантаження перевищує поріг, система генерує сигнал або сповіщення.

Вихідними даними $D_{out} = \{\hat{Y}_{\text{time}}, F, E_{alert}\}$ підсумкового етапу будуть прогноз найближчого навантаження $\hat{Y}_{\text{time}}$, автоматична фільтрація та передобробка потоку навантаження F, система сповіщень про потенційні аномалії або перевищення критичних навантажень E_{alert}.

Модуль прогнозування може бути інтегрований у продакшн середовище, де він може працювати в режимі близькому до реального часу, отримуючи дані від вебсервера та надаючи прогнози для моніторингу.

Для наочності у таблиці 3.9 представлено структурований короткий опис всіх етапів реалізації методу прогнозування навантаження на вебсервер.

Процес автоматизованого підбору оптимального вейвлету у розробленій системі відбувається поетапно та базується на кількісному порівнянні ефективності різних базисних функцій за визначеними критеріями. Спочатку виконується вейвлет-декомпозиція вихідного ряду навантаження для кількох типів вейвлетів (наприклад, Добеші, Коіфлет, Біортогональний, Симлет). Для кожного з них обчислюються метрики якості. Далі система здійснює багатокритеріальний аналіз із використанням вагових коефіцієнтів критеріїв $r_1 - r_{10}$. Кожен вейвлет оцінюється за цими критеріями, після чого формується інтегральний показник ефективності. На основі цього показника система автоматично обирає найкращий вейвлет, тобто той, що забезпечує мінімальну похибку прогнозування при збереженні стабільності декомпозиції. Такий підхід дозволяє уникнути ручного підбору та багатогодинних експериментів: алгоритм самостійно перевіряє кілька найбільш перспективних вейвлетів, оцінює результати за формалізованими критеріями і видає оптимальний варіант для подальшого використання у WNN-моделі.

Висновки до розділу 3

У розділі було поетапно реалізовано модель прогнозування навантаження на вебсервер, яка охоплює збір логів та метрик, їх попередню обробку, вейвлет-декомпозицію, моделювання та подальшу інтеграцію результатів у робоче середовище. Такий підхід дозволяє ефективно розділяти сигнал на трендові та коливальні компоненти, що підвищує точність прогнозування як довготривалих змін, так і короткочасних пікових навантажень.

Модель ІМППНВ являє собою каскадний процес, де на етапі попередньої обробки формуються лагові вектори, далі застосовується вейвлет-декомпозиція, після чого окремо прогнозуються трендова та високочастотні складові. Тренд обчислюється за допомогою Prophet, що добре моделює сезонність та довготривалі зміни. Для високочастотних коливань використовується LSTM або TDNN, здатні враховувати складні часові залежності. На виході результати об’єднуються та інтегруються в інформаційну систему.

Застосування вейвлет-аналізу дало змогу покращити роботу з нестаціонарними часовими рядами, адже він забезпечує високу деталізацію у часово-частотному просторі. Поєднання статистичних методів із глибинними нейронними моделями дозволило отримати більш збалансовані результати та знизити похибку прогнозів.

Практична цінність моделі полягає у можливості завчасно виявляти перевантаження та запобігати їм завдяки автоматичному масштабуванню, оптимізації використання обчислювальних ресурсів та інтеграції з інструментами моніторингу. Розроблена модель може застосовуватись адміністраторами вебресурсів як у тестових середовищах для генерації синтетичних навантажень, так і у реальних умовах для підтримання стабільності роботи серверів, здатних підлаштовуватися під динаміку трафіку, у використанні більших наборів даних для навчання та в аналізі зовнішніх факторів (сезонність, рекламні кампанії, події). Це відкриває перспективи для ще більш точного прогнозування та гнучкого управління вебсерверами у змінному середовищі.

На основі інтегральної моделі було розроблено метод визначення

найефективнішого типу материнського вейвлету, який забезпечує об’єктивний вибір базисної функції для аналізу часових рядів вебнавантаження. Метод базується на багатокритеріальному підході, що враховує такі властивості вейвлетів, як ортогональність, компактність, регулярність, асиметричність і ступінь геометричної подібності до характеру змін вебтрафіку. Такий підхід дозволив підвищити точність відновлення сигналу після декомпозиції та мінімізувати втрату суттєвих динамічних компонентів у процесі попередньої обробки даних.

На основі результатів вибору оптимального базису також розроблено метод прогнозування навантаження на вебсервер, який поєднує вейвлет-декомпозицію з нейромережевим прогнозуванням. Застосування обраного материнського вейвлету забезпечує ефективне виділення трендових і високочастотних компонентів трафіку, що, у свою чергу, підвищує адаптивність моделі до змін середовища та знижує середньоквадратичну похибку прогнозу. Розроблений метод може бути використаний як основа для побудови інтелектуальних систем керування навантаженням у вебсерверах.

РОЗДІЛ 4. ОСОБЛИВОСТІ СИСТЕМИ ПРОГНОЗУВАННЯ НАВАНТАЖЕННЯ НА ВЕБСЕРВЕР ЗА ДОПОМОГОЮ ВЕЙВЛЕТ-ПЕРЕТВОРЕНЬ

4.1. Архітектура системи прогнозування навантаження на вебсервер

Відповідно до сформованих методів у попередньому розділі, першим кроком у створенні системи прогнозування навантаження на вебсервер у рамках запропонованої моделі стала розробка актуальної архітектури інформаційної системи, спрямованої на підвищення ефективності прогнозування навантаження на вебсервер за допомогою вейвлет-перетворень.

Під час побудови моделі було застосовано зібраний датасет (Додаток А), що містить інформацію про навантаження на вебсервер за кілька місяців. Зібрані дані включали кількість запитів до сервера за одиницю часу, час обробки запитів, а також інші параметри, які впливають на рівень навантаження. Для проведення аналізу дані були очищені від аномалій та шуму, а також нормалізовані для забезпечення коректності подальшої обробки.

Одним із ключових аспектів при створенні вейвлет-моделі стало визначення відповідної вейвлет-функції, оскільки вибір базису безпосередньо впливає на якість розкладання сигналу та точність подальшого прогнозування. Тип вейвлету визначається характером часових рядів, зокрема ступенем їх гладкості, наявністю шумів, різких стрибків або періодичних компонент.

Так, для сигналів із різкими змінами або імпульсами, наприклад при різкому зростанні навантаження на вебсервер у пікові години, ефективними є короткі, компактно підтримувані вейвлети, такі як Haar (db1) або Daubechies 2–4 (db2–db4). Вони добре фіксують локальні стрибки та дискретні зміни в часових рядах.

Натомість для плавних або квазісинусоїдальних сигналів, що відображають поступові зміни навантаження, більш доцільним є використання вейвлетів із вищим порядком регулярності, наприклад Coiflet (coif3–coif5) або Symlet (sym5–sym8). Ці вейвлети мають більшу кількість моментів і забезпечують кращу гладкість апроксимації, що дозволяє точніше вловлювати повільні тенденції.

У випадках, коли необхідно зберегти симетрію сигналу та уникнути фазових зсувів, доцільно застосовувати біортогональні вейвлети (bior3.5, bior4.4), які мають пари прямого та оберненого перетворення з однаковою часовою локалізацією. Це особливо важливо для коректного відновлення прогнозованих часових рядів після зворотного DWT.

Рис. 4.1. Архітектура інтегральної моделі

Архітектура інтегральної моделі прогнозування навантаження на вебсервер (рис. 4.1) побудована за принципом багаторівневої обробки даних, що поєднує методи сигналової аналітики та алгоритми глибокого навчання. Основна ідея

полягає у створенні гібридної обчислювальної системи, де кожен рівень відповідає за окрему функціональну роль – від попереднього фільтрування даних до високорівневого прогнозування динаміки трафіку. Така архітектура забезпечує адаптивність до змін у структурі вхідних сигналів і масштабованість при збільшенні обсягу даних або кількості серверів.

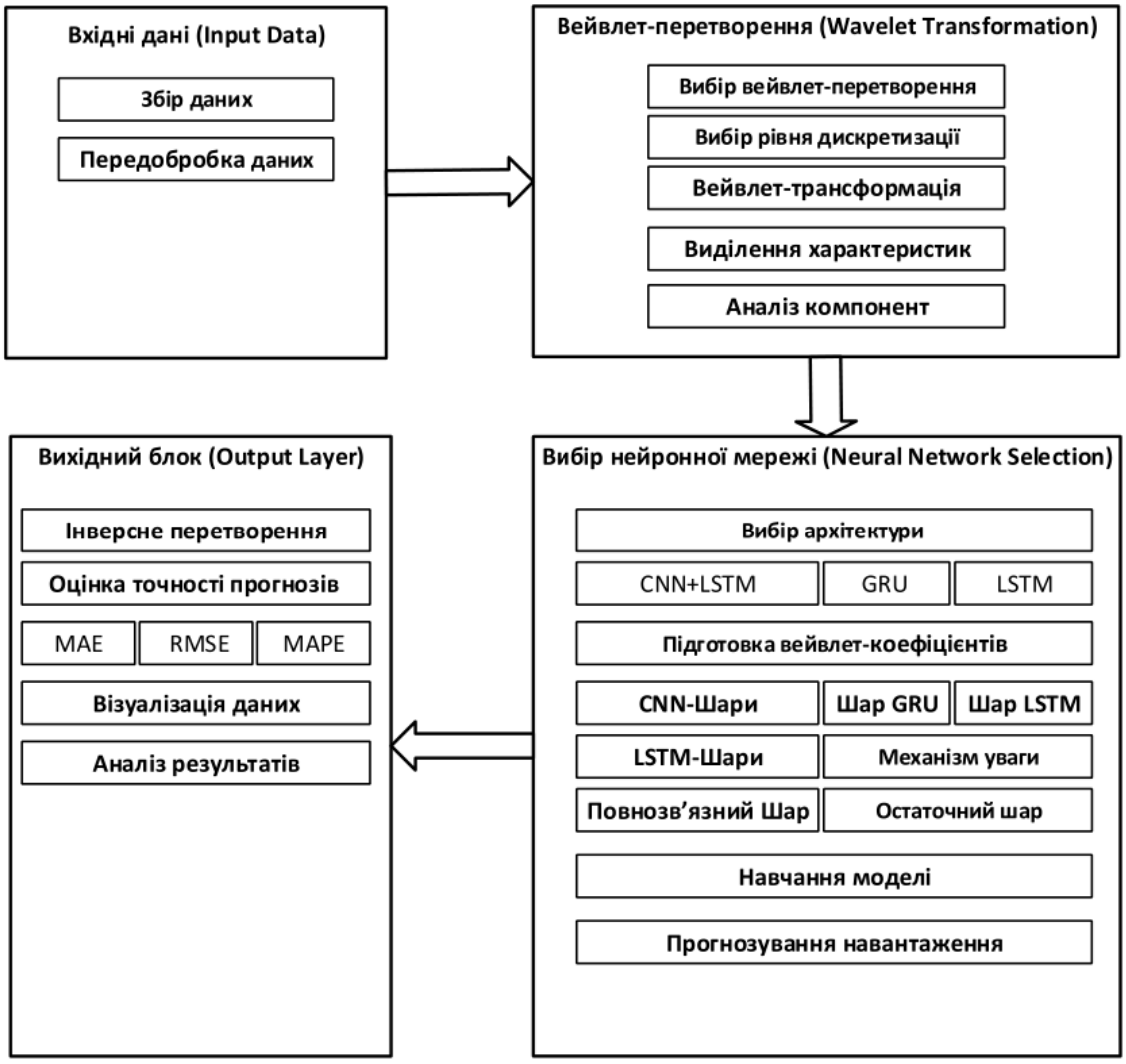

Рис. 4.2. Структура побудови інтегральної моделі прогнозування навантаження

На нижньому рівні моделі було реалізовано попередню обробку та вейвлет-декомпозицію, яка виконує розкладання часових рядів на підсигнали різної частотної складності. Це дозволяє ізолювати короткочасні коливання від довготривалих тенденцій та підвищити інформативність подальшого аналізу. Для цього застосовується дискретне вейвлет-перетворення à trous, що забезпечує

збереження часової роздільності без втрати даних. Отримані вейвлет-коефіцієнти утворюють набір характеристик, який надходить до нейронного рівня обробки.

Середній рівень архітектури складається з нейронних мереж, зокрема WNN, яка здійснює нелінійне згладжування і виділення латентних закономірностей між частотними компонентами. Цей рівень відіграє роль проміжного перетворювача, що адаптивно поєднує результати сигналової обробки з ознаками, релевантними для прогнозування. Далі інформація передається до рекурентної нейронної мережі типу LSTM, яка відповідає за часовий аналіз і побудову коротко-, середньо- та довгострокових прогнозів навантаження.

Верхній рівень архітектури забезпечує інтеграцію прогнозів і контроль якості результатів. Тут здійснюється об’єднання вихідних даних з декількох моделей, нормалізація значень і корекція помилок на основі метрик, таких як RMSE чи MAPE. Крім того, реалізовано модуль адаптивного масштабування для керування параметрами, який дозволяє моделі автоматично перебудовуватись при зміні характеристик мережного середовища або появі нових типів запитів.

Інтегрована модель для прогнозування навантаження на вебсервери включає використання як вейвлет-перетворення, так і нейронних мереж (наприклад, Prophet, LSTM із механізмом уваги, або гібридна WNN+LSTM). Побудова цієї моделі охоплює такі основні етапи (рис. 4.2):

– Збір даних: отримання часових рядів, що відображають навантаження на сервер (кількість запитів, використання процесора, пам’яті тощо) у різні проміжки часу.

– Передобробка: стандартизація та нормалізація, видалення шуму, формування навчальних і тестових вибірок. Нормалізація у межах [0, 1] підвищує стабільність моделі, особливо при використанні вейвлет-аналізу.

– Вибір вейвлету: можливість використання різних типів дискретного вейвлет-перетворення (DWT), таких як db4, haar, coif, sym, із різною кількістю рівнів декомпозиції.

– Вейвлет-декомпозиція: розкладання вхідного часового ряду на кілька рівнів за допомогою обраної вейвлет-функції (наприклад, Daubechies 4).

– Виділення характеристик: отримані вейвлет-коефіцієнти використовуються для формування ознак сигналу, що враховують як часові, так і частотні властивості.

– Аналіз компонент: окремі частоти сигналу аналізуються з метою виявлення прихованих трендів і закономірностей. Потім дані конкатенуються для подання на вхід нейронної мережі.

– Вибір нейронної архітектури: забезпечується можливість вибору відповідної моделі згідно зі специфікою задачі.

– Підготовка даних: вейвлет-декомпозиція змінює кількість ознак і часових кроків, що враховується при формуванні вхідного масиву для нейромережі.

– Формування моделі: налаштування архітектури згідно з обраною моделлю та конфігурацією її шарів.

– Навчання: використання вейвлет-коефіцієнтів як ознак для навчання нейромережі з оптимізатором Adam і функцією втрат MSE.

– Прогнозування: модель формує прогноз навантаження на основі попередніх значень та вейвлет-перетворених даних. Результати подаються на вихідний шар для отримання передбачення наступного навантаження.

– Зворотне перетворення: відновлення часового ряду на основі передбачених вейвлет-коефіцієнтів із використанням оберненого DWT (iDWT).

– Оцінка якості: застосування метрик MAE, RMSE та MAPE для перевірки точності моделі.

– Візуалізація: графіки оригінального і відновленого ряду, вейвлет-коефіцієнтів на різних рівнях – допомагають оцінити здатність моделі точно відтворювати динаміку даних.

– Аналіз результатів: виявлення тенденцій, аномалій та потенційних слабких місць моделі для її подальшого удосконалення.

Адаптивність інтегральної моделі досягається завдяки поєднанню методів багатомасштабного вейвлет-аналізу, що забезпечує динамічне виділення локальних і глобальних компонент навантаження, з глибинними нейронними структурами, здатними коригувати власні параметри на основі поточного

контексту вхідних даних. Такий підхід дозволяє системі ефективно реагувати на зміни інтенсивності запитів, періодичність трафіку та появу аномальних подій. Масштабованість моделі реалізується за рахунок модульної архітектури багатоступеневого конвеєра, у якому окремі етапи – фільтрація, нормалізація та прогнозування – можуть виконуватись паралельно або розподілятись між кількома обчислювальними вузлами. Це забезпечує стабільність роботи системи при збільшенні обсягів даних і кількості джерел навантаження, зберігаючи прийнятний рівень обчислювальної ефективності.

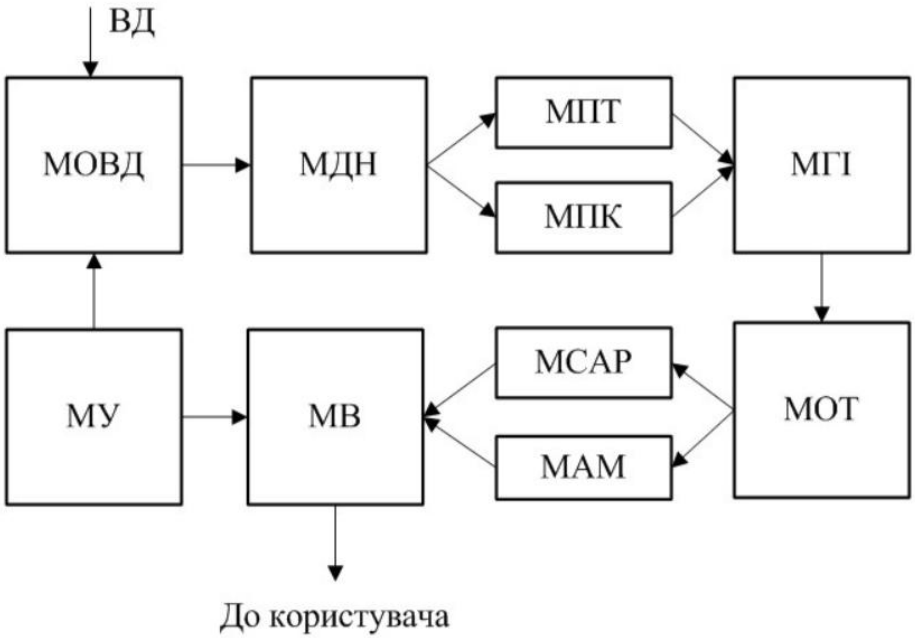

Рис. 4.3. Модулі процесів прогнозування навантаження на вебсервер

Система прогнозування структурно складається із 9 модулів та окремого модулю управління (МУ). Архітектура модулів показана на рис. 4.3.

У системах прогнозування навантаження на вебсервер особливе значення має модульна архітектура, що забезпечує поетапну обробку, аналіз і використання часових рядів. Такий підхід дозволяє досягти високої гнучкості, масштабованості та точності моделей, які застосовуються для передбачення зміни навантаження з метою запобігання перевантаженням і забезпечення стабільної роботи інфраструктури.

1. Модуль обробки вхідних даних (МОВД) є критично важливим компонентом, що відповідає за первинну підготовку інформації для подальшого аналізу. Основним його завданням є агрегування, фільтрація та стандартизація потоків вхідних даних (ВД), отриманих із систем вебсерверного моніторингу. Особливу увагу приділено усуненню шумів та аномальних значень, а також перетворенню часових міток до регулярного формату з можливістю створення лагових векторів, що дозволяє враховувати ефекти часових затримок. У процесі реалізації використовується багатоступенева конвеєрна обробка, яка включає: агрегацію даних із кількох джерел моніторингу вебсерверів (наприклад, лог-файли вебсервера, системи метрик Prometheus тощо), фільтрацію шумів за допомогою ковзного середнього та медіанного фільтра для усунення одиничних викидів, виявлення аномалій через Z-нормування, стандартизацію часових міток – перетворення часових рядів до рівномірної дискретизації (наприклад, із кроком у 1 хвилину чи 5 секунд), що забезпечує коректність подальшого вейвлет-перетворення, формування лагових векторів для збереження часової пам’яті процесу перед подачею даних у нейронну мережу. Таким чином, МОВД формує якісний та узгоджений набір вхідних ознак, який є базовим для коректної роботи наступних модулів.

2. Модуль декомпозиції навантаження (МДН) виконує трансформацію вхідного часового ряду в спектральну область за допомогою дискретного вейвлет-перетворення. Реалізація здійснюється за допомогою бібліотеки PyWavelets та власної функції на базі алгоритму à trous, який зберігає довжину сигналу без редукції розмірності. В результаті формується два типи компонентів: апроксимаційна частина $A(t)$, яка відображає довгострокові тренди, та набір детальних складових $D_j(t)$, що містять високочастотні флуктуації. Така декомпозиція дозволяє здійснювати окремий аналіз кожного рівня масштабування, що критично важливо в умовах складної структури вебнавантаження. Кожен рівень декомпозиції може аналізуватися окремо: $A(t)$ використовується для моделювання довгострокових тенденцій (плавних змін у добовому чи тижневому циклі), $D_1(t) - D_2(t)$ застосовуються для виявлення короткострокових піків або збоїв системи. Для

мінімізації втрат інформації після реконструкції застосовується обернене перетворення (iDWT) із перевіркою похибки за метриками MSE та енергетичного балансу сигналу.

3. Модуль прогнозування тренду (МПТ) орієнтований на екстраполяцію довгострокових компонент вебнавантаження, що були виділені у вигляді апроксимаційної частини $A(t)$. Для цього використовується модель Prophet, яка ефективно обробляє дані з наявними сезонними складовими (добовими, тижневими або місячними патернами). У процесі реалізації модель автоматично виконує: виявлення сезонності та фітинг компонентів (trend + seasonality + holiday effects), регуляризацію параметрів λ_1, λ_2 для уникнення перенавчання, адаптацію до зовнішніх регресорів (показників CPU, I/O-завантаження тощо). Отриманий прогноз $A^*(t)$ потім передається у модуль гібридної інтеграції, де інтегрується з короткостроковими передбаченнями з нейромережевих моделей (WNN/LSTM).

4. Модуль прогнозування коливань (МПК) відповідає за моделювання короткочасних флуктуацій і пікових навантажень, характерних для високочастотних компонент сигналу $D_j(t)$, отриманих після вейвлет-декомпозиції. Основна ідея реалізації полягає у застосуванні глибоких рекурентних архітектур, здатних зберігати часові залежності між подіями. У цьому модулі реалізовано двошаровий підхід: глибока нейронна мережа LSTM (Long Short-Term Memory) – використовується для моделювання нелінійних часових залежностей, яка завдяки механізму «forget» і «input»-вентилів адаптивно реагує на зміну характеру навантаження, зберігаючи релевантний контекст минулих станів, темпоральна згорткова мережа TDNN (Temporal Delay Neural Network) – використовується для обробки локальних закономірностей у часових відрізках. Згорткові шари дозволяють виявляти короткострокові патерни у вхідному потоці (наприклад, сплески активності в певні хвилини чи секунди). Результатом роботи МПК є набір прогнозованих детальних компонент $D^*_j(t)$, які відображають мікродинаміку системи. Їх подальша агрегація з довгостроковими тенденціями здійснюється у модулі гібридної інтеграції.

5. Модуль гібридної інтеграції (МГІ) є центральним компонентом

інтегральної архітектури, який поєднує результати довгострокового прогнозу $A^*(t)$ (отриманого з МПТ) та короткострокових коливань $D^*_j(t)$ (отриманих з МПК) у єдиний вихідний ряд. З технічного погляду, реалізація МГІ базується на: адаптивному зважуванні компонентів відповідно до їх енергетичного внеску у вихідний сигнал (коефіцієнти ваги α_j визначаються на основі відношення дисперсій або через оптимізацію функції втрат), зворотному вейвлет-перетворенні (iDWT) для реконструкції сигналу у часовій області з використанням прогнозованих коефіцієнтів $A^*(t), D^*_j(t)$, постобробці результату – згладженні та нормалізації відновленого ряду для уникнення артефактів, спричинених різними масштабами моделювання. Таким чином, МГІ реалізує принцип багатомасштабного моделювання, забезпечуючи баланс між точністю довгострокових прогнозів і чутливістю до короткотермінових змін. Це дозволяє інтегральній моделі адаптуватися до різних режимів навантаження: стабільного, пікового та деградаційного тощо.

6. Модуль оцінки точності (МОТ) реалізує механізм безперервного контролю якості прогнозування на основі статистичних та машинних метрик. На етапі тестування і під час роботи моделі він виконує: обчислення показників точності, таких як RMSE, MAE, MAPE, R^2 – для оцінки середньої похибки та адекватності відтворення тренду; крос-валідацію з ковзним часовим вікном (TimeSeriesSplit) для перевірки стійкості моделі до зсувів у часових даних; побудову контрольних діаграм похибок і сигналізацію відхилень, коли поточна похибка перевищує встановлений поріг ε_{max}; автоматичну переоцінку параметрів (retraining trigger) у разі систематичного погіршення якості прогнозу. У перспективі цей модуль може бути розширений функціоналом self-learning feedback [], який дозволить моделі самостійно оновлювати вагові коефіцієнти на основі накопичених похибок.

7. Модуль сповіщень та автоматичної реакції (МСАР) виконує функції інтелектуального моніторингу та раннього попередження про наближення критичних станів системи. Його ключова мета – зменшити час реакції адміністратора або системи оркестрації на аномальні зміни навантаження. Реалізація базується на двох підсистемах: система тригерів аналізує результати

прогнозів та поточні метрики (CPU, RAM, RPS) в процесі експлуатації. Якщо прогнозоване навантаження перевищує граничний поріг T_{alert}, модуль формує подію типу warning, critical або emergency; інтеграційна підсистема повідомлень забезпечує взаємодію з зовнішніми комунікаційними сервісами: Telegram API, Grafana Alerts тощо. Повідомлення надсилаються у форматі, що включає час, тип події, прогнозоване значення навантаження та рекомендовану дію (наприклад, "збільшити кількість інстансів на 2"). Модуль може працювати у реактивному або проактивному режимі: у першому випадку він лише інформує користувача, у другому – ініціює виклик API.

8. Модуль автоматичного масштабування (МАМ) є ключовим елементом у забезпеченні самоадаптивності системи. Його функція полягає у прогнозуванні майбутнього використання ресурсів (CPU, RAM, I/O) та кількості HTTP-запитів на одиницю часу (RPS) для прийняття рішення щодо розширення або скорочення обчислювальних ресурсів. У реалізації МАМ використовується двоступеневий механізм: прогнозування навантаження на основі результатів інтегральної моделі (МГІ), де дані про очікуваний RPS передаються у підмодуль управління ресурсами; прийняття рішення про масштабування реалізується через логіку правил або порогів. Наприклад: якщо прогнозоване навантаження > 80% потужності системи → додати інстанси; якщо < 30% → вивільнити ресурси. МАМ може взаємодіяти з AWS Auto Scaling Group або Docker Compose API, використовуючи REST-запити. Таким чином, система не лише прогнозує навантаження, а й автоматично реагує на нього, мінімізуючи ризики деградації продуктивності.

9. Модуль візуалізації (МВ) відповідає за графічне представлення всіх етапів роботи системи – від вхідних даних до прогнозованих значень. Основним завданням є забезпечення інформаційної наочності та зручності аналізу результатів для оператора або аналітика. Модуль реалізується двома основними способами: інтеграція з Grafana для побудови дашбордів, де візуалізуються такі параметри, як CPU, RAM, мережевий трафік, RPS, прогнозоване навантаження, а також індикатори точності (RMSE, MAPE); використання бібліотек Plotly / Dash (Python) для побудови інтерактивних графіків із можливістю масштабування, вибору

часових діапазонів і накладання прогнозів на історичні дані. Завдяки цьому модулю користувач отримує цілісне уявлення про динаміку вебнавантаження: історію за останні 24 години або тиждень, а також прогноз на найближчу годину чи добу. Крім того, MB підтримує кольорове маркування аномалій та візуальні сигнали попередження, синхронізовані з MCAP.

Загалом останні три модулі формують верхній рівень інтеграційного шару системи, який забезпечує автономність, реактивність і зручність взаємодії користувача з аналітичним ядром. Вони реалізують принцип замкненого циклу керування – від збору даних і прогнозування до сповіщення й автоматичної дії, що є важливою рисою інтелектуальних систем моніторингу вебнавантаження.

4.2. Експериментальна установка

Експериментальна установка являє собою апаратно-програмний комплекс, призначений для дослідження ефективності інтегральної моделі прогнозування навантаження на вебсервер, що базується на на запропонованих моделях. Система також підтримує автоматизоване налаштування гіперпараметрів моделей із застосуванням методів оптимізації, таких як Grid Search, Random Search та Bayesian Optimization.

Для реалізації моделі прогнозування вебнавантаження необхідне відповідне апаратне забезпечення. Система повинна мати процесор не нижче Intel Core i7 або AMD Ryzen 7, оперативну пам’ять обсягом 16–32 ГБ, а також графічний прискорювач класу NVIDIA RTX 3060 або 3080 для прискорення глибинного навчання. Накопичувач типу SSD з обсягом щонайменше 512 ГБ забезпечує швидкий доступ до логів та результатів обробки. Операційна система – Ubuntu 20.04 або Windows 10, з встановленими бібліотеками Python, серед яких TensorFlow, PyTorch, PyWavelets, Optuna та Scikit-learn.

Програмна частина проєкту охоплює кілька ключових етапів. Спершу здійснюється інтеграція з логами вебсерверів (наприклад, Apache або Nginx), проводиться попередня обробка: нормалізація, видалення шуму, побудова лагів. Далі сигнал обробляється через вейвлет-декомпозицію за алгоритмом à trous з

використанням базисних функцій (db4, bior3.3), що дає змогу виділити апроксимаційні та деталізовані компоненти.

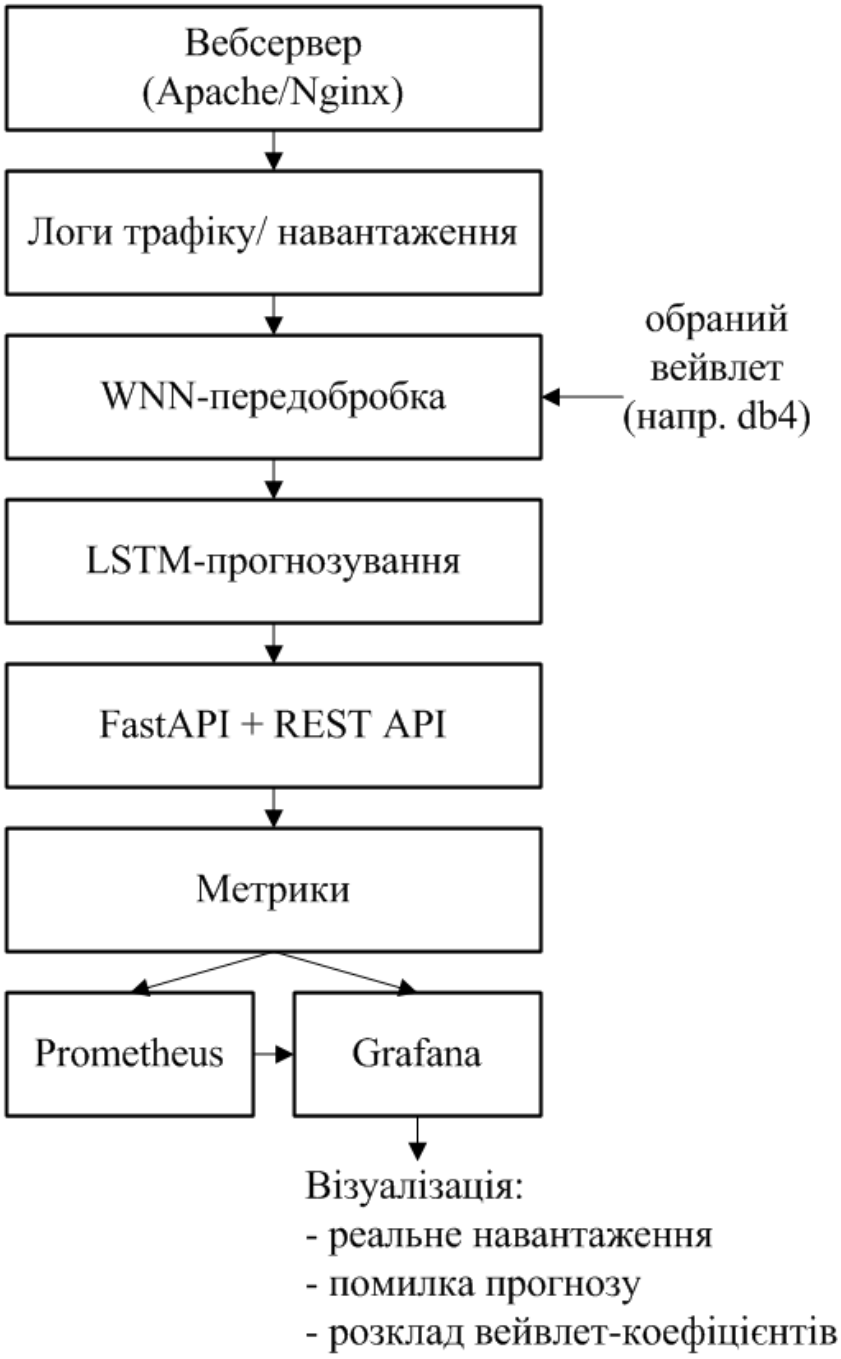

Рис. 4.4. Схема функціонування експериментальної установки

На етапі прогнозування застосовується багатошарова LSTM-мережа з кількістю нейронів 128–512, використанням dropout, нормалізації та активаційних

функцій ReLU/tanh. Автоматичне налаштування гіперпараметрів виконується за допомогою фреймворку Optuna, орієнтуючись на мінімізацію RMSE та MAE на валідаційних вибірках (рис. 4.4).

Після побудови моделі здійснюється її оцінка на тестовому наборі логів, з візуалізацією результатів у вигляді графіків (фактичне vs прогнозоване навантаження) та формуванням звіту у HTML. Для інтеграції в реальне середовище розроблено REST API, що дозволяє подавати нові дані до моделі.

Додатково, передбачено підключення до систем моніторингу, таких як Grafana і Prometheus, що забезпечують зручний візуальний контроль за станом серверного навантаження (рис. 4.5).

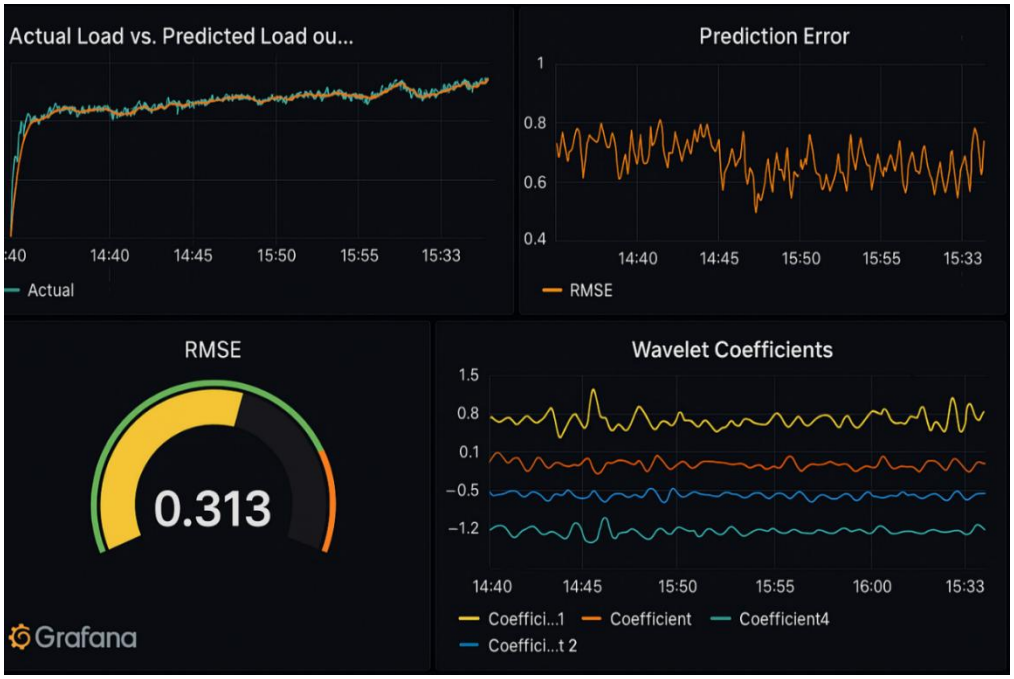

Рис. 4.5. Інтерактивний дашборд Grafana

Ключовим елементом експериментальної установки є створений у середовищі Python програмний застосунок IWMLP (Integrated Wavelet Model for Web Traffic Load Prediction) [112], що реалізує методи та алгоритми, описані у другому та третьому розділах. Частину коду цього застосунку подано в додатках. Інтерфейс головного вікна програми IWMLP зображено на рис. 4.6.

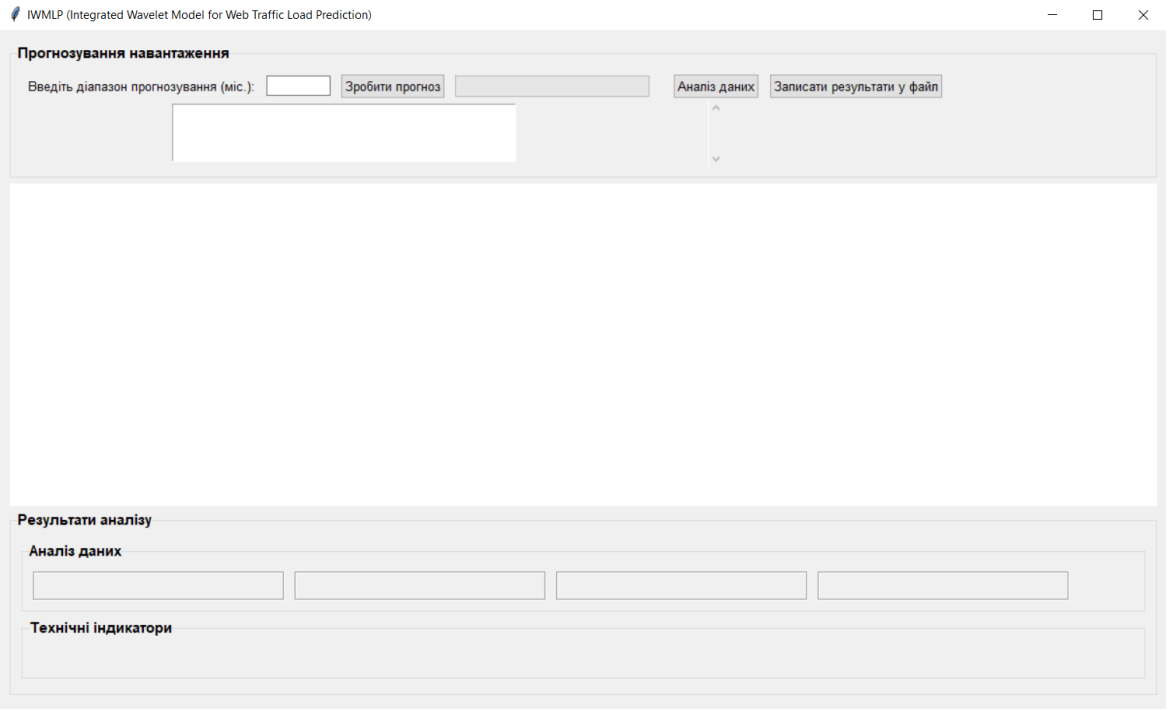

Рис. 4.6. Головне вікно програмного додатку IWMLP

Центральну частину інтерфейсу займає графік навантаження, який демонструє динаміку історичних та прогнозованих даних.

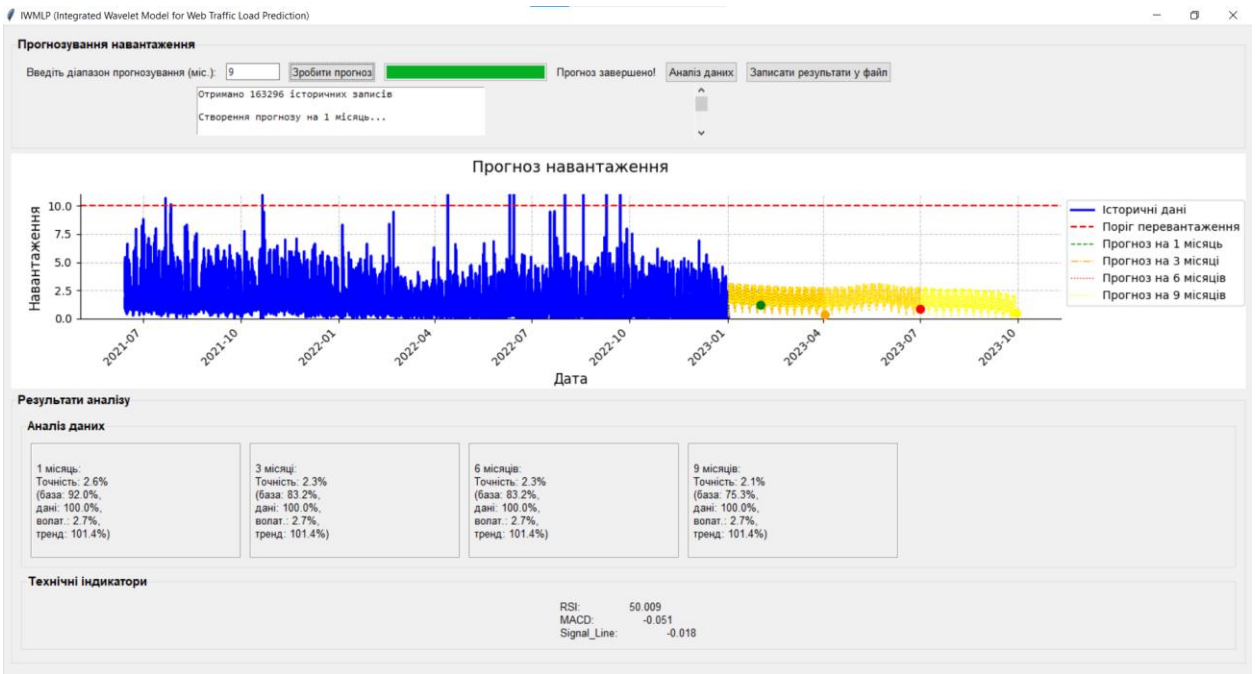

Рис. 4.7. Результат прогнозування навантаження у вікні додатку IWMLP

Насиченим синім кольором представлено фактичні дані, тоді як прогнозні

значення виділено у вигляді у вигляді різнокольорових пунктирних ламаних, які відповідають за різні періоди прогнозування. Чітко позначений поріг перевантаження червоною пунктирною лінією допомагає виявляти критичні періоди (рис. 4.7). У нижньому блоці інтерфейсу розміщено аналітичні результати, які поділено на два сектори: якісний аналіз даних та технічні індикатори для відповідного горизонту прогнозування (1, 3 або 6 місяці тощо).

Таблиця 4.1

Ключові показники системи прогнозування навантаження

Категорія	Показник	Як порахувати в Python
Точність	MAE (Mean Absolute Error)	mean_absolute_error()
	RMSE (Root Mean Squared Error)	np.sqrt(MSE)
	MAPE (%)	np.mean(np.abs((y - ŷ)/y)) * 100
	R^2 (R-squared)	r2_score()
Benchmarking	Час виконання	time.time() або perf_counter()
	Використання пам’яті	tracemalloc
	Кількість параметрів моделі	model.count_params() (у Keras)
	Швидкість прогнозу (samples/sec)	час / кількість
Overfitting	Показник переобучення	train_loss - val_loss
	Learning Curve	matplotlib, plt.plot()
Часовий аналіз	Періодичність	scipy.signal, statsmodels, Fourier
	Параметр Херста (Hurst exponent)	from hurst import compute_Hc
	Фрактальна розмірність	nolds.dfa()

У процесі побудови, тестування та впровадження моделей прогнозування навантаження вебсервера важливим аспектом є багатовимірна оцінка їх ефективності. Це забезпечує комплексний аналіз моделі не лише як математичного об’єкта, а й як інструмента, придатного до практичного застосування у

високонавантажених середовищах (табл. 4.1).

До основної категорії визначення точності прогнозу належать статистичні метрики, що дозволяють безпосередньо оцінити відхилення між прогнозованими та фактичними значеннями. Серед них MAE (3.32) – показник середньої абсолютної похибки, що є інтерпретованим у термінах реальних одиниць вимірювання; RMSE (3.33) – який дозволяє інтерпретувати результат у тих самих одиницях, що й вхідний сигнал; MAPE (3.34) – відносна похибка, представлена у відсотковій формі; SMAPE (Symmetric MAPE) – модифікована версія MAPE, що коректно обробляє нульові значення, забезпечуючи симетричність у випадку зміни напрямку помилки тощо.

Наступний блок стійкості та якості прогнозування включає метрики, що характеризують здатність моделі відтворювати внутрішню структуру даних: R^2 (коефіцієнт детермінації) – частка дисперсії вихідної змінної, поясненої моделлю; Adjusted R^2 – уточнене значення R^2 з урахуванням кількості незалежних змінних; Variance Explained – показник загальної здатності моделі описувати варіації у часовому ряді тощо.

Оскільки вебнавантаження часто характеризується зміною сезонності, трендів і локальних збурень, важливо враховувати такі адаптивні показники: час адаптації моделі – індикатор швидкості підлаштування до нових патернів у даних; вплив вікон згладжування – ефект параметрів згладжування у вейвлет-аналізі на чутливість моделі до локальних змін.

Оцінка обчислювальної ефективності моделі є критично важливою для розгортання моделей у реальному середовищі, особливо в edge-системах, як перспективний напрямок подальшого розвитку: час навчання – тривалість побудови моделі; час виконання прогнозу – середній час отримання одного прогнозного значення; споживання оперативної пам’яті (RAM) – характеристика, важлива для систем з обмеженими ресурсами; завантаження CPU/GPU – рівень використання обчислювальних потужностей, що може впливати на масштабованість.

Модель повинна враховувати специфіку навантаження вебсервера вебнавантаження. Де в якості ключових параметрів можна врахувати: кількість HTTP-запитів – основний індикатор навантаження; середній розмір відповіді –

впливає на пропускну здатність; кількість помилок 4xx/5xx – сигналізує про потенційні перевантаження; час відповіді (Latency) – важливий критерій якості обслуговування; кількість активних з'єднань – поточне навантаження в системі; наявність піків – важливо для виявлення стресових ситуацій.

Для обґрунтованого вибору моделі був проведений порівняльний аналіз методів: Benchmarking – співставлення з класичними методами (ARIMA, LSTM, Prophet тощо); Overfitting Indicator – оцінка різниці між помилками на тренувальних і тестових даних.

Критерії для MCDM-аналізу є специфічними для систем, що використовують гібридний підхід з використанням вейвлет-перетворень: ортогональність-асиметричність – базові властивості вейвлет-функцій; компактність – здатність забезпечити локалізацію в часі; геометрична подібність – ступінь відповідності форми вейвлета характеру сигналу; масштабованість – інваріантність до зміни масштабу вхідного сигналу; час обчислення вейвлет-перетворення – впливає на загальну продуктивність тощо.

4.3. Експериментальні дослідження

Основною метою експериментальних досліджень було підтвердження достовірності ключових результатів, отриманих у межах дисертаційного проєкту. З цією метою дослідження були розподілені на три послідовні етапи. Перший етап полягав у перевірці гіпотези, що розроблений метод дозволяє досягти високої ефективності прогнозування навантаження на вебсервер. Під ефективністю розуміємо дотримання допустимого рівня похибки, оптимальні часові та ресурсні витрати на навчання моделі, а також прийнятний термін формування навчальної вибірки. З практичних міркувань, цей термін становить близько одного року. Для реалізації задачі використовувався персональний комп'ютер із параметрами, описаними у пункті 4.2.

На другому етапі запропонований підхід було реалізовано для аналізу часових рядів, які містять усереднені значення вебтрафіку з інтервалом у п'ять хвилин. Для цього було зібрано лог-файли з однієї з регіональних аварійно-

диспетчерських служб України. Отримані дані дали змогу створити деталізовану модель локального вебнавантаження, а також виявити типові патерни в роботі вебсервера протягом усього року. Вхідні дані представляли собою точкові вимірювання обсягу трафіку в певні моменти часу, що дозволило проводити аналіз на рівні високої точності.

На третьому етапі процесу прогнозування було обрано вейвлети сімейства Daubechies, зокрема db4, для опрацювання нерівномірних коливань у часових рядах. Такий вибір обумовлений низкою переваг: покращеною локалізацією у часі та частоті (що дає змогу виявляти як короткотермінові пікові навантаження, так і довгострокові тренди), високою стійкістю до шумів, характерних для реальних даних, достатньою гладкістю функції (що запобігає появі артефактів при реконструкції сигналу), а також компактною областю підтримки, яка забезпечує швидке обчислення та ефективну обробку великих обсягів інформації.

У даному дослідженні запропоновано метод прогнозування навантаження на вебсервер, що поєднує вейвлет-перетворення з сучасними архітектурами штучних нейронних мереж. Зокрема, розглянуто гібридну модель WNN+LSTM, у якій вейвлет-декомпозиція слугує етапом попередньої обробки даних. Це дозволяє виокремити ключові тренди та приглушити вплив шумів, характерних для часових рядів. Така комбінація поєднує сильні сторони обох методів: WNN добре справляється з локальними структурними особливостями, тоді як LSTM ефективно моделює довготривалі залежності у послідовностях.

Інші типи нейронних мереж, що були досліджені, включають Prophet та LSTM з механізмом уваги. Ці два поширені види нейронних мереж застосовуються для аналізу часових рядів, оскільки вирішують проблему "затухання градієнту", характерну для класичних нейронних мереж.

Інтеграція механізму уваги до LSTM відкриває нові можливості для цієї архітектури, підвищуючи її здатність зосереджуватись на ключових частинах часового ряду, схоже на те, як це роблять трансформери. Механізм уваги дозволяє мережі фокусуватись на важливих сегментах вхідного часового ряду, не обмежуючись суто послідовною природою нейронної мережі.

Додавання механізму уваги до LSTM значно покращує здатність моделі концентрувати увагу на найважливіших частинах часового ряду при генерації вихідних даних. Це є особливо корисним у задачах, де контекст із певних частин часового ряду має більше значення, ніж з інших. Ключові компоненти Prophet або LSTM з механізмом уваги включають стандартні елементи, а також додатковий механізм уваги, що обчислює контекстний вектор – зважену суму прихованих станів всього часового ряду.

Оцінки уваги E_i визначають, наскільки важливим є кожен прихований стан для поточного стану за допомогою мультиплікативної уваги (Luong attention). Мультиплікативна увага використовує скалярний добуток для обчислення подібності між прихованими станами енкодера та декодера, що дає більш ефективний обчислювальний підхід:

$$E_i = h^T W h_i, \tag{4.1}$$

де h – поточний прихований стан декодера, h_i – прихований стан енкодера на i-му кроці, W – вагова матриця, яка контролює взаємодію між прихованими станами.

Оцінки уваги перетворюються у ваги уваги через softmax-функцію, яка нормалізує ці значення, щоб їхня сума дорівнювала 1. Це дозволяє трактувати ваги уваги як ймовірності того, наскільки важливий кожен елемент послідовності для поточного виходу:

$$A_i = \frac{e^{E_i}}{\sum_j e^{E_j}}. \tag{4.2}$$

Контекстний вектор обчислюється як зважена сума прихованих станів вхідної послідовності за допомогою вагів уваги та комбінується з поточним прихованим станом (GRU або LSTM) для генерації остаточного результату:

$$V = \sum_i A_i h_i. \tag{4.3}$$

Таким чином, механізм уваги дозволяє моделі зосереджуватись на найбільш релевантних частинах часового ряду, що підвищує якість обробки даних у задачах з довгими часовими рядами, таких як аналіз навантаження на вебсервер.

Для інтеграції вейвлет-перетворення в обробку часових рядів, необхідно спочатку виконати дискретне вейвлет-перетворення (DWT) на вхідних даних, а потім використовувати отримані коефіцієнти як вхід до нейронної мережі. Така реалізація моделі (табл. 4.2) має такі ключові елементи:

– Підготовка вейвлет-трансформованих даних: після вейвлет-перетворення кількість часових кроків змінюється, і розмір нового тренувального масиву використовує ці нові часові кроки.

– Шар Prophet (LSTM) використовується для обробки вейвлет-трансформованих даних з додаванням ознак.

– Механізм уваги забезпечує створення контекстного вектора, що дозволяє моделі зосереджуватися на важливих частинах перетвореного часового ряду.

– Остаточний шар визначає прогноз на основі виходів нейронної мережі та механізму уваги.

Завдяки додатковим налаштуванням можна змінювати тип вейвлета та рівень деталізації вейвлет-перетворення відповідно до специфіки даних, налаштовуючи параметри wavelet та level.

Цей підхід поєднує переваги вейвлет-перетворення для аналізу частотних компонентів часового ряду з можливостями нейронних мереж для виявлення часових залежностей у даних.

Для оцінки точності прогнозів використовуються метрики, такі як середня абсолютна похибка (MAE), середньоквадратична помилка (RMSE) та середнє абсолютне відсоткове відхилення (MAPE). Ці показники дозволяють кількісно оцінити точність прогнозу та порівняти результати з іншими методами.

Аналіз результатів експериментів показує, що розроблена інтегрована модель покращує традиційні методи прогнозування. Проведені випробування на реальних логах вебнавантаження (зібраних протягом двох тижнів із вебсервера освітньої платформи) показали, що середня абсолютна похибка (MAE) моделі знизилась на 12–15 % порівняно з базовою LSTM-моделлю, а коефіцієнт детермінації R^2 збільшився з 0.87 до 0.93. Це свідчить про те, що вейвлет-фільтрація ефективно усуває високочастотний шум і виділяє релевантні тренди у вхідних

даних, що дозволяє нейронній мережі точніше вловлювати закономірності зміни навантаження. Отже, підвищення точності прогнозу як у оперативному (1 година), так і в короткостроковому (до 24 годин) періоді свідчить про ефективність запропонованої гібридної архітектури, яка поєднує дискретне вейвлет-перетворення та LSTM-рівень прогнозування.

Таблиця 4.2

Параметри моделі

Вейвлет-функція	Daubechies 4 (db4)
Кількість рівнів деталізації	3-5 рівнів
Модель прогнозування	LSTM-мережа з кількома шарами
Кількість шарів	2-3 шари
Кількість нейронів на шар	50-100 нейронів на шар
Функція активації	ReLU або tanh
Оптимізатор	Adam або SGD (Stochastic Gradient Descent)
Розмір пакета	32-64
Кількість епох	50-100
Метрики оцінки	MAE, RMSE, MAPE

Таблиця 4.3

Метрики похибок застосування інтегральної моделі

Архітектура	RMSE	MAE	MAPE
ARIMA	1.0514	0.7990	0.4878
CNN+LSTM	0.0867	0.0606	0.3133
GRU+Attention	0.0620	0.0484	0.2297
LSTM+Attention	0.0602	0.0435	0.2235

Застосування інтегрованого підходу також дозволило знизити помилки прогнозування та краще адаптуватися до змін у структурі трафіку. Наприклад, середньоквадратична помилка виявилася на 15–25% меншою порівняно з моделлю

ARIMA (див. табл. 4.3), що свідчить про вищу точність і ефективність запропонованого методу.

Результати, представлені на рис. 4.8, підтверджують доцільність застосування вейвлет-перетворення у задачах прогнозування навантаження на вебсервер. Це видно з того, що: вихідний часовий ряд з високочастотними коливаннями (синя лінія) після вейвлет-фільтрації (червона лінія) стає більш “згладженим”, при цьому зберігаються основні тренди; прогноз моделі на основі вейвлет-коефіцієнтів точніше повторює реальні дані, про що свідчить менша розбіжність між фактичним і прогнозованим рядом на графіку; порівняння з моделлю без вейвлет-перетворення показує, що помилки прогнозу (MAE, RMSE) дійсно зменшуються, особливо у періоди пікових навантажень. На рис. 4.8 час вимірюється в секундах, а рівень вебтрафіку – в мегабайтах.

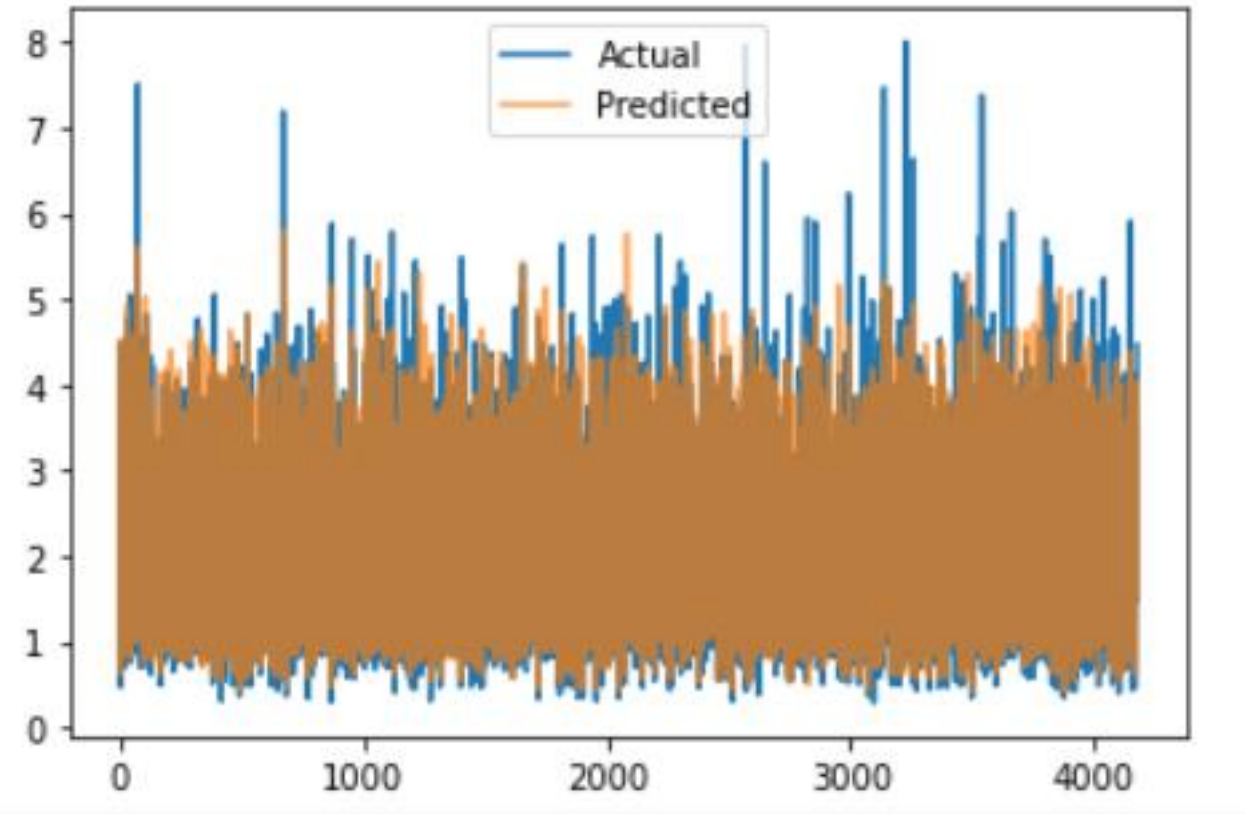

Рис. 4.8. Порівняння фактичних і прогнозованих даних

Водночас слід враховувати певні обмеження цієї моделі. Зокрема, вейвлет-перетворення можуть бути досить ресурсоємними з точки зору обчислень,

особливо при роботі з великими обсягами даних. Це може потребувати значних обчислювальних потужностей і тривалого часу на обробку. Крім того, для побудови якісної прогнозної моделі необхідна велика кількість навчальних даних, що не завжди є доступним у практичних умовах.

Перспективними напрямами подальших досліджень є вдосконалення алгоритмів вейвлет-перетворення з метою зменшення їхньої обчислювальної складності, а також вивчення впливу зовнішніх факторів, таких як сезонність, час доби, масові події або рекламні кампанії. Окрему увагу слід приділити розробці гібридних моделей, які поєднують вейвлет-аналіз із іншими сучасними підходами до прогнозування, що дозволить досягти ще вищої точності результатів.

Побудова інтегральної моделі тісно пов’язана з множиною параметрів, які визначають рівень навантаження на вебсервер. Враховуючи характер задач, що виконуються в системах ВЗП, а також особливості застосовуваних мережевих протоколів, усі параметри доцільно згрупувати у три категорії:

– ресурси апаратного забезпечення сервера;

– ресурси операційної системи;

– мережеві ресурси.

Для побудови довгострокових прогнозів важливо дослідити природу подій, які впливають на навантаження, а також їхній можливий зв’язок із іншими, більш передбачуваними явищами. Наприклад, при аналізі роботи ВЗП необхідно враховувати залежність трафіку від звичних ритмів життя користувачів – добових, тижневих, сезонних чи річних циклів. Це вимагає наявності історичних даних спостережень щонайменше за один повний рік. Виявлення закономірностей та повторюваних періодів є ключем до точного прогнозування, адже навантаження на вебсервер розглядається як функція часу.

Практичний досвід та результати аналізу свідчать про те, що ефективність функціонування корпоративних інформаційних систем значною мірою залежить від продуктивності вебсервера, який обробляє запити віддалених користувачів. Вебсервер повинен мати достатні обчислювальні ресурси для обслуговування запитів. У разі перевантаження він не зможе забезпечити необхідну якість

обслуговування, тоді як надмірне резервування ресурсів призводить до нераціонального використання коштів.

Також було встановлено, що наявні технології прогнозування навантаження вебсерверів не повністю відповідають сучасним вимогам. Основна причина – недостатня точність прогнозів, яка обумовлена обмеженістю математичних моделей, що не дозволяють ефективно адаптуватися до характерних змін параметрів навантаження в динаміці.

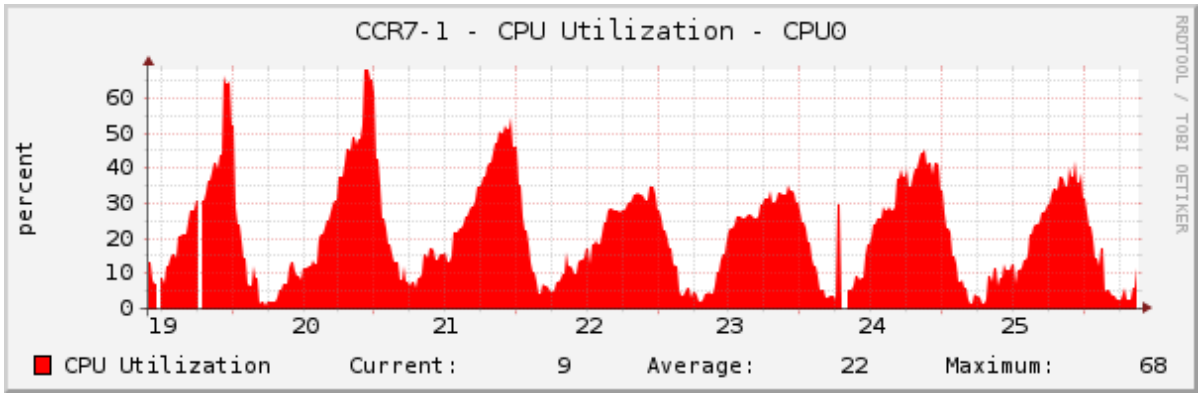

Рис. 4.9. Рівень навантаженості на вебсервер протягом тижня

У спрощеному вигляді можна припустити, що сервер витрачає приблизно однакову кількість ресурсів на обслуговування кожного запиту. Отже, загальне навантаження в значній мірі залежить від кількості HTTP-запитів і встановлених HTTP-з’єднань (рис. 4.9). Водночас ці дії можна уточнити, враховуючи специфіку протоколу TCP/IP, на основі якого функціонує HTTP.

Таким чином, для оцінки навантаження на вебсервер доцільно аналізувати параметри, які відображають обсяги трафіку та кількість з’єднань, здійснених за допомогою протоколів TCP/IP та HTTP [57].

Варто зазначити, що практичні спостереження підтверджують складний характер змін цих параметрів. Робочі режими вебсервера охоплюють як стаціонарні, так і нестаціонарні стани, при цьому функціональні характеристики мають багатоперіодичну природу. Нестаціонарність проявляється, зокрема, у зміні моментів появи або зникнення певних періодичних компонент, що впливають на навантаження.

Для апробації запропонованої моделі прогнозування в межах розробленої інформаційної системи було використано два різних вебсервери за 12 місяців звітнього періоду. Перший – інформаційний сервер із тематичними новинами (далі – вебсервер-1), другий – сервер, що обслуговує міську аварійно-диспетчерську службу (вебсервер-2). У таблицях 4.4 та 4.5 наведено усереднені та повні щомісячні показники роботи вебсервера-1 і вебсервера-2 відповідно за період з липня 2018 по червень 2019 року.

Таблиця 4.4

Щомісячна статистика вебсервера-1 за звітній період

Місяць	В середньому за день				Всього за місяць					
	запитів	файлів	сторінок	відвідувань	сайтів	Мбайт	відвідувань (тис.)	сторінок (тис.)	файлів (тис.)	запитів (тис.)
07	713	664	291	33	246	574	0,4	4,3	9,9	10,7
08	623	604	275	35	378	543	1,1	8,5	18,7	19,3
09	1300	1281	489	163	1526	1653	4,9	14,6	38,4	39,0
10	1176	1162	443	214	1832	1480	6,6	13,7	36,0	36,4
11	1123	1110	332	162	1670	1406	4,8	9,9	33,3	33,7
12	1870	1848	518	220	2543	2641	6,8	16,0	57,3	57,9
01	1522	1505	462	200	2372	1976	6,2	14,3	46,6	47,2
02	1943	1925	512	227	2824	2372	6,3	14,3	53,9	54,4
03	1685	1668	508	251	2545	2248	7,8	15,7	51,7	52,2
04	1761	1732	548	274	2861	2231	8,2	16,4	51,9	52,8
05	1642	1623	558	291	2895	1723	9,0	17,3	50,3	50,9
06	1713	1630	527	280	2432	1918	7,8	14,7	45,6	47,9
Σ						20771	70,3	160,3	494,0	502,8

Таблиця 4.5

Щомісячна статистика вебсервера-2 за звітній період

Місяць	В середньому за день				Всього за місяць					
	запитів (тис.)	файлів (тис.)	сторінок (тис.)	відвідувань	сайтів	Мбайт	відвідувань (тис.)	сторінок (тис.)	файлів (тис.)	запитів (тис.)
07	244	244	242	427	116	12967	13	7507	7561	7566
08	321	321	319	535	89	15844	16	9879	9940	9947
09	395	394	392	624	107	18299	18	11768	11835	11847
10	364	364	361	873	65	32149	27	11190	11276	11286
11	353	353	350	1396	82	51045	41	10513	10587	10599
12	356	356	353	1307	68	49458	40	10948	11033	11045
01	340	340	338	1426	89	54901	44	10487	10548	10557
02	328	328	326	1418	70	56125	39	9132	9187	9194
03	254	254	253	1182	119	48593	36	7835	7883	7892
04	231	231	230	1093	89	46359	32	6881	6921	6930
05	214	214	213	891	100	47650	27	6595	6631	6641
06	252	252	250	2013	68	48389	56	7003	7046	7064
Σ						481779	395	109743	110452	110574

Обробка експериментальних даних здійснювалась з використанням таких програмних засобів: MATLAB 2019b (зокрема, Wavelet Toolbox), MathCad 14.0, Microsoft Excel з пакету MS Office, а також бібліотеки PyWavelets для мови програмування Python.

Як видно з рис. 4.10, для вебсервера-1 характерна чітка періодичність з тижневими й сезонними коливаннями вебтрафіку, що пов’язано з особливостями його контенту та специфікою функціонування. На рис. 4.10 час визначається в

днях, а рівень вебтрафіку – в кілобайтах. Графік кількості запитів демонструє складну часову структуру, зокрема нерегулярні сплески навантаження. Такий тип трафіку характерний для вебресурсів, що обслуговують неоднорідну користувацьку активність, наприклад, у проміжках між робочими та вихідними днями або під час рекламних кампаній.

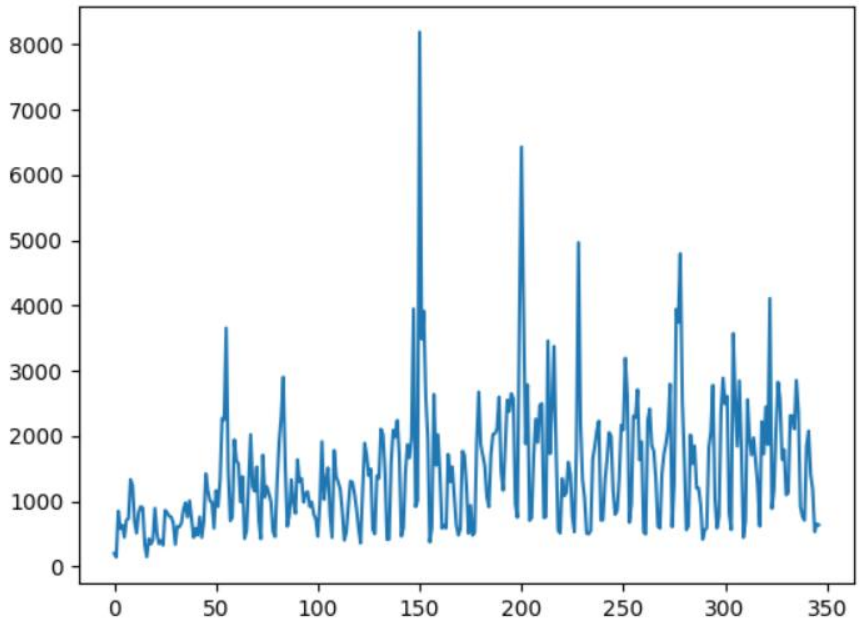

Рис. 4.10. Графік рівня вебтрафіку вебсервера-1

На рис. 4.11, 4.13 зображено результати вейвлет-аналізу рівня вебтрафіку вебсерверів у вигляді скалограм (трьохвимірне представлення значень вейвлет-коефіцієнтів) за допомогою вейвлету типу French Hat (Bumb wavelet), що має гарну частотну локалізацію, забезпечує виявлення як короткочасних імпульсів у трафіку, так і довготривалих трендів. Це дає змогу розкласти сигнал на набір детальних та апроксимаційних компонент, кожна з яких відображає поведінку трафіку на окремому часовому масштабі. Також, це дозволяє не лише краще зрозуміти структуру запитів, а й ідентифікувати потенційні аномалії чи підозрілі шаблони, що можуть вказувати на атаки або збої.

На скалограмах рис. 4.11, 4.13 видно різні рівні сигналу та структури, що дозволяють ефективно виділяти тренди і флуктуації трафіку. Кольори на

скалограмі відповідають різним рівням детальних і апроксимованих коефіцієнтів вейвлету, а одиниці виміру представлені у форматі інтенсивності або нормалізованого трафіку, що дозволяє наочно оцінити змінність та піки навантаження для подальшого аналізу та прогнозування. Темніші тони відповідають сильним коливанням сигналу на відповідному масштабі, зелені тони – менш інтенсивні коливання з середньою амплітудою, світлі кольори вказують на низьку інтенсивність або відсутність значущих коливань на даному рівні. По горизонтальній осі часу відкладені умовні одиниці часу, що відповідають відцентрованому часовому ряду.

Отримані вейвлет-коефіцієнти зберігають більшість енергетично значущих компонентів сигналу, що важливо для подальшого прогнозування. Саме на основі цієї декомпозиції у моделі можна сформувати більш стійкі до шуму ознаки, які враховують як миттєві зміни, так і повільні коливання в навантаженні вебсервера.

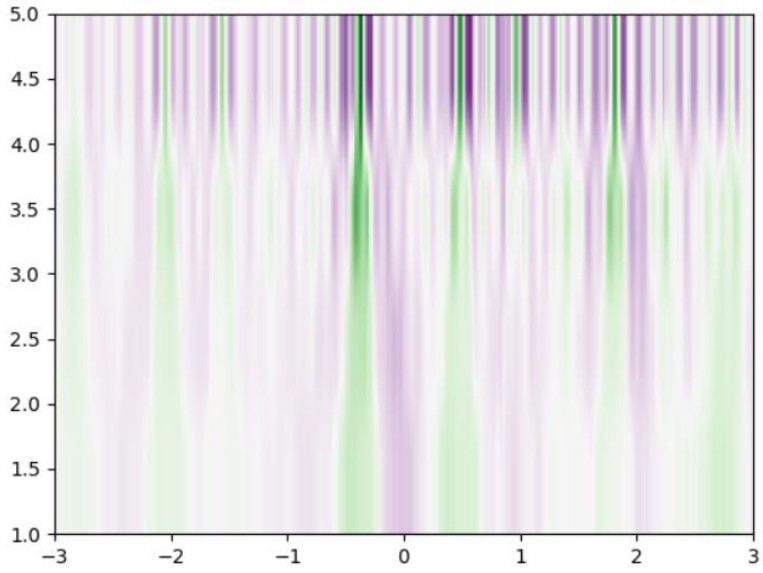

Рис. 4.11. Вейвлет-аналіз рівня вебтрафіку вебсервера-1

Для вебсервера-2 також, як видно з графіка навантаженості та візуалізації вейвлет-аналізу на рис. 4.12 – 4.13, простежується тижневі та добові коливання, але характер навантаження має значний рівень зашумленості. На рис. 4.12 час вимірюється в днях, а рівень вебтрафіку – в мегабайтах.

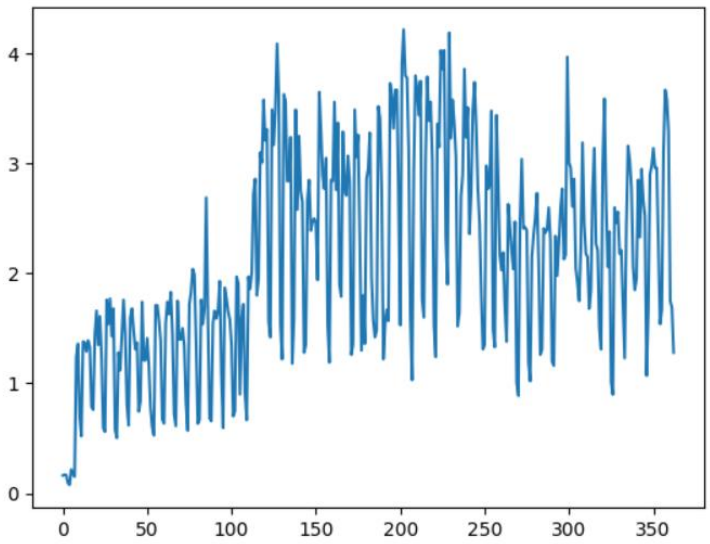

Рис. 4.12. Графік рівня вебтрафіку вебсервера-2 (MB)

Також видно, що з плином часу середній рівень навантаження на вебсервер значно зростає.

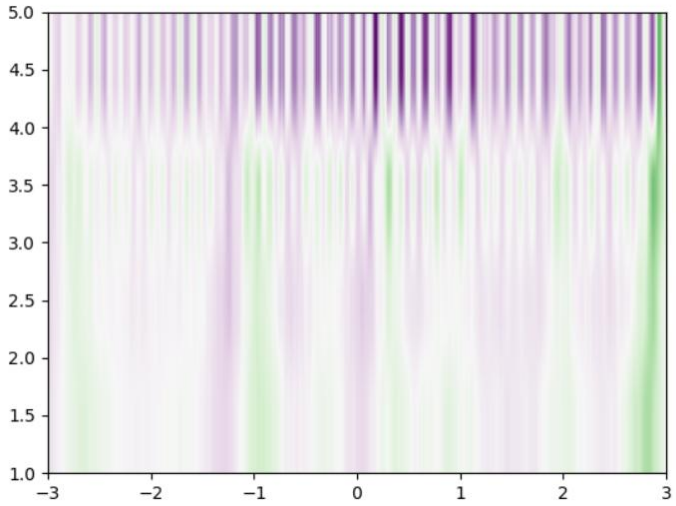

Рис. 4.13. Вейвлет-аналіз рівня вебтрафіку вебсервера-2

Практичний досвід і отримані результати свідчать про те, що ефективність більшості вітчизняних корпоративних інформаційних систем безпосередньо залежить від продуктивності вебсерверу, який забезпечує обробку запитів віддалених користувачів. Очевидно, що обчислювальні ресурси вебсерверу повинні відповідати рівню його завантаження. Інакше вебсервер або не зможе повноцінно задовольняти потреби користувачів інформаційної системи, або

надмірне резервування потужностей призведе до економічних та енергетичних втрат.

Також було виявлено, що існуючі інформаційні технології прогнозування навантаження вебсерверу не відповідають сучасним вимогам. Недостатня точність прогнозів здебільшого пов’язана з недосконалістю наявних методів та моделей прогнозування, що базуються на спрощених математичних підходах і не здатні адаптуватися до змінних параметрів навантаження.

Вибір факторів для прогнозування є окремим завданням, яке потребує попередньої оцінки впливу кожного з них до початку процесу прогнозування.

Аналіз даних вказує на наявність періодичних компонентів, таких як добові, тижневі та сезонні коливання в досліджуваному процесі. Прогнозування навантаження здійснювалось для різних часових інтервалів, які були класифіковані таким чином: оперативний прогноз – в межах поточної години; короткостроковий прогноз – до 1 доби; середньостроковий прогноз – до тижня; довгостроковий прогноз – до 1 місяця, трендовий прогноз – більше 1 місяця (згідно табл. 2.6).

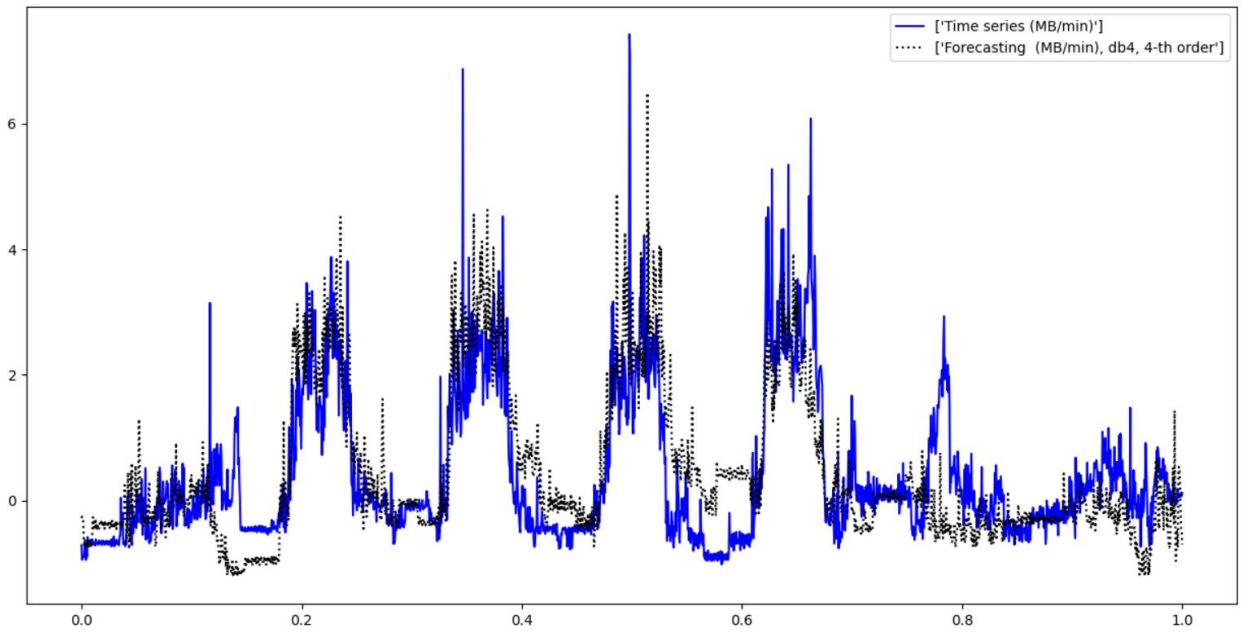

Рис. 4.14. Прогнозування тижневих навантажень на вебсервер

На рис. 4.14 показані результати прогнозування середньострокових (тижневих) навантажень на вебсервер, а на рис. 4.15 – візуалізація цих самих

вейвлет-перетворень у вигляді скалограми за допомогою комплексного вейвлету Морле. При аналізі екстрапольованого часового ряду середньоквадратична помилка апроксимації становила 3 – 8%.

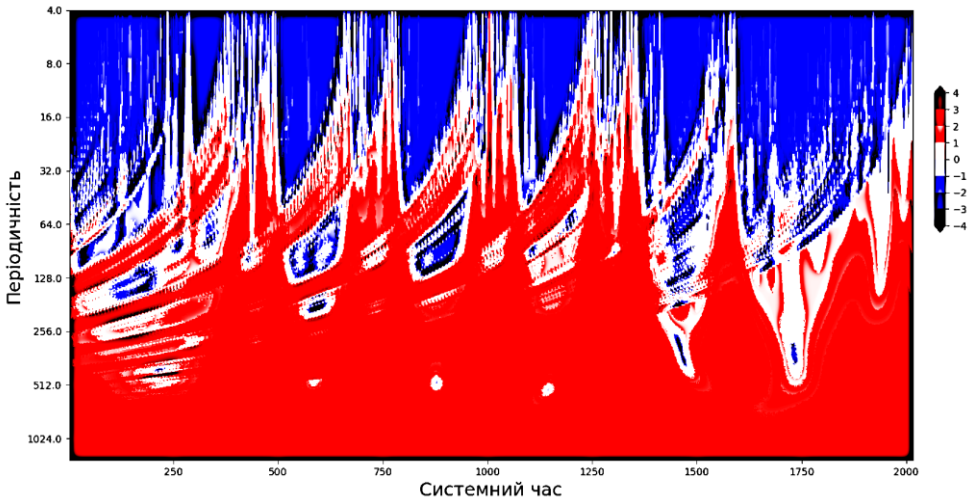

Рис. 4.15. Скалограма вейвлет-перетворень прогнозованих тижневих навантажень

На рис. 4.16 видно результати прогнозування довгострокових (місячних) навантажень на досліджуваний вебсервер, без урахування аномальних викидів трафіку, причому середньоквадратична помилка апроксимації становила 5 – 12%.

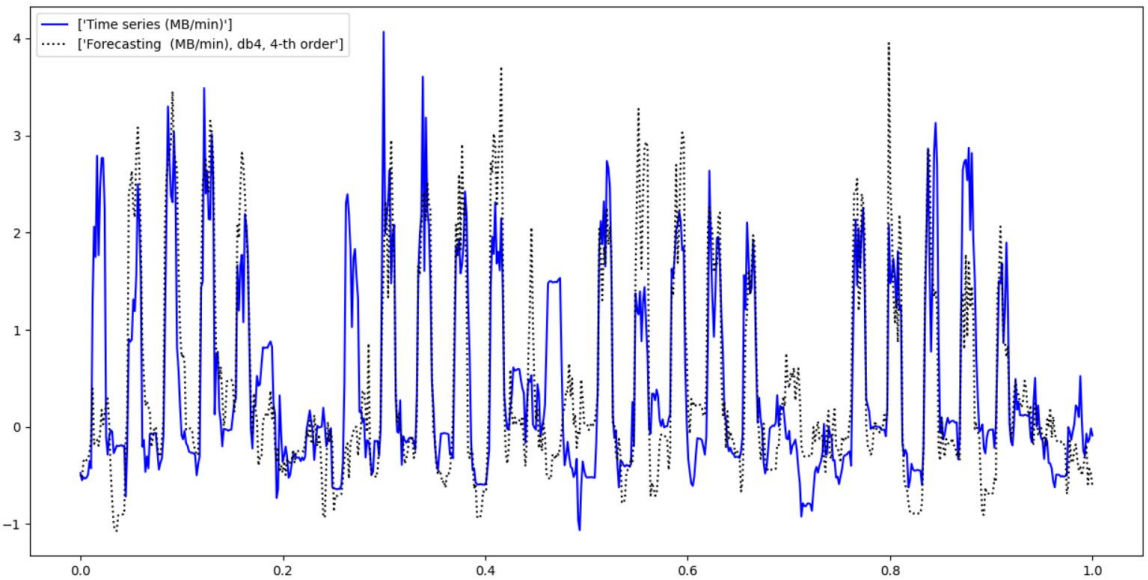

Рис. 4.16. Прогнозування місячних навантажень на вебсервер, без урахування аномальних викидів трафіку

На рис. 4.14, 4.16, 4.17 рівень вебтрафіку вимірюється в мегабайтах. По горизонтальній осі часу відкладено умовний час, де одиницею вимірювання обирається відповідний період прогнозування.

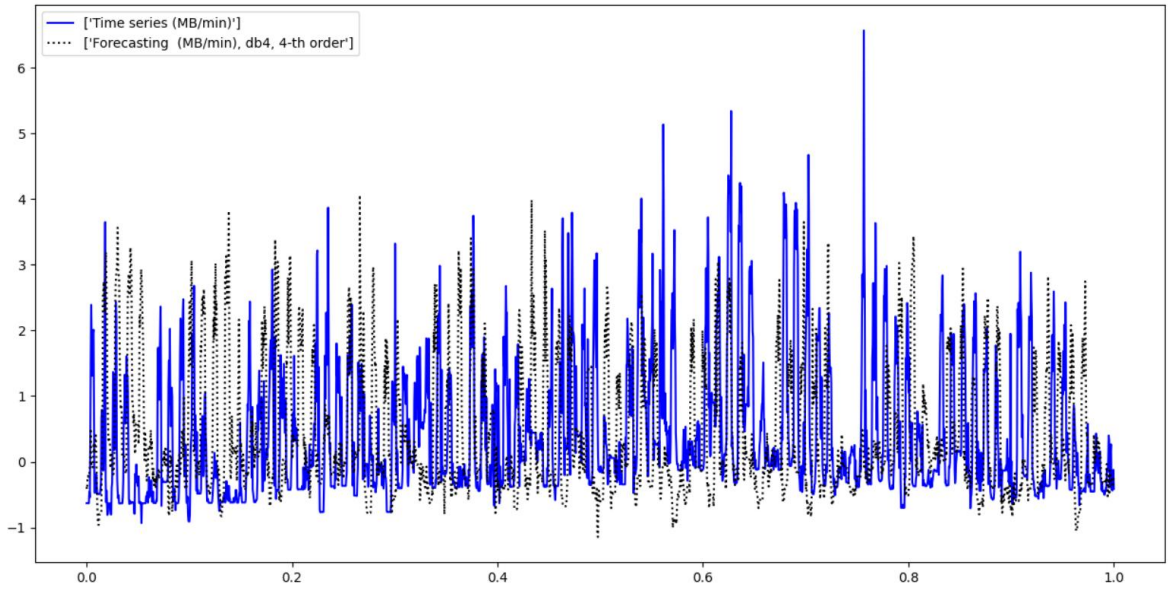

Рис. 4.17. Прогнозування квартальних навантажень на вебсервер

На рис. 4.17 зображені результати прогнозування трендових (квартальних) навантажень на вебсервер, при цьому середньоквадратична помилка апроксимації становила 8 – 15%.

Результати числових експериментів можна оцінити для задач прогнозування вебнавантаження наступним чином: для тижневих прогнозів похибка 3–8 % вважається високоточним результатом, що дозволяє ефективно планувати ресурси; для місячних прогнозів помилка 5–12 % є прийнятною, оскільки деяка похибка очікувана через усереднення та аномалії, але прогнози залишаються корисними; для квартальних прогнозів похибка 8–15 % є задовільною, оскільки довгострокові прогнози мають більшу невизначеність, проте така точність зазвичай достатня для стратегічного планування навантаження.

4.4. Оцінка ефективності методу для заданих переметрів вебсервера

Інтернет є масштабною розподіленою інформаційною системою,

побудованою на клієнт-серверній архітектурі. Таким чином, навантаження на вебсервер формується з численних запитів, що надходять від різних клієнтів. Для дослідження ми проаналізували журнали доступу вебсервера однієї з регіональних аварійно-диспетчерських служб в Україні за період 2021 – 2023 років. Журнали містять інформацію про кожен запит, оброблений сервером, зокрема: ім'я хоста, час запиту, ім'я запитуваного файлу та розмір відповіді в байтах.

Оскільки дані вебсервера мають розріджену структуру з 5-хвилинними інтервалами, ми створили перший набір даних, нормалізований за діапазоном середньогодинного рівня вебтрафіку, що відображає запити на передачу, оброблені сервером. Перевірка даних показала наявність різних режимів у часі через різке збільшення вебтрафіку щодня в робочий час, за винятком вихідних днів, а також фонове навантаження в неробочі дні та години. Тому було вирішено поділити часовий ряд на підсерії тижневої тривалості та використовувати середні 5-хвилинні значення для подальших експериментів.

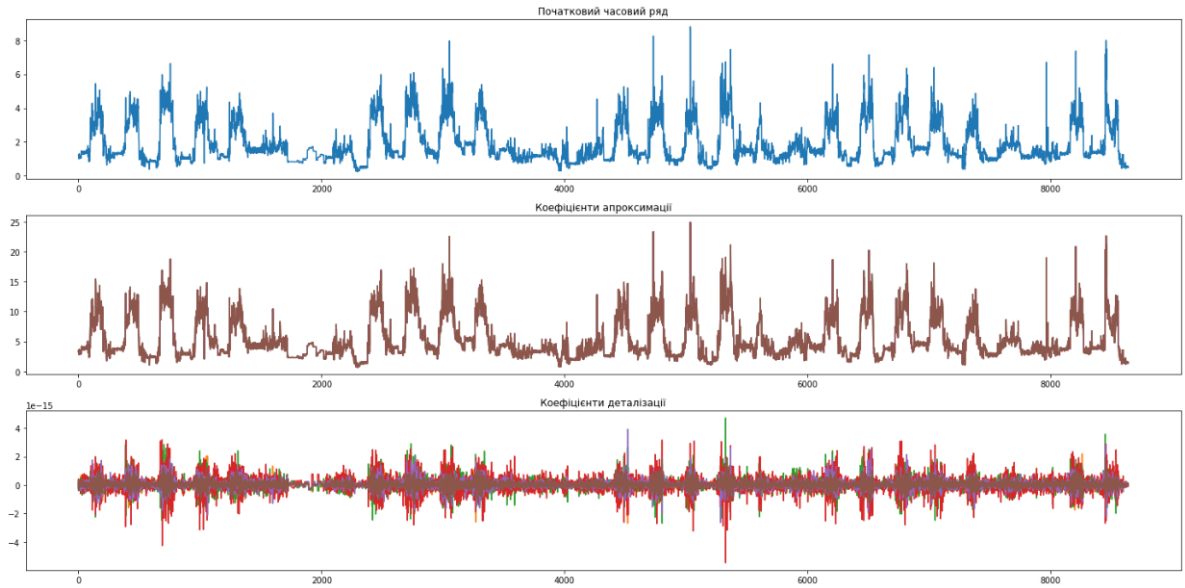

Рис. 4.18. Розклад рівня навантаженості вебсервера на вейвлет-коефіцієнти

Вибір найбільш ефективного базисного вейвлету дозволив провести числові експерименти, спрямовані на верифікацію отриманих результатів шляхом перевірки точності апроксимації статистичних даних та можливості використання

вейвлет-коефіцієнтів для аналізу локальних особливостей процесу навантаження. За результатами аналізу були отримані графіки апроксимації рівня вебтрафіку за допомогою вейвлет-спектра різної глибини розкладу, які зображені на рис. 4.18.

Видно, що в досліджуваному процесі можна виділити чотири періодичні компоненти, що відповідають певним часовим інтервалам доби.

У рамках дослідження було проведено порівняльні експерименти для оцінки методів прогнозування навантаження на вебсервер з використанням статистичних даних вебсервера за період з липня 2018 року по червень 2024 року.

Також було виконано зворотне дискретне вейвлет-перетворення сигналу на основі вейвлет-коефіцієнтів. Графіки спостережуваного і прогнозованого рівнів навантаження представлені на рис. 4.19. Де рівень вебтрафіку вимірюється в мегабайтах. По горизонтальній осі часу відкладено умовний час, де одиницею вимірювання обирається визначений період прогнозування.

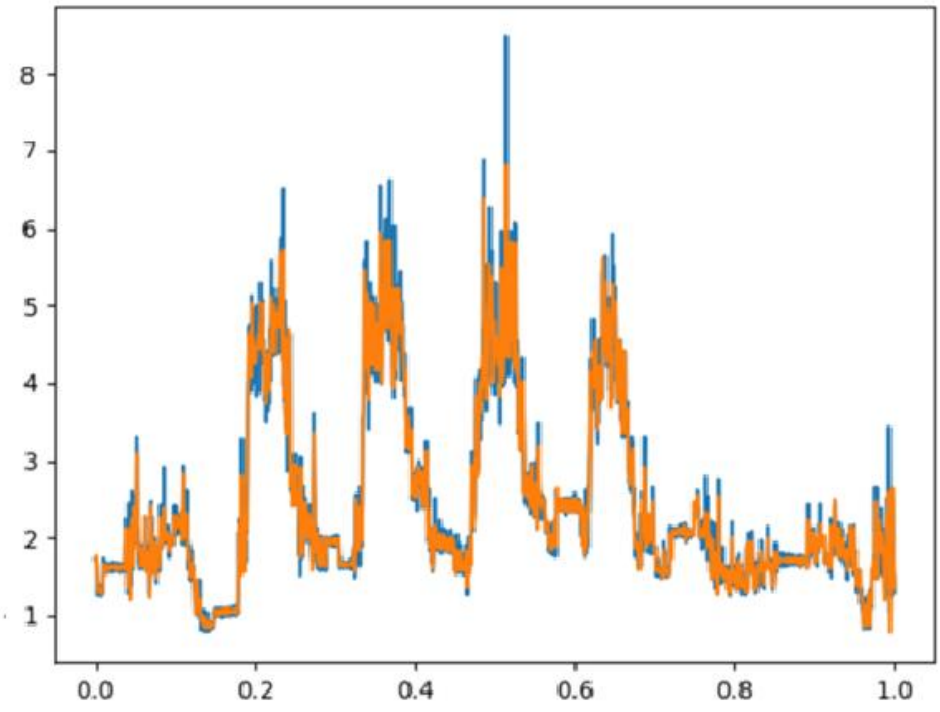

Рис. 4.19. Спостережуваний (синій) і прогнозований (жовтий) рівні вебтрафіку

Зазначені графіки демонструють високу схожість. Середньоквадратична похибка апроксимації тижневих навантажень на вебсервер не перевищує 3 – 8%, в той час як середня похибка апроксимації виконана за допомогою класичних

поліноміальних методів прогнозування, склала близько 10%.

Ефективність прогнозування навантаження на вебсервери значною мірою залежить від реалізації комплексу базових процедур, які формують основу аналітичного підходу. У першому наближенні можна вважати, що критично важливими компонентами $\boldsymbol{G}$ є врахування ключових етапів обробки даних, адаптації моделей до характеристик сигналу та параметризації методів тощо.

Таблиця 4.6

Оцінка ефективності прогнозування навантаження на вебсервер

№	Дослідження	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}	S
1	Tambe [81]	0,0	0,8	0,0	0,0	0,0	0,0	0,0	0,9	0,0	0,0	1,7
2	Lohrasbinasab [51]	0,0	0,0	0,0	0,0	0,0	0,8	0,0	0,0	0,0	0,6	1,4
3	Mahmood [53]	0,0	0,0	0,0	0,0	0,9	0,0	0,0	0,7	0,0	0,8	2,4
4	Wang [91]	0,8	0,8	0,0	0,0	0,0	0,9	0,0	0,0	0,0	0,0	2,5
5	Tian [85]	0,0	0,6	0,0	0,0	0,0	0,7	0,0	0,8	0,0	0,8	2,9
6	Prajam [66]	0,0	0,0	0,7	0,0	0,7	0,0	0,0	0,0	0,0	0,8	2,2
7	Irie [40]	0,0	0,0	0,8	0,0	0,7	0,0	0,0	0,9	0,0	0,9	3,3
8	Tedjopurnomo [82]	0,4	0,0	0,0	0,0	0,0	0,6	0,6	0,7	0,0	0,0	2,3
9	Mamun [55]	0,8	0,0	0,0	0,8	0,0	0,9	0,7	0,0	0,8	0,9	4,9
10	Saxena [74]	0,7	0,0	0,9	0,0	0,4	0,6	0,0	0,6	0,9	0,7	4,8
11	Palvel [63]	0,6	0,0	0,5	0,0	0,8	0,7	0,9	0,0	0,6	0,6	4,7
12	Chaturvedi [11]	0,5	0,6	0,7	0,0	0,8	0,8	0,0	0,0	0,9	0,8	5,1
13	**IWMLP**	**0,4**	**0,9**	**0,9**	**0,8**	**0,9**	**0,7**	**0,8**	**0,6**	**0,9**	**0,9**	7,8

Аналіз характеристик наявних моделей та методів прогнозування підтверджує доцільність застосування розробленого підходу, а перспективи його вдосконалення полягають у більш точному розрахунку критеріїв ефективності та їх вагових коефіцієнтів для підвищення точності прогнозів, що представлено у таблиці 4.6 через критерії $\boldsymbol{G}$ та функцію ефективності S.

На рис. 4.20 зображено фрагмент часового ряду вебтрафіку за листопад 2021

року з інтервалом у 1 годину, де помітно виражений добовий цикл активності та сталий фоновий рівень.

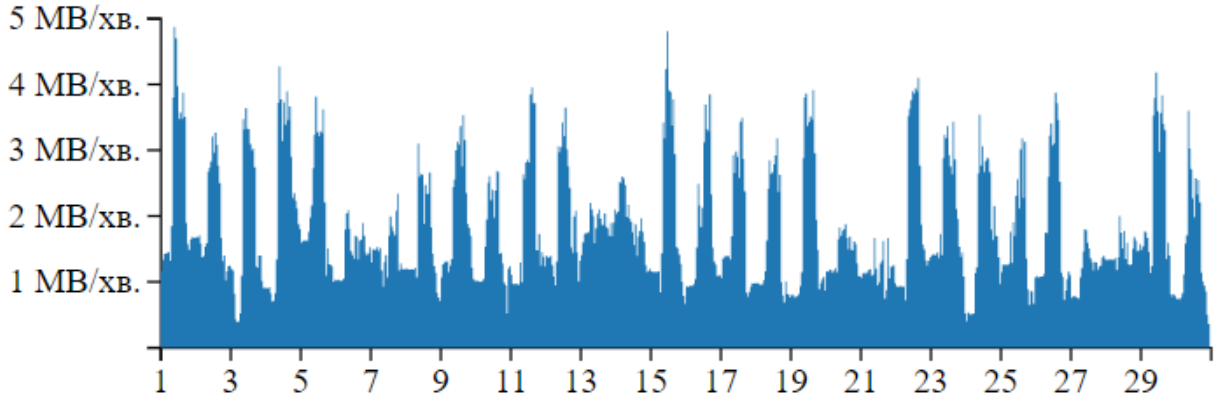

Рис. 4.20. Рівень вебтрафіку досліджуваного вебсервера протягом місяця

З графіка видно, що трафік має чітку добову періодичність, тоді як тижнева сезонність менш очевидна. Наразі дослідження зосереджено не на аналізі причин самоподібності, а на графічному методі її виявлення. Самоподібний часовий ряд характеризується тим, що його агреговані версії зберігають ту саму автокореляційну структуру, що й оригінал, що свідчить про наявність довгострокової залежності, де автокореляція спадає гіперболічно при зростанні лагу. У таких випадках на логарифмічному графіку залежності дисперсії від масштабу m з'являється пряма лінія зі спадом $\beta > -1$, а самоподібність ряду визначається через параметр Херста $H = 1 + \beta/2$. Якщо H наближається до 1, це вказує на високу ступінь самоподібності. Для перевірки важких "хвостів" розподілу використовувався тест на основі центральної граничної теореми з побудовою логарифмічних графіків додаткового кумулятивного розподілу (LLCD), де розподіли з необмеженою дисперсією демонструють сталий нахил незалежно від m. В результаті регресійного аналізу залежності між варіацією та масштабом було отримано оцінку $H = 0.72$, що підтверджує наявність самоподібності у досліджуваному ряді вебтрафіку (рис. 4.21).

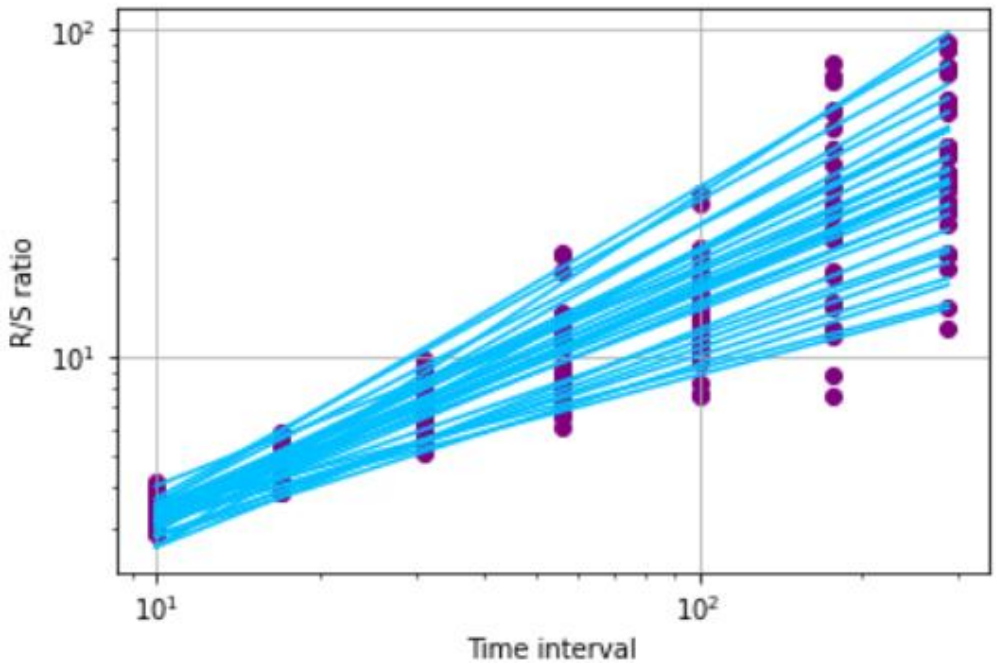

Рис. 4.21. Залежність параметра Херста від добових коливань вебтрафіка

Кількісна оцінка ефективності моделі здійснювалась через показники похибки прогнозування на один крок уперед, зокрема за нормалізованою середньоквадратичною помилкою NMSE (Normalized Mean Square Error):

$$NMSE = \frac{1}{n\sigma_x^2} \cdot \sum_{i=1}^{n} (x_i - \hat{x}_i)^2, \tag{4.4}$$

де x_i – фактичне значення (реальні дані), $\hat{x}_i$ – прогнозоване значення вебтрафіку, σ_x^2 – дисперсія фактичних значень, n – кількість спостережень.

Цей показник обчислювався як відношення середньоквадратичного відхилення прогнозованих значень від фактичних до дисперсії вихідного ряду. Для прогнозування використовувалась рекурентна нейромережа з подачею вейвлет-коефіцієнтів як зовнішніх входів. Було зібрано набір з 2016 безперервних спостережень вебтрафіку за один тиждень, з яких 1728 точок використано для навчання, а решта – для тестування. Через виражену добову періодичність дані мали складну структуру, яку мережа повинна була навчитися відображати. Після серії експериментів обрана архітектура LSTM у конфігурації 5-5-1 (5 вхідних блоків, 5 прихованих блоків і 1 вихідний блок) продемонструвала добру якість прогнозів, незважаючи на шумові варіації та невелику кількість параметрів

(рис. 4.22).

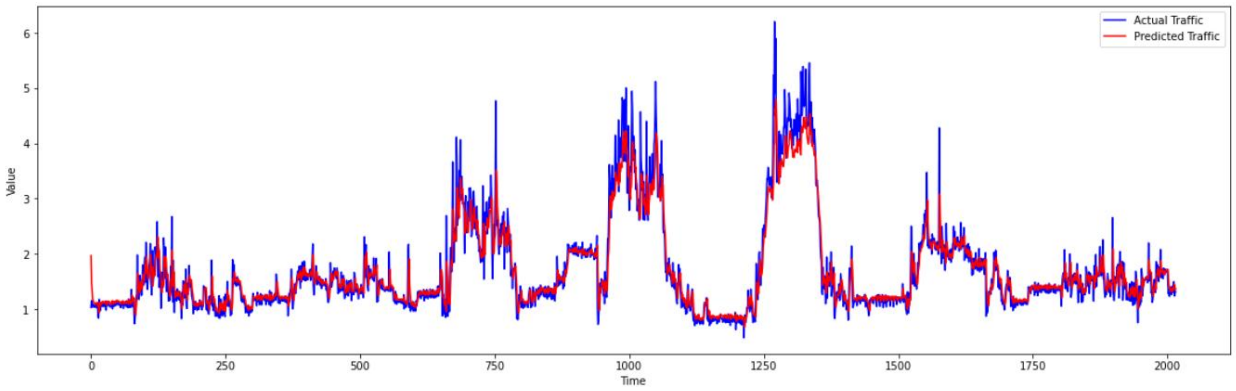

Рис. 4.22. Приклад досліджуваних тижневих коливань рівня вебтрафіка

Похибка прогнозування на тестовому наборі становила $NMSE = 0.9$, що вважається прийнятним результатом. Для підвищення точності було зменшено прогнозний горизонт до 1 години вперед, що дозволило одночасно збільшити актуальність вхідних даних і розмір навчального набору. Та сама архітектура використовувалася для прогнозування окремих вейвлет-компонент, де кожна мережа отримувала значення певного порядку роздільної здатності. Затримки між шарами адаптувались до змінної поведінки коефіцієнтів у залежності від рівня розкладу, охоплюючи часові вікна до 45 хвилин. Прогнози запускались п’ять разів для кожного з п’яти рядів $W(t,j)$, а результати порівнювались із помилками авторегресійного прогнозу, демонструючи покращення точності для вейвлетів з вищим порядком, які характеризуються більш гладкою динамікою.

Нормалізована середньоквадратична помилка (NMSE) для однокрокового прогнозу, отриманого шляхом рекомбінації результатів RNN, склала 0,75. Це свідчить про те, що зі збільшенням порядку роздільної здатності здатність нейронної мережі відтворювати динаміку процесу суттєво знижується. На вищих рівнях розкладу сигнал стає згладженим, і відповідно, інформаційне навантаження зменшується, що ускладнює навчання мережі. Тому було прийнято рішення обмежити вейвлет-декомпозицію п’ятим рівнем.

Для перевірки практичної ефективності підходу було застосовано ту ж архітектуру, що й для тижневого прогнозування вебтрафіку, що дало значення

NMSE = 0,80. Це підтверджує, що поєднання вейвлет-розкладу з рекурентною мережею дійсно покращує точність прогнозування.

Для глибшого аналізу застосовувався коефіцієнт взаємної кореляції, який дозволяє побачити, наскільки один ряд передбачає або слідує за іншим у часі:

$$R_{xy}(k) = E[(x_t - \bar{x}\,)(y_{t+k} - \bar{y}\,)], \tag{4.5}$$

де x_t, y_t – два часові ряди (фактичні дані та прогноз), $\bar{y}, \bar{y}$ – середні значення відповідних рядів, k – лаг, тобто зсув одного ряду відносно іншого, E – математичне сподівання часового ряду.

А також коефіцієнт кореляції – основний індикатор лінійного зв’язку між прогнозованими і фактичними даними. Це нормалізована версія взаємної кореляції, яка виводить значення в діапазоні від -1 до 1:

$$\rho_{xy} = \frac{R_{xy}(0)}{\sigma_x \sigma_y}, \tag{4.6}$$

де $R_{xy}(0)$ – взаємна кореляція без зсуву (при лагу $k = 0$), σ_x, σ_y – стандартні відхилення рядів x і y.

При аналізі результатів стало очевидним, що точність прогнозування погіршується зі збільшенням кількості об’єднаних прогнозів. Наприклад, поєднання лише компонент w_5 і w_4 дало NMSE = 0,15; додавання w_3 збільшило помилку до 0,25, а включення w_2 – до 0,40. Повна рекомбінація всіх компонент призвела до NMSE = 0,75.

Ці результати свідчать про те, що вейвлет-компоненти та їх прогнозовані ряди не є статистично незалежними. Очікування, що результат об'єднання прогнозів дасть середнє значення помилок між найгіршою та найкращою, не справджується через високу взаємну кореляцію похибок, що не була врахована при об'єднанні.

Попри зростання помилки у сукупному прогнозі, можна виділити позитивний момент. У 85% випадків передбачене значення мало той самий напрямок зміни, що й реальне. Це свідчить про потенціал використання нейронних мереж, з даними попередньо обробленими декомпозицією вейвлет-перетворень, для короткострокового прогнозування вебтрафіку. Однак для покращення точності

на високих рівнях деталізації потрібні подальші дослідження та подальше вдосконалення методів.

Перспективним напрямом подальших досліджень є впровадження гібридних моделей прогнозування, де WNN використовується для очищення та декомпозиції сигналу, а LSTM – для точнішого передбачення довгострокових тенденцій. Також доцільно застосовувати адаптивні методи виявлення та корекції аномальних викидів трафіку, багаторівневі вейвлет-перетворення та оптимізацію параметрів моделей, що дозволить зменшити середньоквадратичну помилку прогнозів, особливо для місячних та квартальних інтервалів. Реалізація таких підходів підвищить надійність і точність систем прогнозування навантаження на вебсервери, забезпечуючи більш ефективне управління ресурсами та планування роботи інфраструктури.

Висновки до розділу 4

У розділі 4 було проведено дослідження прогнозування навантаження на вебсервер із використанням вейвлет-перетворень і різних часових масштабів даних (денних, тижневих, місячних тощо). Було показано, що застосування комплексного вейвлету Морле дозволяє ефективно візуалізувати та аналізувати тренди та флуктуації трафіку за допомогою скалограм, що сприяє візуальному представленню вебтрафіку.

Результати прогнозів демонструють різні рівні похибки залежно від масштабу: тижневі прогнози мають найменшу середньоквадратичну помилку (3–8 %), місячні прогнози характеризуються помірною похибкою (5–12 %), а квартальні прогнози – більшою (8–15 %), що відповідає очікуваному зростанню невизначеності при збільшенні інтервалу прогнозування. Аналіз отриманих даних підтверджує, що комбінування вейвлет-фільтрації з подальшим прогнозуванням дозволяє досягати прийнятної точності на різних часових масштабах, а також забезпечує наочну інтерпретацію динаміки навантаження, що важливо для оперативного та стратегічного управління ресурсами вебсерверу.

При цьому було виявлено, що однією з ключових проблем використання

вейвлет-аналізу є правильний вибір типу базисного вейвлету, параметри якого повинні відповідати динаміці навантаження конкретної вебсистеми. Це підтверджується експериментальними результатами: наприклад, під час прогнозування тижневого навантаження обраного вебсерверу диспетчерської служби найнижче значення середньоквадратичної помилки (4,7 %) було отримано при використанні вейвлету Койфлета Coif5, що свідчить про його високу здатність до точного відтворення характеру зміни сигналу. Для біортогонального вейвлету (bior3.7) похибка становила 5,2 %, що також демонструє його ефективність при аналізі згладжених компонент сигналу. У свою чергу, вейвлети Добеші та Симлет забезпечили дещо вищі значення похибки – 7,3 % та 6,8 % відповідно, що може бути пов’язано з меншою симетричністю базисних функцій і складністю у відновленні швидких змін навантаження.

Такі кількісні характеристики демонструють важливість узгодження форми базисної функції з особливостями досліджуваного сигналу. Однією з ключових переваг розробленої системи є можливість автоматизованого підбору оптимального вейвлету без потреби у тривалих обчислювальних експериментах, що суттєво скорочує час побудови моделі та підвищує ефективність процесу прогнозування.

ВИСНОВКИ

У межах виконаного дисертаційного дослідження було вирішено науково-практичну задачу підвищення точності прогнозування навантаження на вебсервер за рахунок використання вейвлет-перетворень у поєднанні з моделями глибинного навчання. Робота охоплює повний цикл розробки від аналізу предметної області до реалізації експериментальної системи, що демонструє ефективність запропонованого підходу.

Було показано, що одним з найбільш перспективних напрямків розвитку систем прогнозування навантаженості на вебсервер є удосконалення математичного забезпечення шаблонів його нормальної поведінки на основі вейвлет-аналізу. Проведено глибокий аналіз теоретичних основ вейвлет-перетворень та особливостей їх застосування до часових рядів. Додатково класифіковано підходи до вибору базисних вейвлетів на основі формалізованих критеріїв, таких як симетричність, ортогональність, компактність, здатність до масштабування тощо.

Основні результати роботи полягають в наступному.

1. Вперше розроблено інтегровану модель прогнозування навантаження вебсерверу комп’ютерних мереж загального призначення, яка на відміну від існуючих враховує критерії для вибору оптимального рівня вейвлет-декомпозиції, що дозволяє уникати надмірного згладжування сигналу та зниження точності. У моделі функціональні параметри подані у вигляді багатоперіодичного ряду даних із використанням аналітичного апарату їх фільтрації та обмежень щодо застосування вейвлет-перетворень. Модель поєднує вейвлет-перетворення в ролі інтелектуального фільтру, що забезпечує багатошарове представлення сигналу з фіксацією як довгострокових тенденцій, так і короткотермінових змін, та нейронну мережу типу RNN (зокрема LSTM). Це забезпечує можливість розробки методу застосування вейвлет-перетворень для прогнозування навантаження на вебсервер.

2. Вперше розроблено метод вибору типу базисного вейвлету, який на відміну від існуючих за рахунок застосування запропонованих принципів та

критеріїв оцінки ефективності типів вейвлетів дозволяє уникнути довготривалих експериментів, пов’язаних з визначенням найбільш ефективного типу базисного вейвлету.

3. Отримала подальший розвиток методологічна база прогнозування навантаження на вебсервер, що завдяки розробленим принципам та критеріям оцінки ефективності типів материнського вейвлету забезпечила можливість створення методу вибору оптимального типу материнського вейвлету. Важливою перевагою розробленого методу є можливість уникнення довготривалих чисельних експериментів під час побудови вейвлет-моделі для визначення типу базисного вейвлету.

4. Отримав подальший розвиток метод застосування вейвлет-перетворень для прогнозування навантаження на вебсервер, що за рахунок використання розробленої моделі прогнозування функціональних параметрів вебсерверу та розробленого методу вибору типу материнського вейвлету, дозволяє зменшити на 10-12% похибку прогнозування навантаження на вебсервер, який застосовується в комп’ютерних мережах загального призначення, порівняно з класичними аналогічними методами.

5. Розроблено програмне забезпечення з вбудованим модулем обробки вебтрафіку, вейвлет-аналізом і прогнозуючим блоком, за допомогою якого були протестовані експериментальні дані, що підтверджують коректність та ефективність теоретично отриманих результатів. Проведено серію експериментів, що довели здатність моделі до ефективного прогнозування варіацій навантаження з відносно невисокою середньоквадратичною помилкою (NMSE = 0,75). Хоча рекомбінування прогнозів з усіх масштабів не завжди підвищує точність (через корельовані похибки), використання обмеженої кількості найбільш інформативних компонентів дозволяє досягти значно кращих результатів (наприклад, NMSE = 0,13 при використанні лише двох масштабів).

Практична значущість роботи полягає у можливості інтеграції розробленої системи в інфраструктуру вебсерверів для забезпечення адаптивного керування навантаженням та підвищення надійності обслуговування користувачів.

Перспективи подальших досліджень пов’язані з уточненням процесу розрахунку критеріїв ефективності запронованої інтегральної моделі та вдосконаленням запропонованого методу вибору типу материнського вейвлету шляхом автоматизації процедури визначення функціональних параметрів вебсерверу. Також результати дослідження можуть бути розширені шляхом використання ансамблів моделей, автоматичної оптимізації глибини вейвлет-декомпозиції, а також включення контекстної інформації про сеанси користувачів у прогнозну модель.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Abramovitch F., Benjamini Y. Adaptive thresholding of wavelet coefficients. *Computational Statistics and Data Analyses*. – 1996. – Vol. 222, № 2. – P. 351–361.

2. Abramovitch F., Sapatinas T., Silverman B. W. Wavelet thresholding via a Bayesian approach. *Journal of the Royal Statistical Society. Series B*. – 1998. – № 60. – P. 725–749.

3. Aldroubi A., Cabrelli C., Molter U. Wavelets on irregular grids with arbitrary dilation matrices, and frame atoms for L2(Rd). – *Preprint*. – March 30, 2004.

4. Aldroubi A., Sun Q., Tang W.-S. Non-uniform average sampling and reconstruction in multiply generated shift-invariant spaces. *Constructive Approximation*. – 2004. – Vol. 20. – P. 173–189.

5. Arslan S. A hybrid forecasting model using LSTM and Prophet for energy consumption with decomposition of time series data. *PeerJ Computer Science*, vol. 8, 2022. Article e1001. DOI: 10.7717/peerj-cs.1001.

6. Banakar A., Azeem M.F., Kumar V. Comparative study of wavelet based neural network and neuro-fuzzy systems. *International Journal of Wavelets, Multiresolution and Information Processing*. – 2007. – Vol. 5, No. 6. – P. 879–906. – DOI: 10.1142/S0219691307002099.

7. Bapiyev I.M., Aitchanov B.H., Tereikovskyi I.A., Tereikovska L.A., Korchenko A.A. Deep neural networks in cyber attack detection systems. *International Journal of Civil Engineering and Technology (IJCIET)*. – 2017. – Vol. 8, Issue 11. – P. 1086–1092.

8. Benjamini Y., Hochberg Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B*. – 1995. – № 57. – P. 289–300.

9. Brenner M. Where Does Your Website Traffic Come From? Top Web Traffic Sources Research. *Marketing Insider Group*, April 10, 2023. [Online]. Available: https://marketinginsidergroup.com/content-marketing/top-web-traffic-sources

10. Bruce A. G., Gao H.-Y. Waveshrink with firm shrinkage. *Statistica Sinica*. –

1997. – Vol. 4, № 6. – P. 855–874.

11. Chaturvedi S., Rajasekar E., Natarajan S., McCullen N. A comparative assessment of SARIMA, LSTM RNN and Fb Prophet models to forecast total and peak monthly energy demand for India. *Energy Policy*, vol. 168, 2022. Article 113097. DOI: 10.1016/j.enpol.2022.113097.

12. Cheng J., Tiwari S., Khaled D., Mahendru M., Shahzad U. Forecasting Bitcoin prices using artificial intelligence: Combination of ML, SARIMA, and Facebook Prophet models. *Technological Forecasting and Social Change*, vol. 198, 2024. Article 122938. DOI: 10.1016/j.techfore.2023.122938.

13. Chen Q.-S., Zhang X., Xiong S.-H., Chen X.-W. Short-term Power Load Forecasting with Least Squares Support Vector Machines and Wavelet Transform. // *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*, 2008, Vol. 3, pp. 1425–1429.

14. Chen Zefei, Xu Jianmin, Lin Yongjie, Feng Bin, Huang Zihao. A Traffic Flow Forecasting Method Regarding Traffic Network as a Digraph. *International Journal of Pattern Recognition and Artificial Intelligence*. – 2021. – Vol. 35, No. 15. – P. 2159043. – DOI: 10.1142/S0218001421590436.

15. Chiti F., Fantacci R., Baffoni F., Vespri V. "Admission Control Algorithms based on Self-Similar Traffic Modeling for IP Networks," in *Applied and Industrial Mathematics in Italy*, June 2005, pp. 225–236.

16. Chui C. K. An Introduction to Wavelets (Vol. 1). – San Diego: *Academic Press*, 1992. – 266 p.

17. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to Algorithms*. – Third Edition. – Cambridge: *MIT Press*, 2009. – 1292 p.

18. Dang X. Ch., Hao Z. J., Li Y., Lu Z. Y., Gao Q. Network traffic forecasting combination model based on wavelet transform and chaos algorithm. *International Journal of Wavelets, Multiresolution and Information Processing*. – 2014. – Vol. 12, № 3. DOI: 10.1142/S0219691314500295.

19. Dasari R., Renikunta R., Perati M. R. "Self-Similar Network Traffic Modelling Using Fractal Point Process-Markovian Approach," in *Fractals, Wavelets, and*

Their Applications, Springer Proceedings in Mathematics & Statistics, vol. 92. Cham: Springer, 2014. DOI: 10.1007/978-3-319-08105-2_27.

20. Daubechies I., Guskov I., Sweldens W. Commutation for irregular subdivision. *Constructive Approximation*. – 2001. – Vol. 17, № 4. – P. 479–514.

21. Donoho D. L. De-noised by soft-thresholding. *IEEE Transactions on Information Theory*. – 1995. – Vol. 41, № 3. – P. 613–627.

22. Donoho D. L., Johnstone I. M. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*. – 1995. – Vol. 90, № 432. – P. 1200–1224.

23. Donoho D. L., Johnstone I. M. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*. – 1994. – Vol. 81, № 3. – P. 425–455.

24. Ellins J. Google Analytics: hour of day & day of week reports. – Posted on 07/02/2020. – URL: https://www.hallaminternet.com/google-analytics-hour-of-day-day-of-week-reports

25. Golub G. H., Heath M., Wahba G. Generalized cross validation as a method for choosing a good ridge parameter. *Technometrics*. – 1979. – Vol. 21. – P. 215–222.

26. González-Audícana M., Otazu X., Fors O., Seco A. Comparison between Mallat's and the 'à trous' discrete wavelet transform based algorithms for the fusion of multispectral and panchromatic images. *International Journal of Remote Sensing*, vol. 26, no. 3, 2005. P. 595–614.

27. Graven C., Wahba G. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross validation. *Numerische Mathematik*. – 1979. – Vol. 31, № 3. – P. 377–403.

28. Guenoukpati, A., Agbessi, A. P., Salami, A. A., & Bakpo, Y. A. Hybrid Long Short-Term Memory Wavelet Transform Models for Short-Term Electricity Load Forecasting. *Energies*, 2024, *17*(19), 4914. https://doi.org/10.3390/en17194914

29. G'ulomov S. H. R., Jumayev S. N. "Recent advances in algorithms for detecting and mitigating network traffic anomalies," *III-Int. Sci. Conf. "Scientific Foundations of Raising the Use of Information Technologies to a New Level and Modern Problems of Automation",* Nov. 20, 2024, pp. 662–667. DOI: 10.5281/zenodo.14220202.

30. Hariharan R. Web Server Performance Modeling / R. Hariharan, Van der Mei, P. K. Reeser. // *Telecomunication Systems*. – 2001. – vol. 16. – P. 361 –378.

31. Harmantzia F. C., Hatzinakos D. Heavy Network Traffic Modeling and Simulation using Stable FARIMA Processes // *IEEE Trans. Signal Proc. Lett.* – 2000. – Vol. 5. – P. 48–50.

32. Heyman D. P., Sobel M. J. Stochastic Models in Operations Research: Stochastic optimization. Dover Books on Computer 155 Science Series. – *Dover Publications*, 2003.

33. Hu Z., Tereikovskyi I., Tereikovska L., Tsiutsiura M., Radchenko K. Applying wavelet transforms for web server load forecasting. *In: Hu Z., Petoukhov S., Dychka I., He M. (eds.) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019*. *Advances in Intelligent Systems and Computing*, vol. 938. Cham: Springer, 2020. P. 13–22. https://doi.org/10.1007/978-3-030-16621-2_2

34. Hu Z., Tereykovskiy I. A., Tereykovska L. O., Pogorelov V. V. Determination of structural parameters of multilayer perceptron designed to estimate parameters of technical systems. *International Journal of Intelligent Systems and Applications (IJISA)*. – 2017. – Vol. 9, № 10. – P. 57–62. https://doi.org/10.5815/ijisa.2017.10.07

35. Huang X., Ma C., Zhao Y., Wang K., Meng W. A hybrid model of neural network with VMD–CNN–GRU for traffic flow prediction. *International Journal of Modern Physics C*. – 2023. – Vol. 34, Issue 12. – P. 2350159. – DOI: 10.1142/S0129183123501590.

36. Huang Y., Zhang J., Wu L., Wang H. A trous wavelet and neural network based constellation networks traffic forecasting // *Wavelet Active Media Technology and Information Processing*. – Singapore: World Scientific Publishing Company, 2006. – P. 381–386. – DOI: 10.1142/9789812772763_0059

37. Ikharo A. B., Anyachebelu K. T. "Neural Network Prediction of Self-Similarity Network Traffic," *NIPES J. Sci. Technol. Res.*, vol. 4, no. 4, pp. 18–26, 2022. DOI: 10.5281/zenodo.7390658.

38. Iļņickis S. "Research of the Network Server in Self-Similar Traffic Environment," *Telecommunications and Electronics*, vol. 4, pp. 65–72, 2004.

39. Iranmanesh H., Abdollahzade M., Miranian A., Hassani H. A developed wavelet-based local linear neuro fuzzy model for the forecasting of crude oil price. *International Journal of Energy and Statistics*. – 2013. – Vol. 1, Issue 3. – P. 171-193. – DOI: 10.1142/S2335680413500129.

40. Irie K., Glynn C., Aktekin T. Sequential modeling, monitoring, and forecasting of streaming web traffic data. *Annals of Applied Statistics*, vol. 16, no. 1, 2022. P. 300–325.

41. Katkovnik V., Eqiazarian K., Astova J. Weighted median filter with varying adaptive windows size. *Proceedings of the IASTED International Conference on Signal Processing and Communications*. – Spain, 2000. – P. 329–333.

42. Khatoon S., Ibraheem, Gupta P., Shahid M. "Comparison of fuzzy time series, ANN and wavelet techniques for short term load forecasting," *Int. J. Power Electron. Drive Syst.*, vol. 14, no. 2, pp. 1260–1269, 2023. DOI: 10.11591/ijpeds.v14.i2.pp1260-1269.

43. Kumar A., Agrawal D.P., Joshi S.D. Multiresolution forecasting for US retailing using wavelet decompositions. *International Journal of Wavelets, Multiresolution and Information Processing*. – 2003. – Vol. 1, No. 4. – P. 449–463. – DOI: 10.1142/S0219691303000281.

44. Kuo W., Zuo M. Optimal Reliability Modeling. – New York: *John Wiley & Sons*, 2002. – 560 p.

45. Kutzkov K. ARIMA vs Prophet vs LSTM for Time Series Prediction. *Neptune.ai*, Jan. 24, 2025. [Online]. Available: https://neptune.ai/blog/arima-vs-prophet-vs-lstm

46. Lahmiri S., Boukadoum M. Intelligent ensemble forecasting system of stock market fluctuations based on symmetric and asymmetric wavelet functions. *Fluctuation and Noise Letters*. – 2015. – Vol. 14, Issue 4. – P. 1550033. – DOI: 10.1142/S0219477515500339.

47. Lee G. R., Gommers R., Wasilewski F., Wohlfahrt K., O'Leary A. PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software*. – 2019. – Vol. 4 (36). – Article № 1237. https://doi.org/10.21105/joss.01237

48. Li S., Jiang M. "The New Complex-Valued Wavelet Neural Network," *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 12, pp. 613–622, 2014. DOI: 10.12928/telkomnika.v12i3.95.

49. Liu S., He Q., Chen Y., Zhang F. Wavelet and VMD enhanced traffic forecasting and scheduling method for edge cloud networks. *Computers and Electrical Engineering*. – 2025. – Vol. 121. DOI: 10.1016/j.compeleceng.2024.109862.

50. Liu Y., Ouyang J., Yan Y. De-Noising of Life Feature Signals Based on Wavelet Transform // *Industrial Engineering, Machine Design and Automation (IEMDA 2014) & Computer Science and Application (CCSA 2014)*. – 2015. – P. 284–291. DOI: 10.1142/9789814689007_0040.

51. Lohrasbinasab A., Shahraki A., Taherkordi A., Jurcut A. D. From statistical- to machine learning-based network traffic prediction. *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 4, 2022.

52. Lu Y., Sheng B., Fu G., Luo R., Chen G., Huang Y. Prophet-EEMD-LSTM based method for predicting energy consumption in the paint workshop. *Applied Soft Computing*, vol. 143, 2023. Article 110447. DOI: 10.1016/j.asoc.2023.110447.

53. Mahmood S., Ameen S., Yasin H., Omar N., Kak S., Najat Z., Salih A., Salim N. O. M., Ahmed D. Web Server Performance Improvement Using Dynamic Load Balancing Techniques: A Review. *Asian Journal of Research in Computer Science*, vol. 10, 2021. P. 47–62. DOI: 10.9734/AJRCOS/2021/v10i130234.

54. Mallat S. A Wavelet Tour of Signal Processing (2nd ed.). – San Diego: *Academic Press*, 1999. – 851 p.

55. Mamun Md S., Sohel Md N., Sami Md N., Sunny Md S., Roy D., Hossain E. A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models. *IEEE Access*, vol. PP, 2020. P. 1–1. DOI: 10.1109/ACCESS.2020.3010702.

56. Menascé D. A., Almeida V. A. F. Capacity Planning for Web Services: Metrics, Models, and Methods (Revised ed.). – Upper Saddle River: *Prentice Hall*, 2001. – 572 p.

57. Mihaylenko, Victor, Tereikovska, Liudmyla. (2019). Conceptual model of

neural network recognition of emotional condition of listeners of the distance learning system. *Management of Development of Complex Systems*, 38, 193 – 199, dx.doi.org/10.6084/m9.figshare.9788720.

58. Millán G. "A Simple Multifractal Model for Self-Similar Traffic Flows in High-Speed Computer Networks," *EasyChair Preprint* no. 5147, Mar. 15, 2021. [Online]. Available: https://easychair.org/publications/preprint/KGbN

59. Nason G. P. Choice of wavelet smoothness, primary resolution and threshold in wavelet shrinkage. *Statistics and Computing*. – 2002. – Vol. 12, № 2. – P. 219–227.

60. Nason G. P. Wavelet shrinkage using cross validation. *Journal of the Royal Statistical Society. Series B*. – 1998. – № 60. – P. 463–479.

61. Neelamani R., Choi H., Baraniuk R. ForWaRD: Fourier-wavelet regularized deconvolution for ill-conditioned systems. *IEEE Transactions on Signal Processing*. – 2004. – Vol. 52, № 2. – P. 418–426.

62. Oliynyk O., Taranenko Y., Losikhin D., Shvachka A. Examining the Kalman filter in the field of noise and interference with the non-Gaussian distribution. *Eastern European Journal of Enterprise Technologies*. – 2018. – Vol. 4, № 4 (94). – P. 36–42. https://doi.org/10.15587/1729-4061.2018.140649

63. Palvel S. Time Series Forecasting with Prophet and LSTM Hybrid Mode. *Medium*, Sept. 18, 2023. [Online]. Available: https://subashpalvel.medium.com/time-series-forecasting-with-prophet-and-lstm-hybrid-mode-75f5295605e5

64. Paxson V. "Fast approximate synthesis of fractional Gaussian noise for generating self-similar network traffic," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 5, pp. 5–18, 1997. DOI: 10.1145/269790.269792.

65. Paxson V. Fast Approximation of Self-Similar Network Traffic. *Preprint, Lawrence Berkeley Laboratory and EECS Division*, University of California, 1996. P. 1–12.

66. Prajam S., Wechtaisong C., Khan A.A. Applying machine learning approaches for network traffic forecasting. *Indian Journal of Computer Science and Engineering*, vol. 13, no. 2, 2022. P. 324–335.

67. Qiao Y., Wang Y., Ma C., Yang J. Short-term traffic flow prediction based on

1DCNN-LSTM neural network structure. *Modern Physics Letters B*. – 2020. – Vol. 35, Issue 2. – P. 2150042. – DOI: 10.1142/S0217984921500421.

68. VIAVI Solutions. *2023 State of the Network Study.* 15th consecutive VIAVI State of the Network survey, 2023. URL: https://www.viavisolutions.com/en-us/enterprise/resources/state-network/2023-state-network-study.

69. Radchenko K., Tereikovskyi O. Wavelet transforms for web server load calculating. *ITSec: Безпека інформаційних технологій: IX Міжнародна науково-технічна конференція*, 22–27 березня 2019 р. – Київ: НАУ, 2019. – Pp. 28–29.

70. Radchenko K., Tereykovskyi I. Integrated neural network and wavelet-based model for web server load forecasting. *SISIOT (Security of Infocommunication Systems and Internet of Things)*. – 2024. – Vol. 2, № 2, paper 02006. – P. 1 – 6. https://doi.org/10.31861/sisiot2024.2.02006

71. Radchenko K., Tereykovskyi I., Xing Tao. Automated Big Data Analysis Module. *XVII Науково-практична конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2024)* (м. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 20–22 листопада 2024 р.). – С. 239–245.

72. Radchenko K., Xing Tao. Design and implementation of an automated Big Data analysis module. *Матеріали XI Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 27 листопада 2024. – Київ: НУХТ, 2024. – С. 40–41.

73. Salis C. I., Paschalis A. E. M., Bizopoulos A., Tzallas A. T., Angelidis P. A., Tsalikakis D. G. Denoising simulated EEG signals: A comparative study of EMD, wavelet transform and Kalman filter. *Proceedings of the 13th IEEE International Conference on Bioinformatics and Bioengineering, Chania, Greece, November 10–13, 2013*. – IEEE, 2013. – P. 1–4. https://doi.org/10.1109/BIBE.2013.6701613

74. Saxena D., Kumar J., Singh A., Schmid S. Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud. *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, 2023. P. 1–18. DOI: 10.1109/TPDS.2023.3240567.

75. Shelatkar T., Tondale S., Yadav S., Ahir S. Web traffic time series forecasting using ARIMA and LSTM RNN. *ITM Web of Conferences*, vol. 32, 2020. P. 03017.

76. Shensa M.J. The discrete wavelet transform: wedding the à trous and Mallat algorithms. *IEEE Transactions on Signal Processing*, vol. 40, no. 10, 1992. P. 2464–2482.

77. Shooman M. L. Reliability of Computer Systems and Networks. – New York: *John Wiley & Sons, Inc.*, 2002. – 528 p.

78. Yao, S. P., Hu, C. Z., & Zheng, L. (2006). Research on combination prediction of Web traffic based on wavelets. *Zhongguo Kuangye Daxue Xuebao/Journal of China University of Mining and Technology*, *35*(4), 540-544.

79. Stark H.-G. Wavelets and Signal Processing: An Application-Based Introduction. – Springer Science & Business Media, 2005. – 150 p.

80. Steinbuch M., van de Molengraft M. J. G. Wavelet Theory and Applications: A Literature Study. – Eindhoven: *Eindhoven University of Technology*, 2005. – 39 c.

81. Tambe V., Golait A., Pardeshi S., Javheri R., Arsalwad P. G. Web traffic time series forecasting using ARIMA model. *International Journal of Research in Applied Science and Engineering Technology*, vol. 10, no. 5, 2022. P. 2447–2453.

82. Tedjopurnomo D.A., Bao Z., Zheng B., Choudhury F., Qin A.K. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2020. P. 1–1.

83. Tereikovska L. A., Radchenko K. O., Kharlamov K. O. The application of discrete wavelet transform for forecasting the operational load of the Web server of the distance education system. *The Eighth World Congress "Aviation in the XXI-st Century" – «Safety in Aviation and Space Technologies»*. – Kyiv Aviation University, 10–12 October 2018. – Pp. 3.2.13–3.2.17.

84. Tereykovska L., Tereykovskiy I., Aytkhozhaeva E., Tynymbayev S., Imanbayev A. Encoding of neural network model exit signal, that is devoted for distinction of graphical images in biometric authenticate systems. *News of the National Academy of Sciences of the Republic of Kazakhstan. Series of Geology and Technical Sciences*. – 2017. – Vol. 6, № 426. – P. 217–224.

85. Tian Z. Network traffic prediction method based on wavelet transform and multiple models fusion. *International Journal of Communication Systems*, vol. 33, 2020. Article e4415. DOI: 10.1002/dac.4415.

86. Time Series (ARIMA-SARIMA-Prophet-RNN-LSTM-GRU). *Kaggle*. [Online]. Available: https://www.kaggle.com/code/humagonen/time-series-arima-sarima-prophet-rnn-lstm-gru

87. Top website traffic sources. [Online]. Available: https://www.oberlo.com/statistics/ecommerce-traffic-sources

88. Ulanovs P., Petersons E. "Modeling of Self-similar traffic in high-performance Computer Networks," *RTU, Report Collection for the 42nd International Conference*, Riga, 2001.

89. Vidakovic B. Statistical Modeling by Wavelets. Wiley Series in Probability and Statistics. – New York: *John Wiley & Sons Inc.*, 1999. – 365 p.

90. Wang H.-K., Zhang X., Long H., Yao S., Zhu P. W-FENet: Wavelet-based Fourier-Enhanced Network Model Decomposition for Multivariate Long-Term Time-Series Forecasting. *Neural Processing Letters*, vol. 56, 2024. P. 1–18. DOI: 10.1007/s11063-024-11478-3.

91. Wang, Y., Guo, P., Ma, N., & Liu, G. Robust Wavelet Transform Neural-Network-Based Short-Term Load Forecasting for Power Distribution Networks. *Sustainability*, 2023, *15*(1), 296. https://doi.org/10.3390/su15010296.

92. Wang Y., Sun S., Fathi G., Eslami M. Improving the method of short-term forecasting of electric load in distribution networks using wavelet transform combined with ridgelet neural network optimized by self-adapted Kho-Kho optimization algorithm. *Heliyon*. – 2024. – Vol. 10, No. 7. DOI: 10.1016/j.heliyon.2024.e28381.

93. Wang Zimin, Tan Yonghong. Research of Wavelet Neural Network Based Host Intrusion Detection Systems. *Wavelet Active Media Technology and Information Processing*. – World Scientific Publishing Company, 2006. – P. 1007–1012. DOI: 10.1142/9789812772763_0152.

94. Xu Y., Li Q., Meng S. "Self-similarity Analysis and Application of Network Traffic," in *Mobile Computing, Applications, and Services (MobiCASE 2019)*, Lecture

Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 290. Cham: Springer, 2019. DOI: 10.1007/978-3-030-28468-8_9.

95. Yang Z. Research on server load prediction based on wavelet packet theory. *Proceedings of the 2007 First IEEE International Symposium on Information Technologies and Applications in Education*. – 2007. – P. 610–613.

96. Yao S., Hu C., Peng W. Server Load Prediction Based on Wavelet Packet and Support Vector Regression. // *Proceedings of the 2006 International Conference on Computational Intelligence and Security*, 2006, Vol. 2, pp. 1016–1019.

97. Yao S. J., Song Y. H., Zhang L. Z., Cheng X. Y. Wavelet transform and neural networks for short-term electrical load forecasting. *Energy Conversion and Management*. – 2000. – Vol. 41, № 18. – P. 1975–1988. DOI: 10.1016/S0196-8904(00)00035-2

98. Zhang, R., & Yu, Z. Research on power load forecasting of wavelet neural network based on the improved genetic algorithm. *International Journal of Ambient Energy*, 2019, 43(1), 1036–1040. https://doi.org/10.1080/01430750.2019.1682042

99. Zhenghui Y., Shuaimin W., Yujing X., Bei L. "The Application of Wavelet Neural Network in the Settlement Monitoring of Subway," *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 15, no. 1, pp. 109–114, 2017. DOI: 10.12928/TELKOMNIKA.v15i1.4310.

100. Бессараб В. І., Ігнатенко Е. Г., Червінський В. В. Генератор самоподібного трафіку для моделей інформаційних мереж // *Наукові праці Донецького нац. техн. ун-ту.* – Vol. 15(130) of Обчислювальна техніка та автоматизація. – Донецьк, 2008. – Р. 23–29.

101. Бєльков Д. В. Дослідження мережевого трафіку // *Наукові праці Донецького нац. техн. ун-ту*. – Vol. 10(153) of Обчислювальна техніка та автоматизація. – Донецьк, 2009. – Р. 212–215.

102. Дичка, І., Радченко, К., Терейковський, І., & Терейковська, Л. (2024). Концептуальна модель процесу прогнозування навантаження на вебсервер. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (54), 74-83. https://doi.org/10.36910/6775-2524-0560-2024-54-09.

103. Дичка І., Терейковський І., Терейковська Л., Радченко К. О. Метод визначення ефективного типу базисного вейвлету для застосування в шаблонах нормальної поведінки вебсервера. *Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні*. – 2018. – № 2 (36). – С. 46–55.

104. Довлад О. А. Дослідження та розробка моделі процесу атаки та трафіку локальної мережі. *Захист інформації*. – 2009. – № 1. – С. 83–86.

105. Дронюк І. М. Прогнозування трафіку комп'ютерних мереж для підвищення ефективності використання мережевого обладнання / Дронюк І.М., Федевич О.Ю. // *Наук. вісн. НЛТУ України*. – 2015. – Вип.25.5. – С.301-307.

106. Міхайленко В. М., Терейковська Л. О., Терейковський І. А., Ахметов Б. Б. Нейромережеві моделі та методи розпізнавання фонем в голосовому сигналі в системі дистанційного навчання: монографія. *Київ: ЦП «Компринт»*, 2017. 252 с. ISBN 978-966-929-430-2.

107. Нормативний документ системи технічного захисту інформації (НД ТЗІ) 2.5-010-03 (2003) «Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу». Затверджено наказом Департамент спеціальних телекомунікаційних систем та захисту інформації СБУ (наказ № 33 від 02.04.2003; зміна № 1 – наказ Адміністрація Державної служби спеціального зв'язку та захисту інформації України № 806 від 28.12.2012). Київ: *Держспецзв'язку,* 2003. URL: https://tzi.com.ua/downloads/2.5-010-03.pdf (дата звернення 17.08.2023).

108. Платов В. В., Петров В. В. Дослідження самоподібної структури телетрафіку бездротової мережі // *Електротехнічні та інформаційні комплекси і системи*. – Vol. 3. – 2004. – P. 38–49.

109. Половенко К. Г. Маштабний аналіз електроенцефалограм на основі вейвлет-перетворень з базисною функцією Добеші. *Прикладна радіоелектроніка: науково-технічний журнал*. – 2011. – Т. 10, № 1. – С. 15–21.

110. Радченко К. О. Виявлення різких змін рівня вебтрафіку за допомогою вейвлет-перетворень Добеші. *Кібербезпека державних інституцій та подолання кризових станів: матеріали II Міжнародної науково-практичної конференції: у 2 т.* – Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2023. – Т. 1. – С. 190.

111. Радченко К. О. Застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер комп'ютерних мереж загального призначення. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. – Луцьк. – № 45. – 2021. – С. 90 – 96. DOI: https://doi.org/10.36910/6775-2524-0560-2021-45-13.

112. Радченко К.О. Комп'ютерна програма «Wavelet analysis» / Терейковський Ігор Анатолійович; Радченко Костянтин Олександрович; Дичка Іван Андрійович; Романкевич Віталій Олексійович; Тарасенко-Клятченко Оксана Володимирівна – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» // *Свідоцтво про реєстрацію авторського права на твір №7089 від 15.04.2024р.* https://sis.nipo.gov.ua/en/search/detail/1803492/ Опубліковано 31.05.2024, бюл. №81.

113. Радченко К. О. Концептуальна модель забезпечення ефективності прогнозування навантаження на вебсервер. *Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки*. – 2020. – Т. 31 (70), № 6. – С. 135–141.

114. Радченко К. О. Особливості архітектури нейромережевої моделі розпізнавання кібератак. *Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах: Матеріали VII Міжнародної науково-практичної конференції*. – Чернівці: Місто, 2018. – С. 149–150.

115. Радченко К. О. Особливості прогнозування рівня вебтрафіку у комп'ютерних мережах загального призначення. *Проблеми інформатизації та управління*. – Національний авіаційний університет – № 3(71). – 2022. – С. 41 – 50. DOI: https://doi.org/10.18372/2073-4751.71.17002.

116. Радченко К. О. Прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень Добеші. *Інтегровані інтелектуальні робототехнічні комплекси (ІІРТК-2023): матеріали XVI Міжнародної науково-практичної конференції* (23–24 травня 2023 р., Київ, Україна). – Київ: НАУ, 2023. – С. 335–336.

117. Радченко, К., & Романенко, М. (2024). Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (56), 261-268. https://doi.org/10.36910/6775-2524-0560-2024-56-33.

118. Радченко К. О., Терейковська Л. О. Аналіз самоподібності рівня вебтрафіку у комп'ютерних мережах загального призначення. *Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 25 листопада 2022. – Київ: НУХТ, 2022. – С. 112–113. – Режим доступу: https://nuft.edu.ua/naukova-diyalnist/naukovi-konferencii

119. Радченко, К. О., & Терейковський, І. А. (2025). Метод прогнозування навантаження на вебсервер з урахуванням вейвлет-аналізу веблогів та вебтрафіку. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (60), 247-259. https://doi.org/10.36910/6775-2524-0560-2025-60-27.

120. Радченко К. О., Терейковський О. І., Харламов К. О. Концепція прогнозування навантаженості вебсерверів за допомогою вейвлет-перетворень. *Сучасні інформаційні технології та кібербезпека: матеріали науково-практичної конференції*. – Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. – С. 205–207.

121. Романкевич В. О., Радченко К. О., Романенко М. В. Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж. *XVII Науково-практична конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2024)* (м. Київ, НТУУ «КПІ ім. Ігоря Сікорського», 20–22 листопада 2024 р.). – С. 424–428.

122. Ситніков В.С., Біленко А.О. Класифікація вейвлет-функцій. *Праці Одеського політехнічного університета*. – 2008. – Вип. 1 (29). – С. 168–171.

123. Стіренко С. Г., Радченко К. О., Апанасенко В. П. Прогнозування рівня навантаження на вебсервер комп'ютерних мереж загального призначення. *Кібербезпека державних інституцій та подолання кризових станів: матеріали І*

Міжнародної науково-практичної конференції (Київ – Вроцлав, 26 травня 2022 р.). – Київ, 2022. – С. 59–60.

124. Струбицький Р. П. Самоподібна модель завантаженості хмаркових сховищ даних // *Вісник Національного університету «Львівська політехніка»*. – 2015. – №814. – С. 147–156.

125. Федевич О. Ю. Інформаційна технологія аналізу та прогнозування трафіку в комп'ютерних мережах : дис. … канд. техн. наук (доктора філософії) за спеціальністю 05.13.06 «Інформаційні технології» (122 – Комп'ютерні науки). Львів : *Національний університет «Львівська політехніка» МОН України*, 2018. 214 с.URL: https://lpnu.ua/sites/default/files/2020/dissertation/1651/disfedevychoyu.pdf

126. Щербовських С. В. Математичні моделі та методи для визначення характеристик надійності багатотермінальних систем із урахуванням перерозподілу навантаження: монографія. – Львів: *Видавництво Львівської політехніки*, 2012. – 296 с.

ДОДАТКИ

ДОДАТОК А.

Зразок датасету рівня вебтрафіку досліджуваного вебсервера

(https://drive.google.com/file/d/1bqym5bB5Za6H9jpc970yETkAzrrx723S/view)

Date	Timestamp	Webtraffic(MB)	14.06.2021 3:40	1623642000	1.433616638
14.06.2021 0:00	1623628800	1.205705452	14.06.2021 3:45	1623642300	1.267709351
14.06.2021 0:05	1623629100	1.180924606	14.06.2021 3:50	1623642600	1.395872498
14.06.2021 0:10	1623629400	1.096587753	14.06.2021 3:55	1623642900	1.449732018
14.06.2021 0:15	1623629700	1.257685852	14.06.2021 4:00	1623643200	1.376338577
14.06.2021 0:20	1623630000	1.093921471	14.06.2021 4:05	1623643500	1.435079575
14.06.2021 0:25	1623630300	0.994774055	14.06.2021 4:10	1623643800	1.382359695
14.06.2021 0:30	1623630600	1.177567482	14.06.2021 4:15	1623644100	1.316936302
14.06.2021 0:35	1623630900	1.304357719	14.06.2021 4:20	1623644400	1.358393097
14.06.2021 0:40	1623631200	1.163511467	14.06.2021 4:25	1623644700	1.286278152
14.06.2021 0:45	1623631500	1.193556976	14.06.2021 4:30	1623645000	1.390120888
14.06.2021 0:50	1623631800	1.104599953	14.06.2021 4:35	1623645300	1.336264229
14.06.2021 0:55	1623632100	1.167625237	14.06.2021 4:40	1623645600	1.39352932
14.06.2021 1:00	1623632400	1.202794838	14.06.2021 4:45	1623645900	1.43856945
14.06.2021 1:05	1623632700	1.024302101	14.06.2021 4:50	1623646200	1.332036972
14.06.2021 1:10	1623633000	1.201189232	14.06.2021 4:55	1623646500	1.396806145
14.06.2021 1:15	1623633300	1.147438431	14.06.2021 5:00	1623646800	1.43928566
14.06.2021 1:20	1623633600	1.152681351	14.06.2021 5:05	1623647100	1.318328285
14.06.2021 1:25	1623633900	1.13115387	14.06.2021 5:10	1623647400	1.397824287
14.06.2021 1:30	1623634200	1.17234478	14.06.2021 5:15	1623647700	1.441786194
14.06.2021 1:35	1623634500	1.132295227	14.06.2021 5:20	1623648000	1.324548531
14.06.2021 1:40	1623634800	1.199741745	14.06.2021 5:25	1623648300	1.401539612
14.06.2021 1:45	1623635100	1.03386097	14.06.2021 5:30	1623648600	1.53822937
14.06.2021 1:50	1623635400	1.156690598	14.06.2021 5:35	1623648900	1.378533554
14.06.2021 1:55	1623635700	1.193185234	14.06.2021 5:40	1623649200	1.422281456
14.06.2021 2:00	1623636000	1.442169571	14.06.2021 5:45	1623649500	1.434975815
14.06.2021 2:05	1623636300	1.374432373	14.06.2021 5:50	1623649800	1.331056213
14.06.2021 2:10	1623636600	1.480179977	14.06.2021 5:55	1623650100	1.387968063
14.06.2021 2:15	1623636900	1.294621086	14.06.2021 6:00	1623650400	1.398769569
14.06.2021 2:20	1623637200	1.396232224	14.06.2021 6:05	1623650700	1.327676392
14.06.2021 2:25	1623637500	1.423329926	14.06.2021 6:10	1623651000	1.306581497
14.06.2021 2:30	1623637800	1.367847061	14.06.2021 6:15	1623651300	1.490525055
14.06.2021 2:35	1623638100	1.41889019	14.06.2021 6:20	1623651600	1.779421616
14.06.2021 2:40	1623638400	1.438035393	14.06.2021 6:25	1623651900	1.53763504
14.06.2021 2:45	1623638700	1.281255722	14.06.2021 6:30	1623652200	1.580422783
14.06.2021 2:50	1623639000	1.445812035	14.06.2021 6:35	1623652500	1.48497448
14.06.2021 2:55	1623639300	1.442607689	14.06.2021 6:40	1623652800	1.234474945
14.06.2021 3:00	1623639600	1.347432899	14.06.2021 6:45	1623653100	1.099517632
14.06.2021 3:05	1623639900	1.361339378	14.06.2021 6:50	1623653400	1.542387962
14.06.2021 3:10	1623640200	1.443802452	14.06.2021 6:55	1623653700	1.264929199
14.06.2021 3:15	1623640500	1.303852463	14.06.2021 7:00	1623654000	1.113230705
14.06.2021 3:20	1623640800	1.432572746	14.06.2021 7:05	1623654300	0.838869476
14.06.2021 3:25	1623641100	1.441644096	14.06.2021 7:10	1623654600	1.259298706
14.06.2021 3:30	1623641400	1.355383492	14.06.2021 7:15	1623654900	1.587325096
14.06.2021 3:35	1623641700	1.424036789	14.06.2021 7:20	1623655200	1.068800735

14.06.2021 7:25	1623655500	1.251526451	14.06.2021 11:35	1623670500	4.42558918
14.06.2021 7:30	1623655800	0.930133438	14.06.2021 11:40	1623670800	5.310858536
14.06.2021 7:35	1623656100	0.854627228	14.06.2021 11:45	1623671100	5.449382973
14.06.2021 7:40	1623656400	1.02440052	14.06.2021 11:50	1623671400	5.331970024
14.06.2021 7:45	1623656700	1.257613564	14.06.2021 11:55	1623671700	3.659396553
14.06.2021 7:50	1623657000	1.591485786	14.06.2021 12:00	1623672000	3.195007706
14.06.2021 7:55	1623657300	1.38200531	14.06.2021 12:05	1623672300	2.948843575
14.06.2021 8:00	1623657600	2.158867455	14.06.2021 12:10	1623672600	3.750720406
14.06.2021 8:05	1623657900	2.513351822	14.06.2021 12:15	1623672900	3.571737862
14.06.2021 8:10	1623658200	3.003773689	14.06.2021 12:20	1623673200	3.153442001
14.06.2021 8:15	1623658500	3.433919716	14.06.2021 12:25	1623673500	3.589443207
14.06.2021 8:20	1623658800	2.316404343	14.06.2021 12:30	1623673800	3.088368225
14.06.2021 8:25	1623659100	2.966391373	14.06.2021 12:35	1623674100	2.717049217
14.06.2021 8:30	1623659400	2.814634514	14.06.2021 12:40	1623674400	2.890903473
14.06.2021 8:35	1623659700	2.193207359	14.06.2021 12:45	1623674700	3.075610352
14.06.2021 8:40	1623660000	2.42926712	14.06.2021 12:50	1623675000	3.411304665
14.06.2021 8:45	1623660300	2.162176323	14.06.2021 12:55	1623675300	3.028695297
14.06.2021 8:50	1623660600	2.666978264	14.06.2021 13:00	1623675600	3.458883286
14.06.2021 8:55	1623660900	2.133832741	14.06.2021 13:05	1623675900	2.929063034
14.06.2021 9:00	1623661200	2.300209808	14.06.2021 13:10	1623676200	4.630477715
14.06.2021 9:05	1623661500	3.184263039	14.06.2021 13:15	1623676500	4.433744431
14.06.2021 9:10	1623661800	4.226577377	14.06.2021 13:20	1623676800	3.961824989
14.06.2021 9:15	1623662100	3.650822258	14.06.2021 13:25	1623677100	4.285827637
14.06.2021 9:20	1623662400	4.279533005	14.06.2021 13:30	1623677400	4.503132248
14.06.2021 9:25	1623662700	3.966377449	14.06.2021 13:35	1623677700	4.462502289
14.06.2021 9:30	1623663000	3.913040543	14.06.2021 13:40	1623678000	4.403263283
14.06.2021 9:35	1623663300	4.268335342	14.06.2021 13:45	1623678300	4.512287903
14.06.2021 9:40	1623663600	3.513494301	14.06.2021 13:50	1623678600	3.407949448
14.06.2021 9:45	1623663900	3.857548523	14.06.2021 13:55	1623678900	4.113834763
14.06.2021 9:50	1623664200	3.761440468	14.06.2021 14:00	1623679200	4.006119728
14.06.2021 9:55	1623664500	3.283551979	14.06.2021 14:05	1623679500	4.369997215
14.06.2021 10:00	1623664800	2.856668091	14.06.2021 14:10	1623679800	3.906344986
14.06.2021 10:05	1623665100	3.205405998	14.06.2021 14:15	1623680100	3.795233536
14.06.2021 10:10	1623665400	2.60015049	14.06.2021 14:20	1623680400	3.446879768
14.06.2021 10:15	1623665700	3.205682182	14.06.2021 14:25	1623680700	4.076053238
14.06.2021 10:20	1623666000	2.80664978	14.06.2021 14:30	1623681000	5.068473434
14.06.2021 10:25	1623666300	2.372478867	14.06.2021 14:35	1623681300	4.68880291
14.06.2021 10:30	1623666600	2.634030151	14.06.2021 14:40	1623681600	4.336859894
14.06.2021 10:35	1623666900	3.224623489	14.06.2021 14:45	1623681900	4.097542191
14.06.2021 10:40	1623667200	2.701955795	14.06.2021 14:50	1623682200	3.46604023
14.06.2021 10:45	1623667500	2.609527016	14.06.2021 14:55	1623682500	3.987015724
14.06.2021 10:50	1623667800	2.644800758	14.06.2021 15:00	1623682800	4.080296898
14.06.2021 10:55	1623668100	2.486241913	14.06.2021 15:05	1623683100	3.589183807
14.06.2021 11:00	1623668400	2.352973175	14.06.2021 15:10	1623683400	3.254489708
14.06.2021 11:05	1623668700	2.732942772	14.06.2021 15:15	1623683700	2.992959976
14.06.2021 11:10	1623669000	2.53188324	14.06.2021 15:20	1623684000	2.482289124
14.06.2021 11:15	1623669300	2.580596352	14.06.2021 15:25	1623684300	3.223559761
14.06.2021 11:20	1623669600	2.58340435	14.06.2021 15:30	1623684600	2.988635063
14.06.2021 11:25	1623669900	2.417683601	14.06.2021 15:35	1623684900	3.105148506
14.06.2021 11:30	1623670200	2.585598183	14.06.2021 15:40	1623685200	3.207154846

14.06.2021 15:45	1623685500	4.146308136
14.06.2021 15:50	1623685800	3.554886818
14.06.2021 15:55	1623686100	2.46592865
14.06.2021 16:00	1623686400	2.597849846
14.06.2021 16:05	1623686700	3.21415081
14.06.2021 16:10	1623687000	2.977690697
14.06.2021 16:15	1623687300	3.173488998
14.06.2021 16:20	1623687600	2.731225395
14.06.2021 16:25	1623687900	3.091866875
14.06.2021 16:30	1623688200	3.80588131
14.06.2021 16:35	1623688500	2.863464165
14.06.2021 16:40	1623688800	3.595031929
14.06.2021 16:45	1623689100	2.080194855
14.06.2021 16:50	1623689400	1.950969315
14.06.2021 16:55	1623689700	1.605887413
14.06.2021 17:00	1623690000	1.176534843
14.06.2021 17:05	1623690300	1.442082596
14.06.2021 17:10	1623690600	1.478193283
14.06.2021 17:15	1623690900	1.369521141
14.06.2021 17:20	1623691200	1.265170479
14.06.2021 17:25	1623691500	1.226150322
14.06.2021 17:30	1623691800	1.226911926
14.06.2021 17:35	1623692100	1.389153862
14.06.2021 17:40	1623692400	2.206658363
14.06.2021 17:45	1623692700	2.03385582
14.06.2021 17:50	1623693000	2.101385307
14.06.2021 17:55	1623693300	2.433978271
14.06.2021 18:00	1623693600	2.340302849
14.06.2021 18:05	1623693900	2.177408218
14.06.2021 18:10	1623694200	2.358386993
14.06.2021 18:15	1623694500	2.173216248
14.06.2021 18:20	1623694800	2.021664429
14.06.2021 18:25	1623695100	1.76886158
14.06.2021 18:30	1623695400	1.809002876
14.06.2021 18:35	1623695700	2.029360008
14.06.2021 18:40	1623696000	1.945454979
14.06.2021 18:45	1623696300	1.844679832
14.06.2021 18:50	1623696600	1.047297668
14.06.2021 18:55	1623696900	1.192791748
14.06.2021 19:00	1623697200	1.536259079
14.06.2021 19:05	1623697500	1.523658371
14.06.2021 19:10	1623697800	1.755470276
14.06.2021 19:15	1623698100	0.813552475
14.06.2021 19:20	1623698400	0.799451256
14.06.2021 19:25	1623698700	0.631140137
14.06.2021 19:30	1623699000	1.056395149
14.06.2021 19:35	1623699300	0.946650696
14.06.2021 19:40	1623699600	0.913947487
14.06.2021 19:45	1623699900	1.272117233
14.06.2021 19:50	1623700200	0.893605614
14.06.2021 19:55	1623700500	0.963715935
14.06.2021 20:00	1623700800	1.000683403
14.06.2021 20:05	1623701100	0.935525894
14.06.2021 20:10	1623701400	0.967839813
14.06.2021 20:15	1623701700	0.959128571
14.06.2021 20:20	1623702000	1.103359985
14.06.2021 20:25	1623702300	0.961123657
14.06.2021 20:30	1623702600	0.983175468
14.06.2021 20:35	1623702900	1.601000214
14.06.2021 20:40	1623703200	1.592256546
14.06.2021 20:45	1623703500	1.511079407
14.06.2021 20:50	1623703800	1.332746315
14.06.2021 20:55	1623704100	1.27052021
14.06.2021 21:00	1623704400	1.304493523
14.06.2021 21:05	1623704700	0.885404968
14.06.2021 21:10	1623705000	0.767733002
14.06.2021 21:15	1623705300	0.802919197
14.06.2021 21:20	1623705600	0.790878868
14.06.2021 21:25	1623705900	0.984749603
14.06.2021 21:30	1623706200	0.903432655
14.06.2021 21:35	1623706500	0.798027611
14.06.2021 21:40	1623706800	0.786052132
14.06.2021 21:45	1623707100	1.76562767
14.06.2021 21:50	1623707400	1.251860237
14.06.2021 21:55	1623707700	1.033739281
14.06.2021 22:00	1623708000	1.074730873
14.06.2021 22:05	1623708300	1.050585175
14.06.2021 22:10	1623708600	1.04012928
14.06.2021 22:15	1623708900	1.097132301
14.06.2021 22:20	1623709200	1.15383873
14.06.2021 22:25	1623709500	1.060460472
14.06.2021 22:30	1623709800	1.213525009
14.06.2021 22:35	1623710100	1.044508743
14.06.2021 22:40	1623710400	1.04331913
14.06.2021 22:45	1623710700	1.141656113
14.06.2021 22:50	1623711000	1.101067734
14.06.2021 22:55	1623711300	1.231540489
14.06.2021 23:00	1623711600	1.245638657
14.06.2021 23:05	1623711900	1.110489464
14.06.2021 23:10	1623712200	1.092486572
14.06.2021 23:15	1623712500	1.145413208
14.06.2021 23:20	1623712800	1.348812675
14.06.2021 23:25	1623713100	1.351627922
14.06.2021 23:30	1623713400	1.371600151
14.06.2021 23:35	1623713700	1.305108643
14.06.2021 23:40	1623714000	1.376770973
14.06.2021 23:45	1623714300	1.161617661
14.06.2021 23:50	1623714600	1.048313713
14.06.2021 23:55	1623714900	1.082486343

ДОДАТОК Б.

Лістинг коду методу визначення найефективнішого типу материнського вейвлету

https://colab.research.google.com/drive/1PKn8BImek5B14grkCEdw7LadjnL1wnR1

```
import pandas as pd
import pywt
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import time
from sklearn.preprocessing import minmax_scale

df = pd.read_csv(url, sep=';', engine='python', on_bad_lines='skip')
date_time = pd.to_datetime(df['Timestamp'], unit='s')
#WebTraffic = df['WebTraffic(MB)']
WebTraffic = df.iloc[:, 2]

def zscore_wavelet(x,z_thresh=9.0, replace_with='median'):
    x_clean = x.copy()
    mean_x = np.mean(x_clean)
    std_x = np.std(x_clean)
    z_scores = (x_clean - mean_x) / (std_x + 1e-12)
    if replace_with == 'median':
        med = np.median(x_clean)
        print(med)
        x_clean[np.abs(z_scores) > z_thresh] = med
    elif replace_with == 'clamp':
        upper = mean_x + z_thresh * std_x
        lower = mean_x - z_thresh * std_x
        print(upper, lower)
        x_clean = np.clip(x_clean, lower, upper)
    else:
        raise ValueError("replace_with must be 'median' or 'clamp'")
    x_ref = x if len(x) == len(x_clean) else x_clean
    return x_clean
# Обчислення
x_score = zscore_wavelet(WebTraffic)
print(x_score.describe())
# Побудова графіка
plt.figure(figsize=(12,6))
plt.plot(date_time, WebTraffic, label='Web Traffic (MB)', color='blue', alpha=0.6)
plt.plot(date_time, df['MovingAvg'], label='Moving Average (10 samples)',
color='red', linewidth=1)
plt.plot(date_time, x_score, label='Z-score', color='green', linewidth=2)
plt.title('Web Server Load, Moving Average and Z-Score')
plt.xlabel('Time')
plt.ylabel('Traffic, MB')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

WebTraffic = x_score
# ========================
# Етап 1: Збір та попередня обробка даних
# ========================
def preprocess_data(X, L= None, M= None):
    """ L, X, M - вхідні дані: L - рівень трафіку, X - часові ряди, M - додаткові
параметри     Повертає: Γ, Φ, Δ, X_tilde"""
```

```
    # Приклад нормалізації даних
    X_tilde = (X - np.mean(X)) / np.std(X)
    Γ = np.var(X_tilde)
    Φ = np.mean(X_tilde)
    Δ = np.max(X_tilde) - np.min(X_tilde)
    return X_tilde#, Γ, Φ, Δ

#candidate_wavelets =  ['haar', 'db4', 'sym4', 'coif2', 'bior3.5', 'db2',
'coif1']  #
candidate_wavelets = pywt.wavelist(kind='discrete')
##-- Нормалізуємо сигнал і обрізаємо до довжини кратної 2^n
def pad_signal(X, wavelet, level=None):
    import pywt
    max_level = pywt.dwt_max_level(len(X), pywt.Wavelet(wavelet).dec_len)
    if level is None or level > max_level:
        level = max_level
    # Обрізаємо сигнал до кратної 2^level
    target_len = (len(X) // (2**level)) * (2**level)
    print(0,max_level,level,target_len)
    return X[:target_len], level

def select_candidate_wavelets(Phi, X_tilde, Lambda=None):
    """
    Phi, X_tilde - оброблені дані
    Повертає список базисних вейвлетів W_b і їх рейтинг R_q
    """
    W_b = []
    R_q = []
    epsilon_i = []
    for w in candidate_wavelets:
        try:
            coeffs = pywt.wavedec(X_tilde, w)# Декомпозиція
            energy = sum([np.sum(c**2) for c in coeffs])
            W_b.append(w)
            R_q.append(energy)# як критерій ефективності
            # Відновлення сигналу
            X_hat = pywt.waverec(coeffs, w)
        # Якщо відновлений сигнал довший/коротший, обрізаємо до довжини оригіналу
            X_hat = X_hat[:len(X_tilde)]
        # Розрахунок відносної помилки відновлення
        #epsilon = ||X_tilde - X_hat|| / ||X_tilde|| — відносна похибка
відновлення
            epsilon = np.linalg.norm(X_tilde - X_hat) / np.linalg.norm(X_tilde)
            epsilon_i.append(epsilon)
        except Exception as e:
            print(f"Вейвлет {w} не підтримується: {e}")
            epsilon_i.append(np.inf)  # призначаємо нескінченність для невдалих
вейвлетів

        # Вибір найефективнішого вейвлету
    best_idx = np.argmin(epsilon_i)
    w_star_final = candidate_wavelets[best_idx]
    print("Помилки відновлення epsilon_i:", epsilon_i)
    print(best_idx, "Найефективніший материнський вейвлет:", w_star_final)

    return W_b, R_q, epsilon_i
# =========================
# Етап 2: Визначення ефективних дискретних базисних вейвлетів
# =========================
def compute_similarity(x, wavename, level=5):
    """
    Обчислення подібності сигналу x до базисної функції wavelet.
    Повертає число від 0 до 1.
    """
```

```
    wavelet = pywt.Wavelet(wavename)
    try:
        # wavefun може повертати 2 або 3 масиви
        wf = wavelet.wavefun(level=level)
        if len(wf) == 2:
            _, psi = wf
        else:
            _, psi, _ = wf
        # Обмежуємо довжину до мінімальної для обох масивів
        min_len = min(len(psi), len(x))
        if min_len == 0:
            return 0.0  # якщо порожній масив
        similarity = abs(np.corrcoef(psi[:min_len], x[:min_len])[0,1])
    except Exception as e:
        # Якщо щось не спрацювало, повертаємо дефолтне значення
        similarity = 0.0
    # Нормалізація на випадок NaN
    if np.isnan(similarity):
        similarity = 0.0
    return similarity

# --- Крок 1. Оцінка критеріїв для одного вейвлету ---
def evaluate_wavelet_criteria(x, wavename, level=4,noise_std=0.25,
sparsity_thresh=1e-3):
    wavelet = pywt.Wavelet(wavename)
    coeffs = pywt.wavedec(x, wavename, level=level)
    rec = pywt.waverec(coeffs, wavename)
    # --- симетричність переводимо в числовий коефіцієнт ---
    sym = wavelet.symmetry.lower()
    if 'asym' in sym:
        asymmetry = 1.0      # максимально асиметричний
    elif 'near' in sym:
        asymmetry = 0.5
    else:
        asymmetry = 0.0      # симетричний
    compactness = 1 / wavelet.dec_len# r8: коротша фільтр-функція → компактніша
    similarity = compute_similarity(x, wavename, level=5)
    # 1Відносна помилка реконструкції
    #epsilon = ||X_tilde - X_hat|| / ||X_tilde|| — відносна похибка відновлення
    epsilon = np.linalg.norm(x - rec) / (np.linalg.norm(x) + 1e-12)
    reconstruction = 1 - epsilon # r7: точність відновлення
    # 2Енергія компонент
    energies = [np.sum(c**2) for c in coeffs[1:]] if len(coeffs) > 1 else [0.0]

    scalability = 1 / (np.var(energies) + 1e-6) # r6: стабільність енергії між
рівнями

    total_energy = sum([np.sum(c**2) for c in coeffs]) + 1e-12
    energy_ratio = sum(energies) / total_energy  # частка енергії в деталях #r₁ —
Надлишковість, r₉ — Подібність
    #характеристика спектрального розподілу енергії — скільки енергії сигналу
вейвлет зберігає у високочастотній частині.описує інформативність деталей.
    # 3Sparsity
    all_coeffs = np.hstack([c.ravel() for c in coeffs])
    # критерії
    redundancy = 1 - np.count_nonzero(all_coeffs) / len(all_coeffs)#Рахує, яка
частка коефіцієнтів дорівнює нулю.
    #redundancy = 1 - len(coeffs) / (len(x) / 2)          # r1: чим більше
коефіцієнтів, тим більша надлишковість

    #all_coeffs = np.hstack([c.ravel() for c in coeffs])
    sparsity = np.sum(np.abs(all_coeffs) > sparsity_thresh) /
len(all_coeffs)#Рахує, яка частка коефіцієнтів істотно відмінна від нуля, тобто
"активні" коефіцієнти.
```

```
    # Надлишковість: частка нульових коефіцієнтів       redundancy = 1 -
np.count_nonzero(all_coeffs) / len(all_coeffs)
    # Скупість: частка значущих коефіцієнтів             sparsity =
np.sum(np.abs(all_coeffs) > sparsity_thresh) / len(all_coeffs)

    # 4Обчислювальна вартість
    fastness = 1 / wavelet.dec_len

    t0 = time.time()
    _ = pywt.wavedec(x, wavename, level=level)
    empirical_speed = time.time() - t0
    # Комбінований критерій
    alpha=0.5
    tcost = alpha * fastness + (1 - alpha) * empirical_speed

    # 5Зміна SNR при денойзингу
    #ефективність шумозаглушення" або англійською - Noise Reduction Efficiency
(NRE).
    if noise_std > 0:
        x_noisy = x + np.random.normal(0, noise_std, size=len(x))
        coeffs_n = pywt.wavedec(x_noisy, wavename, level=level)
        sigma = noise_std
        thr = sigma * np.sqrt(2 * np.log(len(x)))
        coeffs_thr = [pywt.threshold(c, thr, mode='soft') if i>0 else c for i, c
in enumerate(coeffs_n)]
        x_deno = pywt.waverec(coeffs_thr, wavename)[:len(x)]
        def snr(a,b):
            return 10*np.log10((np.mean(a**2)+1e-12)/(np.mean((a-b)**2)+1e-12))
        snr_gain = snr(x, x_noisy) - snr(x, x_deno)
    else:
        snr_gain = 0.0

    rank = {
        "redundancy": redundancy,
        "fastness": tcost,
        "asymmetry": asymmetry,
        "scalability": scalability,
        "reconstruction": reconstruction,
        "compactness": compactness,
        "similarity": similarity,
        'energy_ratio': energy_ratio,
        'sparsity': sparsity,
        'snr_gain': snr_gain
    }
    return rank, energies

# --- Крок 2. MCDM ранжування ---
def rank_wavelets_MCDM(x, candidates, weights, level=4):
    results = {}
    energies = {}
    for wname in candidates:
        if wname=='haar': continue
        try:
            results[wname], energies[wname] = evaluate_wavelet_criteria(x, wname,
level)
            #print(wname, results[wname])
        except Exception as e:
            print(f"⚠ {wname} error:", e)

    # нормалізація (щоб усі значення були 0-1)
    all_vals = {k: [results[w][k] for w in results] for k in weights.keys()}
    mins = {k: np.min(v) for k,v in all_vals.items()}
    maxs = {k: np.max(v) for k,v in all_vals.items()}
```

```
    normalized = {}
    for wname, crits in results.items():
        normalized[wname] = {k: (crits[k]-mins[k])/(maxs[k]-mins[k]+1e-12) for k
in weights.keys()}

    # інтегральний показник
    scores = {}
    for wname, crits in normalized.items():
        scores[wname] = sum(weights[k]*crits[k] for k in weights.keys())

    ranked = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    return ranked, results, normalized,energies

# --- Крок 3. Використання ---
if __name__ == "__main__":
    # тестовий сигнал
    #t = np.linspace(0, 10, 2000)
    #x = np.sin(0.5*t) + 0.2*np.random.randn(len(t))
    X = WebTraffic.iloc[:2016]

    # кандидати
    candidates = candidate_wavelets #['db2', 'db4', 'sym4', 'coif1', 'bior3.3',
'rbio1.3']

    # ваги критеріїв r1-r9
    weights = {
        "redundancy": 0.05,
        "fastness": 0.10,
        "asymmetry": 0.05,
        "scalability": 0.05,
        "compactness": 0.05,
        'energy_ratio': 0.20,
        'sparsity': 0.10,
        "reconstruction": 0.20,
        'snr_gain': 0.15,
        "similarity": 0.05,
    }
    # ========================
# запуск пайплайну
    # Етап 1 Gamma, Phi, Delta,
    X_tilde = preprocess_data(X)

# Етап 2
   # W_b, R_q, epsilon_i = select_candidate_wavelets(Phi, X_tilde)

    ranked, results, normalized, energies  = rank_wavelets_MCDM(X_tilde,
candidates, weights, level=5)

    #print(normalized)
    print("\nWavelet ranking by integrated efficiency:")
    for w, s in ranked:
        print(f"{w:8s} → score={s:.4f}")

    print("\nTop wavelet:", ranked[0][0])

# --- Крок 3. Побудова інтерактивної Radar Chart ---
def plot_interactive_radar(normalized, top_wavelets, weights):
    labels = list(weights.keys())
    fig = go.Figure()

    for wname in top_wavelets:
        values = [normalized[wname][k] for k in labels]
        values += [values[0]]
        fig.add_trace(go.Scatterpolar(
```

```
            r=values,
            theta=labels + [labels[0]],
            fill='toself',
            name=wname,
            hovertemplate="%{theta}: %{r:.3f}<extra>%{fullData.name}</extra>"
        ))

    fig.update_layout(
        polar=dict(
            radialaxis=dict(visible=True, range=[0, 1], tickfont=dict(size=10)),
        ),
        title="Interactive Radar Chart of Wavelet Evaluation (MCDM)",
        showlegend=True,
        legend=dict(font=dict(size=10))
    )
    fig.show()

# --- Крок 4. Тест ---
# Побудова інтерактивної діаграми для топ-3
top3 = [r[0] for r in ranked[:]]
plot_interactive_radar(normalized, top3, weights)

print(ranked)
print(results)
print(energies)

# Розділяємо імена вейвлетів та їхні інтегральні бали
W_b = [w[0] for w in ranked]           # ['rbio3.1', 'db1', ...]
R = [float(w[1]) for w in ranked]      # [0.676, 0.564, ...]

# Визначаємо найкращий вейвлет
w_star_final = W_b[R.index(max(R))]
#w_star_final = ranked[0][0]

# Перетворення словника у DataFrame
Wb_df = pd.DataFrame(results).T  # .T транспонує, щоб вейвлети були рядками
Wb_df.index.name = 'Wavelet'
Wb_df.reset_index(inplace=True)
#W_b = Wb.index.tolist()   # Список назв вейвлетів

#coeffs = pywt.wavedec(X_tilde, w_star_final)
E_j = energies[w_star_final] # E_j = [np.sum(c**2) for c in C]
print(range(len(E_j)), E_j)

# ==========================
# Візуалізація енергетичного аналізу
# ==========================
plt.figure(figsize=(10,5))
plt.bar(range(len(E_j)), E_j, color='skyblue')
plt.title('Енергетичний внесок коефіцієнтів обраного материнського вейвлету')
plt.xlabel('Номер коефіцієнта')
plt.ylabel('Енергія E_j')
plt.grid(True)
plt.show()

# ==========================
# Порівняння вибору найефективнішого вейвлету
# ==========================
plt.figure(figsize=(10,5))
colors = ['red' if w == w_star_final else 'orange' for w in W_b]
plt.bar(W_b, R, color=colors)
plt.title(f'Рейтинг ефективності базисних вейвлетів\n Найефективніший материнський
вейвлет: {w_star_final}')
plt.xlabel('Вейвлет')
```

```
plt.ylabel('Нормалізована вага R')
plt.xticks(rotation=65, ha='right')  # ha='right' — вирівнювання по правому краю
plt.tight_layout()
plt.show()

n = 2016
X = WebTraffic.iloc[:n]
X_tilde = preprocess_data(X)
X2 = WebTraffic.iloc[1*n:2*n]
X2_tilde = preprocess_data(X2)
import pywt
import numpy as np

def select_optimal_level(x, wavelet_name='db4', max_level=None, tol=1e-12):
# Автоматичний вибір оптимального рівня декомпозиції для сигналу.
    wavelet = pywt.Wavelet(wavelet_name)
    N = len(x)
    if max_level is None:
        max_level = pywt.dwt_max_level(data_len=N, filter_len=wavelet.dec_len)

    prev_eps = None
    metrics = {}
    optimal_level = 1

    for level in range(1, max_level + 1):
        try:
            coeffs = pywt.wavedec(x, wavelet, level=level)
            x_rec = pywt.waverec(coeffs, wavelet)
            x_rec = x_rec[:len(x)]

            eps = np.linalg.norm(x - x_rec) / (np.linalg.norm(x) + 1e-12)
            snr = 10 * np.log10(np.sum(x**2) / (np.sum((x - x_rec)**2) + 1e-12))

            metrics[level] = {'ε': eps, 'SNR': snr}

            if prev_eps is not None:
                delta_eps = abs(eps - prev_eps)
                if delta_eps < tol:
                    optimal_level = level
                    break
            prev_eps = eps

        except Exception as e:
            metrics[level] = {'ε': np.inf, 'SNR': -np.inf}
            continue

    return optimal_level, metrics

opt_level, metrics = select_optimal_level(X_tilde, w_star_final)
print("Оптимальний рівень:", opt_level)
print("Метрики:", metrics)

def verify_wavelet(x, x_ref,w_star_final, candidates, level=4, alpha=0.5,
beta=0.3, gamma=0.2):
    """
    Перевіряє ефективність кандидатів вейвлетів на нових даних
    за комбінованим критерієм L = f(ε, SNR, rank).
    Повертає:
    rank_w : dict
        Ранги матриці коефіцієнтів для кожного вейвлету.
    best_wavelet : str
        Назва найкращого вейвлету.
    L_w : dict
        Значення комбінованого критерію для кожного вейвлету.
```

```
    """
    rank_w = {}
    L_w = {}

    for w in candidates:
        try:
            # Декомпозиція та реконструкція
            coeffs = pywt.wavedec(x, w, level=level)
            x_rec = pywt.waverec(coeffs, w)

            # Відносна помилка
            eps = np.linalg.norm(x_rec[:len(x_ref)] - x_ref) /
(np.linalg.norm(x_ref) + 1e-12)

            # SNR
            snr = 10 * np.log10(np.sum(x_ref**2) / (np.sum((x_rec[:len(x_ref)] -
x_ref)**2) + 1e-12))
            # Ранг матриці коефіцієнтів
            coeff_matrix = np.concatenate([c.flatten() for c in coeffs])
            rank = np.linalg.matrix_rank(coeff_matrix.reshape(-1, 1))
            rank_w[w] = rank

            # Комбінований критерій (чим вище — тим краще)
            # eps обертаємо, бо менша помилка краща
            L = alpha * (1/(eps + 1e-6)) + beta * (snr) + gamma * (rank)
            L_w[w] = L

        except Exception as e:
            rank_w[w] = None
            L_w[w] = -np.inf
            print(f"Помилка при обробці {w}: {e}")
    # Вибір найкращого
    best_wavelet = max(L_w, key=L_w.get)

    return rank_w, best_wavelet, L_w

rank_w, w_star_ver, L_w = verify_wavelet(X2_tilde,X_tilde,
w_star_final,candidates=W_b[:10])
print("Найкращий вейвлет після перевірки:", w_star_ver)
print("Інтегральні показники L(w):")
for w, val in L_w.items():
    print(f"{w}: {val:.4f}")

def apply_wavelet_transform(x, A_J, D_j, w_star):
    """
    Застосування зворотного вейвлет-перетворення та оцінка похибки реконструкції.
    """
    try:
        coeffs = [A_J] + D_j
        X_hat = pywt.waverec(coeffs, w_star)
        # Оцінка помилки відновлення
        n = min(len(X_hat), len(x))
        E_recon = np.linalg.norm(X_hat[:n] - x[:n]) / (np.linalg.norm(x[:n]) + 1e-
12)*100

    except Exception as e:
        print(f"Помилка при реконструкції для {w_star}: {e}")
        X_hat = np.zeros_like(x)
        E_recon = np.nan

    return X_hat, E_recon
```

ДОДАТОК В.

Лістинг коду CLI-скрипта для автоматичного збору загального рівня вебтрафіку сервера Apache

```
#!/usr/bin/env php
<?php
// collect_traffic.php
// PHP CLI script for background collection of Apache web traffic metrics.
// Usage: php collect_traffic.php
// Recommended: run as systemd service or via nohup/cron to keep it in background.
// NOTE: run with PHP CLI and ensure pcntl extension is enabled for graceful
shutdown.

declare(ticks=1);

$cfg = [
    'access_log'   => '/var/log/apache2/access.log',   // шлях
    'iface'        => 'eth0',                          // інтерфейс для мереж.
метрик (або leave blank)
    'interval'     => 5,                              // секунди між зборами
    'state_file'   => '/var/run/web_traffic_collector.state',
    'out_dir'      => '/var/log/web_traffic_metrics',  // директорія для
збереження JSON
    'max_files'    => 240,                             // скільки файлів зберігати
(ротація)
];

// --- підготовка ---
if (!is_dir($cfg['out_dir'])) {
    @mkdir($cfg['out_dir'], 0755, true) or die("Не вдалось створити каталог
{$cfg['out_dir']}\n");
}
$stop = false;
pcntl_signal(SIGTERM, function() use (&$stop){ $stop = true; });
pcntl_signal(SIGINT, function() use (&$stop){ $stop = true; });

// --- стан (offset, inode, prev iface bytes) ---
$state = ['offset' => 0, 'inode' => 0, 'prev_iface_rx' => 0, 'prev_iface_tx' =>
0];
if (file_exists($cfg['state_file'])) {
    $s = @file_get_contents($cfg['state_file']);
    if ($s) {
        $tmp = json_decode($s, true);
        if (is_array($tmp)) $state = array_merge($state, $tmp);
    }
}

// helper: save state
function save_state($path, $state) {
    @file_put_contents($path, json_encode($state));
    @chmod($path, 0644);
}

// helper: read /proc/net/dev for interface bytes
function read_iface_bytes($iface) {
    if (!$iface || !is_readable('/proc/net/dev')) return ['rx' => 0, 'tx' => 0];
    $txt = @file_get_contents('/proc/net/dev');
    if (!$txt) return ['rx' => 0, 'tx' => 0];
    foreach (explode("\n", $txt) as $line) {
        $line = trim($line);
```

```
        if (!$line) continue;
        if (strpos($line, $iface.':') === 0 || preg_match('/^' .
preg_quote($iface, '/') . '\s*:/', $line)) {
            // format: iface: bytes ... bytes
            $parts = preg_split('/\s+/', str_replace(':', ' ', $line));
            // $parts[1] = rx bytes, $parts[9] = tx bytes (typical)
            return ['rx' => (int)($parts[1] ?? 0), 'tx' => (int)($parts[9] ?? 0)];
        }
    }
    return ['rx' => 0, 'tx' => 0];
}

// helper: parse apache combined log (with optional response time at the end)
$logRegex = '/^(\S+) \S+ \S+ \[([^\]]+)\] "(\S+) (.+?) (\S+)" (\d{3}) (\S+)(?:
"([^"]*)" "([^"]*)")?(?: (\d+))?$/';

// main loop
$accessLog = $cfg['access_log'];
$interval = (int)$cfg['interval'];
$outDir = rtrim($cfg['out_dir'], '/');
$iface = $cfg['iface'];

while (!$stop) {
    clearstatcache(true, $accessLog);
    if (!file_exists($accessLog) || !is_readable($accessLog)) {
        // файл логів недоступний — логируем нульові або системні метрики
        $lines = [];
        $currentInode = 0;
    } else {
        $currentInode = fileinode($accessLog);
    }

    // відкриваємо лог (якщо зміна inode -> ротація)
    $fp = false;
    if ($currentInode && is_readable($accessLog)) {
        $fp = @fopen($accessLog, 'r');
        if ($fp === false) {
            // не вдалось відкрити
            $fp = false;
        } else {
            // якщо inode змінився -> лог був ротований; починаємо з 0
            if ($state['inode'] !== $currentInode) {
                $state['inode'] = $currentInode;
                $state['offset'] = 0;
            }
            // перейдемо на offset
            fseek($fp, $state['offset']);
        }
    }

    // агрегати
    $metrics = [
        'timestamp' => time(),
        'interval' => $interval,
        'requests' => 0,
        'bytes' => 0,
        'methods' => [],
        'status' => [],
        'top_urls' => [],
        'avg_bytes_per_req' => 0.0,
    ];

    $urlCounts = [];
```

```
    if ($fp) {
        while (!feof($fp)) {
            $line = fgets($fp);
            if ($line === false) break;
            $line = trim($line);
            if ($line === '') continue;
            $metrics['requests']++;

            if (preg_match($logRegex, $line, $m)) {
                // m captures: 1=ip 2=time 3=method 4=url 5=proto 6=status 7=bytes
8=ref 9=ua 10=opt_time
                $method = $m[3];
                $url = $m[4];
                $status = $m[6];
                $bytes = ($m[7] === '-' ? 0 : (int)$m[7]);

                $metrics['bytes'] += $bytes;
                if (!isset($metrics['methods'][$method]))
$metrics['methods'][$method] = 0;
                $metrics['methods'][$method]++;

                if (!isset($metrics['status'][$status]))
$metrics['status'][$status] = 0;
                $metrics['status'][$status]++;

                $urlCounts[$url] = ($urlCounts[$url] ?? 0) + 1;
            } else {
                // невідповідний рядок — лише приблизно спробуємо розбити
                $parts = preg_split('/\s+/', $line);
                $last = end($parts);
                $b = is_numeric($last) ? (int)$last : 0;
                $metrics['bytes'] += $b;
            }
        }
        // оновимо offset
        $state['offset'] = ftell($fp);
        fclose($fp);
    }

    // вирахуємо top urls
    arsort($urlCounts);
    $metrics['top_urls'] = array_slice($urlCounts, 0, 10, true);
    $metrics['avg_bytes_per_req'] = $metrics['requests'] ? ($metrics['bytes'] /
$metrics['requests']) : 0.0;

    // --- системні метрики ---
    // CPU load (1m,5m,15m)
    $load = sys_getloadavg(); // [1,5,15]
    $metrics['loadavg'] = $load;

    // memory (from /proc/meminfo)
    $meminfo = @file_get_contents('/proc/meminfo');
    if ($meminfo) {
        preg_match('/MemTotal:\s+(\d+)\s+kB/i', $meminfo, $mt);
        preg_match('/MemAvailable:\s+(\d+)\s+kB/i', $meminfo, $ma);
        $memTotal = isset($mt[1]) ? (int)$mt[1] : 0;
        $memAvail = isset($ma[1]) ? (int)$ma[1] : 0;
        $metrics['mem_kb'] = ['total_kb' => $memTotal, 'available_kb' =>
$memAvail];
    }

    // network iface bytes delta
    $ifaceBytes = read_iface_bytes($iface);
    $rx = $ifaceBytes['rx'];
```

```
    $tx = $ifaceBytes['tx'];
    $prevRx = $state['prev_iface_rx'] ?? 0;
    $prevTx = $state['prev_iface_tx'] ?? 0;
    $deltaRx = $prevRx ? max(0, $rx - $prevRx) : 0;
    $deltaTx = $prevTx ? max(0, $tx - $prevTx) : 0;
    $metrics['iface'] = [
        'iface' => $iface,
        'rx_bytes_delta' => $deltaRx,
        'tx_bytes_delta' => $deltaTx,
        'rx_total' => $rx,
        'tx_total' => $tx,
    ];
    $state['prev_iface_rx'] = $rx;
    $state['prev_iface_tx'] = $tx;

    // requests per second (approx)
    $metrics['rps'] = $interval > 0 ? ($metrics['requests'] / $interval) :
$metrics['requests'];

    // --- збереження результату ---
    $filename = $outDir . '/metrics_' . date('Ymd_His', $metrics['timestamp']) .
'.json';
    @file_put_contents($filename, json_encode($metrics, JSON_PRETTY_PRINT |
JSON_UNESCAPED_SLASHES));
    @chmod($filename, 0644);

    // ротація: видалити найстаріші файли, якщо перевищено max_files
    $files = glob($outDir . '/metrics_*.json');
    if ($files && count($files) > $cfg['max_files']) {
        usort($files, function($a,$b){ return filemtime($a) - filemtime($b); });
        $toRemove = count($files) - $cfg['max_files'];
        for ($i=0; $i<$toRemove; $i++) @unlink($files[$i]);
    }

    // зберегти стан
    save_state($cfg['state_file'], $state);

    // чекати
    $slept = 0;
    while ($slept < $interval && !$stop) {
        sleep(1);
        $slept++;
    }
}

// graceful exit
save_state($cfg['state_file'], $state);
echo "Collector stopped\n";
exit(0);
```

ДОДАТОК Г.

Лістинг коду cron-скрипта для автоматизованого запису метрик у MySQL

```
#!/usr/bin/env php
<?php
/**
 * collect_web_traffic_mysql.php
 * Версія для shared-хостингу з записом метрик у MySQL.
 * Метрики: загальна кількість запитів, методи, статуси, HTTP-версії, протоколи,
топ-10 URL.
Підготовка:
Створити базу даних і користувача:
CREATE DATABASE web_traffic CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON web_traffic.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
Додати cron job, наприклад кожні 5 хвилин:
*/5 * * * * /usr/bin/php /home/username/collect_web_traffic_mysql.php
Скрипт автоматично створює таблицю traffic_metrics, якщо її немає.*/

date_default_timezone_set('Europe/Kiev');
// ---------- CONFIGURATION ----------
$cfg = [
    'access_log' => '/home/username/logs/access_log',
    'log_regex'  => '/^(\S+) \S+ \S+ \[([^\]]+)\] "(\S+) (.+?) (HTTP\/[0-9.]+)"
(\d{3}) (\S+)(?: "([^"]*)" "([^"]*)")?$/',
    'db' => [
        'host'     => 'localhost',
        'port'     => 3306,
        'dbname'   => 'web_traffic',
        'user'     => 'username',
        'password' => 'password',
        'table'    => 'traffic_metrics',
    ],
];
// ---------- ПІДГОТОВКА ----------
$metrics = [
    'timestamp'      => time(),
    'total_requests' => 0,
    'total_bytes'    => 0,
    'methods'        => [],
    'status'         => [],
    'http_versions'  => [],
    'protocols'      => ['HTTP'=>0,'HTTPS'=>0,'WebSocket'=>0],
    'top_urls'       => [],
];

$urlCounts = [];

// ---------- АНАЛІЗ ЛОГУ ----------
if (is_readable($cfg['access_log'])) {
    $lines = file($cfg['access_log'], FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);
    foreach ($lines as $line) {
        if (preg_match($cfg['log_regex'], $line, $m)) {
            $method  = $m[3];
            $url     = $m[4];
            $httpver = $m[5];
            $status  = $m[6];
            $bytes   = ($m[7]==='-' ? 0 : (int)$m[7]);
            $metrics['total_requests']++;
```

```
            $metrics['total_bytes'] += $bytes;
            $metrics['methods'][$method] = ($metrics['methods'][$method] ?? 0) +
1;
            $metrics['status'][$status] = ($metrics['status'][$status] ?? 0) + 1;
            $metrics['http_versions'][$httpver] =
($metrics['http_versions'][$httpver] ?? 0) + 1;
            if (strpos($url, 'wss://')!==false || strpos($url, 'ws://')!==false) {
                $metrics['protocols']['WebSocket']++;
            } elseif (strpos($url, 'https://')!==false) {
                $metrics['protocols']['HTTPS']++;
            } else {
                $metrics['protocols']['HTTP']++;
            }

            $urlCounts[$url] = ($urlCounts[$url] ?? 0) + 1;
        }
    }
}
// ---------- ТОП-10 URL ----------
arsort($urlCounts);
$metrics['top_urls'] = array_slice($urlCounts, 0, 10, true);
// ---------- ЗАПИС У MySQL ----------
try {
    $dsn =
"mysql:host={$cfg['db']['host']};port={$cfg['db']['port']};dbname={$cfg['db']['dbn
ame']};charset=utf8mb4";
    $pdo = new PDO($dsn, $cfg['db']['user'], $cfg['db']['password'], [
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
    ]);
    // Створити таблицю, якщо її немає
    $sqlCreate = "CREATE TABLE IF NOT EXISTS `{$cfg['db']['table']}` (
        `id` INT AUTO_INCREMENT PRIMARY KEY,
        `timestamp` INT NOT NULL,
        `total_requests` INT NOT NULL,
        `total_bytes` BIGINT NOT NULL,
        `methods` JSON NOT NULL,
        `status` JSON NOT NULL,
        `http_versions` JSON NOT NULL,
        `protocols` JSON NOT NULL,
        `top_urls` JSON NOT NULL
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;";
    $pdo->exec($sqlCreate);
    // Вставка метрик
    $stmt = $pdo->prepare("INSERT INTO `{$cfg['db']['table']}`
        (timestamp, total_requests, total_bytes, methods, status, http_versions,
protocols, top_urls)
        VALUES (:ts, :req, :bytes, :methods, :status, :httpver, :protocols,
:topurls)");
    $stmt->execute([
        ':ts'        => $metrics['timestamp'],
        ':req'       => $metrics['total_requests'],
        ':bytes'     => $metrics['total_bytes'],
        ':methods'   => json_encode($metrics['methods']),
        ':status'    => json_encode($metrics['status']),
        ':httpver'   => json_encode($metrics['http_versions']),
        ':protocols' => json_encode($metrics['protocols']),
        ':topurls'   => json_encode($metrics['top_urls']),
    ]);
    echo "Метрики успішно збережено у MySQL\n";
} catch (PDOException $e) {
    echo "Помилка підключення або запису у MySQL: " . $e->getMessage() . "\n";
}
```

ДОДАТОК Д.

Лістинг коду методу прогнозування навантаження на вебсервер

https://colab.research.google.com/drive/1at5AvyGmaKP7WWI4YbkfvqmDa4xFgGFV

```
!pip install optuna
import optuna

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import minmax_scale, StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import (Model, Sequential, load_model, save_model,
clone_model)
from tensorflow.keras.layers import (
    Input, LSTM, Dense, Dropout, concatenate, BatchNormalization
)
from prophet import Prophet
from tensorflow.keras.layers import Input, LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam, RMSprop
from sklearn.metrics import (
    mean_absolute_error,
    mean_squared_error,
    mean_absolute_percentage_error,
    r2 score
)
import pywt
import plotly.graph_objects as go
import time
from scipy import stats
from scipy.signal import find_peaks, correlate, medfilt
from scipy.stats import wasserstein_distance, ks_2samp, zscore
from sklearn.metrics.pairwise import cosine_similarity
import os
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from statsmodels.tsa.stattools import adfuller
from google.colab import auth
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
import requests
import json

df = pd.read_csv(url, sep=';', engine='python', on_bad_lines='skip')
date_time = pd.to_datetime(df['Timestamp'], unit='s')

WebTraffic = df.iloc[:, 2]
WebTraffic = pd.to_numeric(
    WebTraffic
    .astype(str)
    .str.replace(',', '.', regex=False)
    .str.extract(r'([\d\.]+)')[0],
    errors='coerce'   # нечислові значення стануть NaN
)

def zscore_wavelet(x,z_thresh=9.0, replace_with='median'):

    x_clean = x.copy()
    mean_x = np.mean(x_clean)
    std_x = np.std(x_clean)
```

```
    z_scores = (x_clean - mean_x) / (std_x + 1e-12)
    if replace_with == 'median':
        med = np.median(x_clean)
        print(med)
        x_clean[np.abs(z_scores) > z_thresh] = med
    elif replace_with == 'clamp':
        upper = mean_x + z_thresh * std_x
        lower = mean_x - z_thresh * std_x
        print(upper, lower)
        x_clean = np.clip(x_clean, lower, upper)
    else:
        raise ValueError("replace_with must be 'median' or 'clamp'")
    x_ref = x if len(x) == len(x_clean) else x_clean
    return x_clean

def distribution_similarity(y_true, y_pred, method='wasserstein'):
    """Схожість розподілів фактичних та прогнозованих значень"""
    if method == 'wasserstein':
        return wasserstein_distance(y_true, y_pred)
    elif method == 'ks':
        stat, _ = ks_2samp(y_true, y_pred)
        return stat
    elif method == 'histogram_corr':
        hist_true, bins = np.histogram(y_true, bins=20, density=True)
        hist_pred, _ = np.histogram(y_pred, bins=bins, density=True)
        return np.corrcoef(hist_true, hist_pred)[0, 1]

def peak_alignment(y_true, y_pred, prominence=0.1):
    """Вирівнювання піків між фактичними та прогнозованими значеннями"""
    peaks_true, _ = find_peaks(y_true, prominence=prominence)
    peaks_pred, _ = find_peaks(y_pred, prominence=prominence)
    if len(peaks_true) == 0 or len(peaks_pred) == 0:
        return 0

    # Знаходимо найближчі піки
    alignment_scores = []
    for peak_true in peaks_true:
        closest_peak = peaks_pred[np.argmin(np.abs(peaks_pred - peak_true))]
        distance = np.abs(closest_peak - peak_true)
        # Нормалізуємо відстань (менше = краще)
        score = max(0, 1 - distance / len(y_true))
        alignment_scores.append(score)
    return np.mean(alignment_scores)

def calculate_phase_shift(y_true, y_pred):
    """Обчислення фазового зсуву між рядами"""
    # Крос-кореляція для знаходження зсуву
    correlation = correlate(y_true - np.mean(y_true),
                            y_pred - np.mean(y_pred),
                            mode='full')

    # Знаходимо максимальну кореляцію
    max_corr_idx = np.argmax(correlation)
    phase_shift = max_corr_idx - (len(y_true) - 1)

    return phase_shift

def seasonality_capture(y_true, y_pred, max_lag=50):
    """Оцінка збереження сезонності"""
    def autocorrelation(x, lag):
        return np.corrcoef(x[:-lag], x[lag:])[0, 1]
    # Автокореляція для різних лагів
    acf_true = [autocorrelation(y_true, lag) for lag in range(1, max_lag)]
    acf_pred = [autocorrelation(y_pred, lag) for lag in range(1, max_lag)]
```

```
    # Схожість автокореляційних функцій
    return cosine_similarity([acf_true], [acf_pred])[0, 0]

def smoothness_ratio(y_true, y_pred):
    """Відношення гладкості прогнозованого ряду до фактичного"""
    def calculate_smoothness(series):
        # Міра гладкості - сума квадратів других різниць
        second_diff = np.diff(series, n=2)
        return np.mean(second_diff ** 2)

    smoothness_true = calculate_smoothness(y_true)
    smoothness_pred = calculate_smoothness(y_pred)
    # Уникаємо ділення на нуль
    if smoothness_true == 0:
        return 1.0 if smoothness_pred == 0 else float('inf')
    return smoothness_pred / smoothness_true

def trend_accuracy(y_true, y_pred, window=5):
    """Точність відтворення трендів"""
    def calculate_trends(series, window):
        trends = []
        for i in range(len(series) - window):
            segment = series[i:i+window]
            slope = (segment[-1] - segment[0]) / (window - 1)
            trends.append(1 if slope > 0 else (-1 if slope < 0 else 0))
        return np.array(trends)

    trends_true = calculate_trends(y_true, window)
    trends_pred = calculate_trends(y_pred, window)

    # Точність напрямків трендів
    trend_accuracy = np.mean(trends_true == trends_pred)
    return trend_accuracy

def outlier_preservation(y_true, y_pred, threshold=2):
    #    Збереження викидів та екстремальних значень
    # Визначаємо викиди за стандартним відхиленням
    mean_true, std_true = np.mean(y_true), np.std(y_true)
    outliers_true = np.abs(y_true - mean_true) > threshold * std_true

    if not np.any(outliers_true):
        return 1.0  # Якщо викидів немає

    # Наскільки модель відтворює викиди
    outlier_errors = np.abs(y_pred[outliers_true] - y_true[outliers_true])
    outlier_mape = np.mean(outlier_errors / (np.abs(y_true[outliers_true]) + 1e-
8))

    return 1 - min(outlier_mape, 1)  # Нормалізуємо до [0, 1]

def visual_metrics(y_true, y_pred, include_plots=False):
    """Комплексні метрики для візуальної оцінки якості прогнозу"""
    metrics = {
        # Форма розподілу
        'Distribution_Similarity_Wasserstein': distribution_similarity(y_true,
y_pred, 'wasserstein'),
        'Distribution_Similarity_KS': distribution_similarity(y_true, y_pred,
'ks'),
        'Distribution_Similarity_Corr': distribution_similarity(y_true, y_pred,
'histogram_corr'),

        # Структурні характеристики
        'Peak_Alignment': peak_alignment(y_true, y_pred),
        'Phase_Shift': calculate_phase_shift(y_true, y_pred),
```

```
        'Seasonality_Capture': seasonality_capture(y_true, y_pred),
        'Trend_Accuracy': trend_accuracy(y_true, y_pred),
        # Якість кривої
        'Smoothness_Ratio': smoothness_ratio(y_true, y_pred),
        'Outlier_Preservation': outlier_preservation(y_true, y_pred),
        # Додаткові метрики
        'Shape_Correlation': np.corrcoef(y_true, y_pred)[0, 1],
        'Dynamic_Range_Preservation': (np.ptp(y_pred) / np.ptp(y_true)) if
np.ptp(y_true) != 0 else 1,
    }
    if include_plots:
        create_visual_diagnostics(y_true, y_pred, metrics)
    return metrics

def create_visual_diagnostics(y_true, y_pred, metrics):
    #    Створення візуальних діагностичних графіків
    fig, axes = plt.subplots(2, 3, figsize=(18, 12))
    fig.suptitle('Візуальна діагностика якості прогнозу', fontsize=16,
fontweight='bold')

    # 1. Фактичні vs Прогнозовані значення
    axes[0, 0].plot(y_true, label='Фактичні', alpha=0.8, linewidth=2)
    axes[0, 0].plot(y_pred, label='Прогноз', alpha=0.8, linewidth=2)
    axes[0, 0].set_title('Фактичні vs Прогнозовані значення')
    axes[0, 0].legend()
    axes[0, 0].grid(True, alpha=0.3)

    # 2. Розподіли
    axes[0, 1].hist(y_true, bins=20, alpha=0.7, label='Фактичні', density=True)
    axes[0, 1].hist(y_pred, bins=20, alpha=0.7, label='Прогноз', density=True)
    axes[0, 1].set_title('Розподіли значень')
    axes[0, 1].legend()

    # 3. Діаграма розкиду
    axes[0, 2].scatter(y_true, y_pred, alpha=0.6)
    min_val = min(np.min(y_true), np.min(y_pred))
    max_val = max(np.max(y_true), np.max(y_pred))
    axes[0, 2].plot([min_val, max_val], [min_val, max_val], 'r--', alpha=0.8)
    axes[0, 2].set_title('Діаграма розкиду')
    axes[0, 2].set_xlabel('Фактичні')
    axes[0, 2].set_ylabel('Прогноз')

    # 4. Помилки прогнозу
    errors = y_pred - y_true
    axes[1, 0].plot(errors, alpha=0.8, color='red')
    axes[1, 0].axhline(y=0, color='black', linestyle='-', alpha=0.5)
    axes[1, 0].set_title('Помилки прогнозу')
    axes[1, 0].set_ylabel('Похибка')
    axes[1, 0].grid(True, alpha=0.3)

    # 5. Автокореляція
    max_lag = min(50, len(y_true) // 2)
    lags = range(1, max_lag)
    acf_true = [np.corrcoef(y_true[:-lag], y_true[lag:])[0, 1] for lag in lags]
    acf_pred = [np.corrcoef(y_pred[:-lag], y_pred[lag:])[0, 1] for lag in lags]

    axes[1, 1].plot(lags, acf_true, label='Фактична', alpha=0.8)
    axes[1, 1].plot(lags, acf_pred, label='Прогноз', alpha=0.8)
    axes[1, 1].set_title('Автокореляція')
    axes[1, 1].legend()
    axes[1, 1].grid(True, alpha=0.3)

    # 6. Метрики (текст)
    axes[1, 2].axis('off')
```

```
    metrics_text = "\n".join([f"{k}: {v:.4f}" for k, v in metrics.items()])
    axes[1, 2].text(0.1, 0.9, "ВІЗУАЛЬНІ МЕТРИКИ:", transform=axes[1,
2].transAxes,
                    fontweight='bold', fontsize=12)
    axes[1, 2].text(0.1, 0.7, metrics_text, transform=axes[1, 2].transAxes,
                    fontfamily='monospace', fontsize=10, verticalalignment='top')

    plt.tight_layout()
    plt.show()

# Обчислюємо візуальні метрики
metrics = visual_metrics(y_true, y_pred, include_plots=True)

print("ВІЗУАЛЬНІ МЕТРИКИ ЯКОСТІ ПРОГНОЗУ:")
for key, value in metrics.items():
    print(f"{key:.<35} {value:>8.4f}")

def interpret_visual_metrics(metrics):
    print("\nІНТЕРПРЕТАЦІЯ ВІЗУАЛЬНИХ МЕТРИК:")

    # Схожість розподілів
    if metrics['Distribution_Similarity_Wasserstein'] < 0.5:
        print("Хороша схожість розподілів")
    else:
        print("Розподіли відрізняються")

    # Вирівнювання піків
    if metrics['Peak_Alignment'] > 0.8:
        print("Відмінне вирівнювання піків")
    elif metrics['Peak_Alignment'] > 0.6:
        print("Задовільне вирівнювання піків")
    else:
        print("Погане вирівнювання піків")

    # Фазовий зсув
    if abs(metrics['Phase_Shift']) <= 2:
        print("Мінімальний фазовий зсув")
    else:
        print(f"Помітний фазовий зсув: {metrics['Phase_Shift']:.1f}")

    # Збереження сезонності
    if metrics['Seasonality_Capture'] > 0.9:
        print("Відмінне збереження сезонності")
    elif metrics['Seasonality_Capture'] > 0.7:
        print("Задовільне збереження сезонності")
    else:
        print("Погане збереження сезонності")

# Використання
interpret_visual_metrics(metrics)

# ----------------------------
# Етап 1: WNN – попередня фільтрація сигналу
# ----------------------------
class WaveletNN:
    def   init  (self, wavelet='db4', level=3):
        self.wavelet = wavelet
        self.level = level
        self.coeffs = None
        self.original_mean = None
        self.original_std = None

    def fit_transform(self, X):
        # Зберігаємо параметри нормалізації
```

```
        self.original_mean = np.mean(X)
        self.original_std = np.std(X)
        # Застосовуємо Z-нормалізацію та медіанний фільтр
        #X_norm = zscore(X)
        X_norm = zscore(X, ddof=0)  # використовує такий самий ddof як
StandardScaler
        # Або         X_norm = (X - np.mean(X)) / np.std(X, ddof=0)
        X_filt = medfilt(X_norm, kernel_size=3)

        # Дискретне вейвлет-перетворення (à trous аналог через wavedec)
        self.coeffs = pywt.wavedec(X_filt, self.wavelet, level=self.level)
        # Розрахунок енергії кожної детальної компоненти
        energies = [np.sum(c**2) for c in self.coeffs[1:]]
        total_energy = sum(energies)
        weights = [e/total_energy for e in energies]  # α_j ваги компонент

        # Реконструкція сигналу з зваженими компонентами
        coeffs_weighted = [self.coeffs[0]] + [w*c for w,c in zip(weights,
self.coeffs[1:])]
        X_recon = pywt.waverec(coeffs_weighted, self.wavelet)
        X_recon = X_recon[:len(X)]  # обрізаємо до довжини сигналу
        # Конвертуємо назад у Series (якщо потрібно)
        X_recon = pd.Series(X_recon, index=X.index[:len(X_recon)])

        # Оцінка стабільності χ(w_m) як відношення SNR
        noise = np.random.normal(0, 0.01*np.std(X_recon), X_recon.shape)
        signal_power = np.mean(X_recon**2)
        noise_power = np.mean(noise**2)
        chi_wm = signal_power / (noise_power + 1e-10)

        return X_recon, chi_wm

    def inverse_transform(self, X_transformed):
        """Зворотне перетворення до оригінальної шкали"""
        if self.original_mean is None or self.original_std is None:
            raise ValueError("Спочатку викличте fit_transform()")

        return (X_transformed * self.original_std) + self.original_mean

# --- Етап 2: Формування ознак ---
def create_features(X, wavelet=w_star_final, level=opt_level, window_sma=5,
lags=3):
    df = pd.DataFrame({'X': X})
    df['SMA'] = df['X'].rolling(window=window_sma, min_periods=1).mean()
    df['Var'] = df['X'].rolling(window=window_sma, min_periods=1).var()

    for lag in range(1, lags+1):
        df[f'Lag_{lag}'] = df['X'].shift(lag).bfill()  #.fillna(method='bfill')

    # Вейвлет-декомпозиція
    coeffs = pywt.wavedec(df['X'].bfill(), wavelet, level=level)
    for i, c in enumerate(coeffs[1:], start=1):
        rec = pywt.waverec([np.zeros_like(coeffs[0])] + [c if j==i-1 else
np.zeros_like(coeffs[j+1]) for j in range(level)], wavelet)
        # обрізання до довжини df
        rec = rec[:len(df)]
        df[f'D_{i}'] = rec
        df[f'E_{i}'] = rec**2
    return df.bfill()  # .fillna(method='bfill')

def evaluate_model(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mse  = mean_squared_error(y_true, y_pred)
```

```
    r2 = r2_score(y_true, y_pred)
    mape = np.mean(np.abs((y_true - y_pred)/y_true))*100
    r = {'MAE': mae, 'RMSE': rmse,'MSE': mse, 'R2': r2, 'MAPE': mape}
    print(r)

def adfuller_test(errors, significance=0.05):
    """Тест Augmented Dickey-Fuller на стаціонарність помилок"""
    try:
        result = adfuller(errors)
        p_value = result[1]
        is_stationary = p_value < significance

        return {
            'p_value': p_value,
            'is_stationary': is_stationary,
            'test_statistic': result[0],
            'critical_values': result[4]
        }
    except Exception as e:
        return {
            'p_value': None,
            'is_stationary': False,
            'error': str(e)
        }
def autocorrelation(errors, lag=1):
    """
    Обчислення автокореляції помилок
    """
    try:
        if len(errors) <= lag:
            return 0
        # Використовуємо numpy для обчислення кореляції
        x = errors[:-lag]
        y = errors[lag:]
        return np.corrcoef(x, y)[0, 1] if len(x) > 1 else 0
    except:
        return 0

def trend_accuracy(y_true, y_pred):
    """
    Точність прогнозування напрямку тренду
    """
    try:
        if len(y_true) < 2:
            return 0

        # Перетворюємо в numpy arrays
        y_true = np.array(y_true)
        y_pred = np.array(y_pred)

        # Обчислюємо тренди
        true_trend = np.diff(y_true) > 0  # True якщо зростання, False якщо спад
        pred_trend = np.diff(y_pred) > 0

        # Обчислюємо точність
        accuracy = np.mean(true_trend == pred_trend) * 100
        return accuracy
    except:
        return 0

def peak_accuracy(y_true, y_pred, threshold=0.8):
    """
    Точність прогнозування піків
    """
```

```
    try:
        # Перетворюємо в numpy arrays
        y_true = np.array(y_true)
        y_pred = np.array(y_pred)

        # Визначаємо піки (значення вище 80-го перцентилю)
        true_peaks = y_true > np.percentile(y_true, threshold * 100)
        pred_peaks = y_pred > np.percentile(y_true, threshold * 100)

        # Обчислюємо точність
        accuracy = np.mean(true_peaks == pred_peaks) * 100
        return accuracy
    except:
        return 0

def coverage_probability(y_true, y_pred, errors, confidence=0.95):
    """
    Імовірність покриття довірчим інтервалом
    """
    try:
        std_error = np.std(errors)
        z_score = 1.96  # для 95% довірчого інтервалу

        lower_bound = y_pred - z_score * std_error
        upper_bound = y_pred + z_score * std_error

        coverage = np.mean((y_true >= lower_bound) & (y_true <= upper_bound)) *
100
        return coverage
    except:
        return 0

def rolling_mae(y_true, y_pred, window=5):
    """
    Ковзне MAE для оцінки стабільності прогнозу
    """
    try:
        errors = np.abs(np.array(y_true) - np.array(y_pred))
        if len(errors) < window:
            return np.mean(errors)

        rolling_errors = [np.mean(errors[i:i+window]) for i in range(len(errors)-
window+1)]
        return np.mean(rolling_errors)
    except:
        return 0

def calculate_all_metrics(y_true, y_pred):
    """Розрахунок всіх метрик якості прогнозу"""

    # Базові метрики
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)

    #print(len(y_true),len(y_pred),2,y_true[1:],3,y_true[:-
1],4,y_pred[1:],5,y_pred[:-1])
    #print(len(y_pred[1:]),len(y_pred[:-1]))
    # Додаткові метрики
    metrics = {
        # Базові
        'MAE': mae,
        'MSE': mse,
```

```
        'RMSE': rmse,
        'R2': r2,

        # Відносні похибки
        'MAPE': np.mean(np.abs((y_true - y_pred) / y_true)) * 100,
        'sMAPE': 100 * np.mean(2 * np.abs(y_pred - y_true) / (np.abs(y_true) +
np.abs(y_pred))),
        'MASE': calculate_mase(y_true, y_pred),

        # Кореляційні
        'Correlation': np.corrcoef(y_true, y_pred)[0, 1],
        'Direction_Accuracy': np.mean(np.sign(y_true[1:] - y_true[:-1]) ==
np.sign(y_pred[1:] - y_pred[:-1])) * 100,

        # Статистичні
        'Bias': np.mean(y_pred - y_true),
        'Std_Error': np.std(y_pred - y_true),

        # Інформаційні критерії (якщо є кількість параметрів)
        'AIC': calculate_aic(y_true, y_pred, n_params=100),
        'BIC': calculate_bic(y_true, y_pred, n_params=100),
    }

    return metrics

def calculate_mase(y_true, y_pred, seasonality=1):
    """Mean Absolute Scaled Error з перевіркою на нуль"""
    # Перевірка розмірностей
    if len(y_true) <= seasonality:
        return float('inf')

    naive_errors = np.abs(y_true[seasonality:] - y_true[:-seasonality])
    mean_naive_errors = np.mean(naive_errors)

    # Уникнення ділення на нуль
    if mean_naive_errors == 0:
        return float('inf')

    mase = np.mean(np.abs(y_true - y_pred)) / mean_naive_errors

    return mase
def calculate_aic(y_true, y_pred, n_params):
    """Akaike Information Criterion"""
    n = len(y_true)
    rss = np.sum((y_true - y_pred)**2)
    aic = n * np.log(rss/n) + 2 * n_params
    return aic

def calculate_bic(y_true, y_pred, n_params):
    """Bayesian Information Criterion"""
    n = len(y_true)
    rss = np.sum((y_true - y_pred)**2)
    bic = n * np.log(rss/n) + n_params * np.log(n)
    return bic

def time_series_specific_metrics(y_true, y_pred, dates=None):
    """Спеціалізовані метрики для часових рядів"""

    errors = y_pred - y_true

    ts_metrics = {
        # Стаціонарність помилок
        'Error_Mean': np.mean(errors),
        'Error_Std': np.std(errors),
```

```
        'Error_Stationary': adfuller_test(errors),

        # Автокореляція помилок
        'Error_Autocorr_1': autocorrelation(errors, lag=1),
        'Error_Autocorr_5': autocorrelation(errors, lag=5),

        # Точність напрямку
        'Trend_Accuracy': trend_accuracy(y_true, y_pred),
        'Peak_Accuracy': peak_accuracy(y_true, y_pred),

        # Довірчі інтервали
        'Coverage_95': coverage_probability(y_true, y_pred, errors),

        # Стабільність
        'Rolling_MAE_5': rolling_mae(y_true, y_pred, window=5),
        'Rolling_MAE_10': rolling_mae(y_true, y_pred, window=10),
    }

    return ts_metrics

# Розрахунок всіх метрик
def see_metrics(y_true, y_pred):

  # Перетворюємо в numpy arrays
    y_true_np = np.array(y_true)
    y_pred_np = np.array(y_pred)

    # Перевірка розмірностей
    if len(y_true_np) != len(y_pred_np):
        min_len = min(len(y_true_np), len(y_pred_np))
        y_true_np = y_true_np[:min_len]
        y_pred_np = y_pred_np[:min_len]

    all_metrics = calculate_all_metrics(y_true_np, y_pred_np)
    # print(len(y_val_real),len(y_pred_real))
    ts_metrics = time_series_specific_metrics(y_true_np, y_pred_np)

    print("ОСНОВНІ МЕТРИКИ:")
    for key, value in all_metrics.items():
        if isinstance(value, (int, float, np.number)):
            print(f"  {key}: {value:.4f}")
        else:
            print(f"  {key}: {value}")

    print("\nМЕТРИКИ ЧАСОВИХ РЯДІВ:")
    for key, value in ts_metrics.items():
        if isinstance(value, dict):
            # Обробка словників (наприклад, Error_Stationary)
            print(f"  {key}:")
            for sub_key, sub_value in value.items():
                if isinstance(sub_value, (int, float, np.number)):
                    print(f"    {sub_key}: {sub_value:.4f}")
                else:
                    print(f"    {sub_key}: {sub_value}")
        elif isinstance(value, (int, float, np.number)):
            # Числові значення
            print(f"  {key}: {value:.4f}")
        else:
            # Інші типи
            print(f"  {key}: {value}")

    # Візуалізація найважливіших метрик
    important_metrics = {
        'R2': all_metrics['R2'],
```

```
        'MAPE': all_metrics['MAPE'],
        'Direction_Accuracy': all_metrics['Direction_Accuracy'],
        'Correlation': all_metrics['Correlation'],
        'Trend_Accuracy': ts_metrics.get('Trend_Accuracy', 0),
        'Error_Std': ts_metrics.get('Error_Std', 0)
    }

    print("\nКЛЮЧОВІ ПОКАЗНИКИ:")
    for metric, value in important_metrics.items():
        if isinstance(value, (int, float, np.number)):
            status = "ДОБРЕ" if (
                (metric == 'R2' and value > 0.7) or
                (metric == 'MAPE' and value < 15) or
                (metric == 'Direction_Accuracy' and value > 60) or
                (metric == 'Correlation' and value > 0.7) or
                (metric == 'Trend_Accuracy' and value > 60) or
                (metric == 'Error_Std' and value < 0.5)
            ) else "ПОТРІБНО ПОКРАЩИТИ"
            print(f"  {metric}: {value:.4f} {status}")
        else:
            print(f"  {metric}: {value}")

# ----------------------------
# Етап 1: Оцінка стійкості до шуму +Попередня обробка сигналу
# ----------------------------
#Нормалізація (Min-Max): (x - min) / (max - min) → діапазон [0, 1]
#Стандартизація (Z-score): (x - μ) / σ → μ=0, σ=1

class DataProcessor:
    def __init__(self):
        self.original_mean = None
        self.original_std = None
        self.x_scaler = None
        #self.y_scaler = None

    def prepare_data(self, X,  window=3, snr_noise=0.01):
        #Перетворює дані в стандартний нормальний розподіл (μ=0, σ=1)
        # Перетворюємо в numpy array
        X_array = X.values if hasattr(X, 'values') else np.array(X)

        # Зберігаємо параметри для зворотного перетворення
        self.original_mean = np.mean(X_array)
        self.original_std = np.std(X_array)

        # Z-score нормалізація (працює з 1D масивами)
        X_norm = zscore(X_array, ddof=0)

        # Медіанний фільтр для згладжування викидів
        X_filt = medfilt(X_norm, kernel_size=window)

        # Конвертуємо назад у Series (якщо потрібно)
        X_filt = pd.Series(X_filt, index=X.index[:len(X_filt)])

        # Додаємо гаусовий шум для перевірки стабільності
        noise = np.random.normal(0, snr_noise*np.std(X_filt), X_filt.shape)
        X noisy = X filt + noise

        # Розрахунок SNR для оцінки чутливості
        signal_power = np.mean(X_filt**2)
        noise_power = np.mean((X_noisy - X_filt)**2)
        chi_wm = signal_power / (noise_power + 1e-10)  # показник стійкості

        return X_filt, chi_wm
        # # Масштабування y
```

```
        # self.y_scaler = StandardScaler()
        # y_scaled = self.y_scaler.fit_transform(y.reshape(-1, 1)).flatten()

        # # Створення послідовностей
        # X_seq, y_seq = create_sequences(X_scaled, y_scaled, timesteps)

        # return X_seq, y_seq

    def inverse_transform(self, x_scaled):
        """Зворотне перетворення для X"""
        if self.original_mean is None:
            raise ValueError("Параметри не ініціалізовані. Спочатку викличте
prepare_data")
        return (x_scaled * self.original_std) + self.original_mean


# Використання
# processor = DataProcessor()
# X_train_lstm, y_train_lstm = processor.prepare_data(X_train, y_train,
timesteps=24)

# # Після прогнозування
# y_val_real = processor.inverse_transform_y(y_val_seq[:row])
# y_pred_real = processor.inverse_transform_y(y_pred_scaled)

# ----------------------------
# Функція для оптимізації гіперпараметрів через Optuna
# ----------------------------
def optimize_hyperparameters(X_train, y_train, X_val, y_val,
n_trials=20,batch_size=16):
    def objective(trial):
        lstm_units = trial.suggest_int('lstm_units', 16, 128)
        dropout = trial.suggest_float('dropout', 0.1, 0.5)
        lr = trial.suggest_loguniform('lr', 1e-4, 1e-2)

        model = build_lstm_model(
            input_shape=(X_train.shape[1], X_train.shape[2]),
            lstm_units=lstm_units,
            dropout=dropout,
            lr=lr
        )
        model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=20,
            batch_size=batch_size,
            verbose=0
        )
        y_pred = model.predict(X_val, verbose=0)
        rmse = np.sqrt(mean_squared_error(y_val, y_pred))
        return rmse

    study = optuna.create_study(direction='minimize')
    study.optimize(objective, n_trials=n_trials)

    print("Best trial:")
    print("  Value (RMSE):", study.best_trial.value)
    print("  Params:", study.best_trial.params)

    return study.best_trial.params

def plot_results_hist(y_true, y_pred,x_true,x_pred, dates,
forecast_start_idx=None):
    plt.figure(figsize=(12,6))
```

```
    if forecast_start_idx is None:
        forecast_start_idx = len(y_true)

    # Створюємо повний datetime індекс
    last_date = dates.iloc[-1]

    # Генеруємо дати для прогнозу (припускаємо однаковий інтервал)
    if len(dates) > 1:
        time_delta = dates.iloc[1] - dates.iloc[0]
        forecast_dates = [last_date + (i + 1) * time_delta for i in
range(len(y_pred))]
    else:
        forecast_dates = pd.date_range(start=last_date, periods=len(y_pred)+1,
freq='D')[1:]

    # Створюємо індекси для всього періоду
    all_dates = list(dates[-len(y_true):]) + forecast_dates
    # Об'єднуємо дані
    all_data = np.concatenate([y_true, y_pred])

    forecast_dates_formatted = [date.strftime('%m-%d') for date in forecast_dates]
    dates_formatted = [date.strftime('%m-%d') for date in dates[-len(y_true):]]
    all_dates_formatted = [date.strftime('%m-%d') for date in all_dates]


    # Побудова єдиної лінії
    plt.plot(all_dates, all_data, color='red', linestyle='--', linewidth=1,
alpha=0.7)
    plt.plot(all_dates, np.concatenate([x_true, x_pred]),
color='green',linestyle='--', linewidth=1, alpha=0.7, label='Historical')

    # plt.plot(dates[-len(x_true):], x_true, label='Historical Original',
color='green',linestyle='--', linewidth=1, alpha=0.7)
    # plt.plot(forecast_dates, x_pred, label='Historical Forecast',
color='green',linestyle='--', linewidth=1, alpha=0.7)

    #plt.axvline(x=len(y_true)-1, color='gray', linestyle=':', alpha=0.7,
label='Forecast Start')
    plt.axvline(x=last_date, color='gray', linestyle=':', alpha=0.7,
label='Forecast Start')

    # Побудова графіка
    plt.plot(dates[-len(y_true):], y_true, label='Historical Actual',
color='blue', linewidth=2)
    plt.plot(forecast_dates, y_pred, label='Forecast', color='red', linestyle='--
', linewidth=2)


    '''# Історичні дані + прогноз
    historical_indices = range(len(y_true))
    future_indices = range(len(y_true), len(y_true) + len(y_pred))

    plt.plot(historical_indices, y_true, label='Historical Actual', color='blue')
    plt.plot(future_indices, y_pred, label='Forecast', color='red', linestyle='--
', marker='s', markersize=2, linewidth=1)'''

    # Форматуємо ось X з мітками "місяць-день"
    plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%m-
%d'))
    plt.gcf().autofmt_xdate()  # автоматичне форматування дат


    plt.legend()
    plt.xlabel('Time Period')
```

```
    plt.ylabel('Value (MB)')
    plt.title('Historical Data and Forecast')
    plt.grid(True, alpha=0.3)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

def plot_results_history(y_true, y_pred,x_true,x_pred, dates,form = 'd',
forecast_start_idx=None):
    plt.figure(figsize=(12,6))
    if forecast_start_idx is None:
        forecast_start_idx = len(y_true)

    # Створюємо повний datetime індекс
    last_date = dates.iloc[-1]

    # Генеруємо дати для прогнозу (припускаємо однаковий інтервал)
    if len(dates) > 1:
        time_delta = dates.iloc[1] - dates.iloc[0]
        forecast_dates = [last_date + (i + 1) * time_delta for i in
range(len(y_pred))]
    else:
        forecast_dates = pd.date_range(start=last_date, periods=len(y_pred)+1,
freq='D')[1:]

    # Форматуємо для графіка
    forecast_dates_formatted = [date.strftime('%m-%d') for date in forecast_dates]

    # Побудова графіка
    plt.plot(forecast_dates, x_pred, label='Historical Data', color='blue',
linewidth=1)
    plt.plot(forecast_dates, y_pred, label='Forecast', color='red', linestyle='--
', linewidth=2)

    # Форматуємо ось X з мітками "місяць-день"
    form = '%H:%M' if form == 'd' else '%m-%d'
    #plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%m-
%d'))
    plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter(form))
    plt.gcf().autofmt_xdate()  # автоматичне форматування дат


    plt.legend()
    plt.xlabel('Time Period')
    plt.ylabel('Value (MB)')
    plt.title('Historical Data and Forecast')
    plt.grid(True, alpha=0.3)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    metrics1 = evaluate_model(x_pred, y_pred)
    print(metrics1)
    #print(len(x_pred),len(y_pred))
    #Для нормалізованих даних часових рядів:MAE: < 0.2 RMSE: < 0.3 R²: > 0.7 MAPE:
< 15%
    see_metrics(x_pred, y_pred)


#Етап 1
if(flag==1):
    # Етап 1: WNN
    wnn = WaveletNN(wavelet=w_star_final, level=opt_level)
    X_filt, chi_wm = wnn.fit_transform(X_train)
```

```
    print("Stability χ(w_m):", chi_wm) #> 1000 Відмінно Сигнал практично без шуму.
χ(w_m): 10412.148462545016
else:
    processor = DataProcessor()
    X_filt, chi_wm = processor.prepare_data(X_train)
    print("Stability χ(w_m):", chi_wm)


# Пошук оптимальних гіперпараметрів
'''best_params = optimize_hyperparameters(
    X_train_lstm, y_train_lstm,
    X_val_lstm, y_val_lstm,
    n_trials=10, batch_size=16
)
print(best_params)'''
'''Best trial:
  Value (RMSE): 0.3582932578221502
  Params: {'lstm_units': 47, 'dropout': 0.36028369738466826, 'lr':
0.00021720539299376453}
{'lstm_units': 47, 'dropout': 0.36028369738466826, 'lr':
0.00021720539299376453}'''
best_params = {'lstm_units':47, 'dropout': 0.36028369738466826, 'lr':
0.00021720539299376453}

#777
df = pd.read_csv(url, sep=';', engine='python', on_bad_lines='skip')
#date_time = pd.to_datetime(df.iloc[:, 0])
date_time = pd.to_datetime(df['Timestamp'], unit='s')

#WebTraffic = df['WebTraffic(MB)']
WebTraffic = df.iloc[:, 2]
WebTraffic = pd.to_numeric(
    WebTraffic
    .astype(str)
    .str.replace(',', '.', regex=False)
    .str.extract(r'([\d\.]+)')[0],
    errors='coerce'   # нечислові значення стануть NaN
)

w_star_final = 'rbio3.1'
opt_level= 2
n = 2016 #week
X_train = WebTraffic.iloc[:n]
X2_train = WebTraffic.iloc[1*n:2*n]
rng = date_time.iloc[:n]

web = WebTraffic[:n*12]
SEQ_LEN = n


#ФОРМУВАННЯ LSTM-ПОСЛІДОВНОСТЕЙ
def create_sequences(data, seq_len=2016):
    X, y = [], []
    for i in range(seq_len, len(data)):
        X.append(data[i-seq_len:i])
        y.append(data[i])
    return np.array(X), np.array(y)

def prepare_multi_target_data(X, y,
                              h1=12, h6=72, h24=288, d7 = 2016):
    y1, y6, y24, yd7 = [], [], [], []

    for i in range(len(X)):
        if i + d7 < len(y):
```

```
            y1.append(y[i:i+h1])
            y6.append(y[i:i+h6])
            y24.append(y[i:i+h24])
            yd7.append(y[i:i+d7])

    min_len = min(len(y1), len(y6), len(y24), len(yd7))

    return (
        X[:min_len],
        {
            "1_hour":  np.array(y1[:min_len]),
            "6_hours": np.array(y6[:min_len]),
            "24_hours":np.array(y24[:min_len]),
            "7_days":np.array(yd7[:min_len])
        }
    )

#ПОБУДОВА МОДЕЛІ ( MULTI-OUTPUT)
def build_multi_horizon_lstm(input_shape):
    inp = Input(shape=input_shape)

    x = LSTM(256, return_sequences=True)(inp)
    x = Dropout(0.15)(x)

    x = LSTM(128, return_sequences=False)(x)
    x = Dropout(0.15)(x)

    x = Dense(256, activation="relu")(x)
    x = Dense(128, activation="relu")(x)
    x = Dense(64,  activation="relu")(x)

    out_1h = Dense(12, name="1_hour")(x)
    out_6h = Dense(72, name="6_hours")(x)
    out_24h = Dense(288, name="24_hours")(x)
    out_7d   = Dense(2016,name="7_days")(x)

    model = Model(
        inputs=inp,
        outputs=[out_1h, out_6h, out_24h, out_7d]
    )

    model.compile(
    optimizer=Adam(learning_rate=0.0003,   clipnorm=1.0) #(learning_rate=0.001),
    loss={
        "1_hour": "mse",
        "6_hours": "mse",
        "24_hours": "mse",
        "7_days": "mse",
    },
    loss_weights={
        "1_hour": 0.3,
        "6_hours": 0.20,
        "24_hours": 0.25,
        "7_days": 0.25
    },
    metrics={
        "1_hour": ["mae"],
        "6_hours": ["mae"],
        "24_hours": ["mae"],
        "7_days": ["mae"]
    }
    )

    return model
```

```
#PROPHET — БАЗОВИЙ ПРОГНОЗ
df_prophet = pd.DataFrame({
    "ds": pd.date_range(
        start="2021-06-14",
        periods=len(web),
        freq="5min"
    ),
    "y": web.values
})

prophet = Prophet(
    daily_seasonality=True,
    weekly_seasonality=True,
    yearly_seasonality=False
)

prophet.fit(df_prophet)

future = prophet.make_future_dataframe(
    periods=n, freq="5min"
)

forecast = prophet.predict(future)

prophet_in_sample = forecast["yhat"].values[:len(web)]
prophet_future = forecast["yhat"].values[-n:]

#ДАНІ ДЛЯ LSTM
residuals = web.values - prophet_in_sample
#residuals = web.values

#МАСШТАБУВАННЯ
scaler = StandardScaler()
traffic_scaled = scaler.fit_transform(
    residuals.reshape(-1, 1)
).flatten() #2016*81=163296
print(traffic_scaled,len(traffic_scaled))

X_all, y_all = create_sequences(traffic_scaled, SEQ_LEN)
X_all = X_all[..., np.newaxis]# (samples, timesteps, 1)
#MULTI-HORIZON TARGETS
X_all, y_all = prepare_multi_target_data(X_all, y_all)
#TRAIN / VALID SPLIT (TIME-BASED)
n_samples = X_all.shape[0]
#Розбити ІНДЕКСИ, а не дані
split = int(n_samples * 0.8)

X_train = X_all[:split]
X_val   = X_all[split:]

y_train = {
    k: v[:split] for k, v in y_all.items()
}
y_val = {
    k: v[split:] for k, v in y all.items()
}
#ПЕРЕВІРКА
print(X_train.shape, X_val.shape)
for k in y_train:
    print(k, y_train[k].shape, y_val[k].shape)

model_plus = build_multi_horizon_lstm(
    input_shape=(X_train.shape[1], X_train.shape[2])
```

```
)

model_plus.summary()

#НАВЧАННЯ

history = model_plus.fit(
    X_train,
    y_train,
    validation_data=(X_val, y_val),
    epochs=15,
    batch_size=288,
    verbose=1
)
#ГРАФІК НАВЧАННЯ
plt.plot(history.history["loss"], label="Train")
plt.plot(history.history["val_loss"], label="Validation")
plt.legend()
plt.title("Training Loss")
plt.show()
#ПРОГНОЗ
X_last = X_val[-1:] # одне вікно

pred_1h, pred_6h, pred_24h, pred_7d = model_plus.predict(X_last)
#ПОВЕРНЕННЯ В РЕАЛЬНИЙ МАСШТАБ
pred_7d_real = scaler.inverse_transform(
    pred_7d.reshape(-1, 1)
).flatten()
#ВІЗУАЛІЗАЦІЯ ПРОГНОЗУ
real_7d_ = scaler.inverse_transform(
    y_val["7_days"][-1].reshape(-1, 1)
).flatten()
print(real_7d_)
plt.figure(figsize=(12,4))
plt.plot(real_7d_, label="Real")
plt.plot(pred_7d_real, label="Predicted")
plt.legend()
plt.title("Forecast")
plt.show()

#ГОТОВЕ ДО ВИКОРИСТАННЯ В СИСТЕМІ
if np.max(pred_1h_real := scaler.inverse_transform(pred_1h.reshape(-1,1))) > 2000:
    print("Expected overload — scale infrastructure")
#ФІНАЛЬНИЙ HYBRID ПРОГНОЗ
final_7d = prophet_future + pred_7d_real
#ВІЗУАЛІЗАЦІЯ
real_7d = web.values[-n:]
print(real_7d)
plt.figure(figsize=(13,4))
plt.plot(real_7d, label="Real")
plt.plot(prophet_future, label="Prophet")
plt.plot(final_7d, label="Hybrid Prophet + LSTM")
plt.legend()
plt.title("Hybrid Forecast")
plt.show()

#з мульти-ЛСТМ прогнозу на тиждень зробити ітеративний прогноз на місяць.
forecasts = 4 #4 week
steps = n  # 5-хв інтервал → 2016 точок на тиждень
SEQ_LEN = X_last.shape[1]  # довжина вікна LSTM

#Ітеративний прогноз
# Копіюємо останнє вікно
history_scaled = X_last.copy().reshape(-1)  # 1D
```

```
pred_residuals = []

for d in range(forecasts):
    # Формуємо вхід для LSTM
    X_input = history_scaled[-SEQ_LEN:].reshape(1, SEQ_LEN, 1)

    # Прогноз мульти-ЛСТМ
    pred_1h, pred_6h, pred_24h, pred_7d = model_plus.predict(X_input)

    # Беремо прогноз на 7days
    pred_7d = pred_7d.flatten()

    # Повертаємо в реальний масштаб
    pred_7d_real = scaler.inverse_transform(pred_7d.reshape(-1,1)).flatten()

    # Додаємо до загального прогнозу
    pred_residuals.extend(pred_7d_real)

    # Оновлюємо історію для наступного дня
    history_scaled = np.concatenate([history_scaled, pred_7d])

#Додавання Prophet прогнозу (опціонально)
prophet_future_ = np.tile(prophet_future, forecasts)
final_forecast = prophet_future_ + np.array(pred_residuals)

#Візуалізація
plt.figure(figsize=(15,5))
plt.plot(final_forecast, label="Hybrid forecast (28 days)")
plt.title("28-Day Hybrid Forecast (Prophet + Multi-LSTM)")
plt.xlabel("5-min intervals")
plt.ylabel("Traffic")
plt.legend()
plt.show()

id_keras = '1ZQoo0CBIZq7oE3KnucQafIpUfOJ_2vJI' #"lstm_model.keras"
drive_link = f"https://drive.google.com/uc?export=download&id={id_keras}"

def update_existing_drive_file( drive_link, filename="model.keras"):
    """Оновлення існуючого файлу в Google Drive"""
    # Аутентифікація
    auth.authenticate_user()
    drive_service = build('drive', 'v3')
    try:
        # Отримати file_id з посилання
        if 'id=' in drive_link:
            file_id = drive_link.split('id=')[1].split('&')[0]
        elif '/file/d/' in drive_link:
            file_id = drive_link.split('/file/d/')[1].split('/')[0]
        else:
            print("Неправильний формат посилання")
            return None
        print(f"Оновлення файлу з ID: {file_id}")
        # Оновити файл
        media = MediaFileUpload(filename, resumable=True)

        updated file = drive service.files().update(
            fileId=file_id,
            media_body=media
        ).execute()

        print(f"Модель успішно оновлена в Google Drive!")
        print(f"Посилання: {drive_link}")
        return drive_link
```

```
    except Exception as e:
        print(f"Помилка оновлення: {e}")
        return None

# 1. Зберегти модель локально
filename = "lstm_model.keras"
save_model(model_plus, filename)
file_size = os.path.getsize(filename) / (1024 * 1024)
print(f"Модель збережена локально: {filename} ({file_size:.2f} MB)")

# 2. Оновлення моделі в існуючому файлі
updated_link = update_existing_drive_file(
    drive_link,
    filename
)

def download_from_drive(drive_link, filename):
    """Правильне завантаження файлу з Google Drive"""
    try:
        print(f"Завантажуємо з: {drive_link}")
        # Отримуємо ID файлу з посилання
        if 'id=' in drive_link:
            file_id = drive_link.split('id=')[1].split('&')[0]
        elif '/file/d/' in drive_link:
            file_id = drive_link.split('/file/d/')[1].split('/')[0]
        else:
            file_id = drive_link

        # Формуємо пряме посилання для завантаження
        download_url = f"https://drive.google.com/uc?export=download&id={file_id}"

        # Робимо запит з сесією для обробки cookies
        session = requests.Session()
        response = session.get(download_url, stream=True)

        # Обробка великих файлів (підтвердження)
        if "confirm=" in response.url:
            # Отримуємо токен підтвердження
            confirm_token = response.url.split('confirm=')[1].split('&')[0]
            download_url =
f"https://drive.google.com/uc?export=download&id={file_id}&confirm={confirm_token}
"
            response = session.get(download_url, stream=True)

        # Перевіряємо успішність запиту
        response.raise_for_status()

        # Завантажуємо файл
        with open(filename, 'wb') as f:
            for chunk in response.iter_content(chunk_size=8192):
                if chunk:
                    f.write(chunk)

        # Перевіряємо, що файл створений і не порожній
        if os.path.exists(filename) and os.path.getsize(filename) > 0:
            file size = os.path.getsize(filename) / (1024 * 1024)
            print(f"Файл завантажено: {filename} ({file_size:.2f} MB)")
            return filename
        else:
            print("Файл порожній або не створений")
            return None

    except Exception as e:
        print(f"Помилка завантаження: {e}")
```

```
        return None

model_path = download_from_drive(drive_link, "lstm_model.keras")

if model_path and os.path.exists(model_path):
    model9 = load_model(model_path)
    print("Модель успішно завантажена!")
else:
    print("Не вдалося завантажити модель")

pred_1h, pred_6h, pred_24h, pred_7d = model9.predict(X_last)
#ПОВЕРНЕННЯ В РЕАЛЬНИЙ МАСШТАБ
pred_7d_real = scaler.inverse_transform(
    pred_7d.reshape(-1, 1)
).flatten()
pred_24h_real = scaler.inverse_transform(
    pred_24h.reshape(-1, 1)
).flatten()
pred_1h_real = scaler.inverse_transform(
    pred_1h.reshape(-1, 1)
).flatten()
pred_6h_real = scaler.inverse_transform(
    pred_6h.reshape(-1, 1)
).flatten()

#ФІНАЛЬНИЙ HYBRID ПРОГНОЗ
final_7d = prophet_future + pred_7d_real
final_24h = prophet_future[:288] + pred_24h_real
final_1h = prophet_future[:12] + pred_1h_real
final_6h = prophet_future[:72] + pred_6h_real
#ВІЗУАЛІЗАЦІЯ
real_7d = web.values[-n:]
real_24h = web.values[-288:]
real_6h = web.values[-72:]
real_1h = web.values[-12:]

plt.figure(figsize=(13,4))
plt.plot(real_7d, label="Real")
plt.plot(final_7d, label="Forecast")
plt.legend()
plt.show()
evaluate_model(real_7d,final_7d)

plt.figure(figsize=(13,4))
plt.plot(real_24h, label="Real")
plt.plot(final_24h, label="Forecast")
plt.legend()
plt.show()
evaluate_model(real_24h,final_24h)

plt.figure(figsize=(13,4))
plt.plot(real_1h, label="Real")
plt.plot(final_1h, label="Forecast")
plt.legend()
plt.show()
evaluate_model(real_1h,final_1h)

plt.figure(figsize=(13,4))
plt.plot(real_6h, label="Real")
plt.plot(final_6h, label="Forecast")
plt.legend()
plt.show()
evaluate_model(real_6h,final_6h)
```

```
from tensorflow.keras.optimizers import RMSprop
# Завантажуємо модель для отримання архітектури
original_model = load_model("lstm_model.keras")
# Створюємо нову модель з тією ж архітектурою
lstm_model_loaded = clone_model(original_model)
# Компілюємо з бажаним оптимізатором
lstm_model_loaded.compile(
    optimizer=RMSprop(learning_rate=0.001),
    loss='mse',
    metrics=['mae']
)

# Відновлення оригінальних значень
y_pred_real = wnn.inverse_transform(y_pred_scaled)
y_val_real = wnn.inverse_transform(y_val_seq[-row:])

plot_results_history(y_val_real, y_pred_real,X_train[-
row:],WebTraffic.iloc[n:n+row], rng,'m')


#80
!pip install dash
import seaborn as sns
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
# ------------------------------
# Візуалізація результатів прогнозу
# ------------------------------
def generate_report(y_true, y_pred, model_name='LSTM/WNN'):
    # Розрахунок похибок
    errors = y_true - y_pred
    mae = np.mean(np.abs(errors))
    rmse = np.sqrt(np.mean(errors**2))
    mape = np.mean(np.abs(errors / (y_true + 1e-10))) * 100
    r2 = 1 - np.sum(errors**2)/np.sum((y_true - np.mean(y_true))**2)

    print(f"Model: {model_name}")
    print(f"MAE: {mae:.4f}, RMSE: {rmse:.4f}, MAPE: {mape:.2f}%, R2: {r2:.4f}")

    # Лінійний графік фактичних та прогнозованих значень
    plt.figure(figsize=(12,5))
    plt.plot(y_true, label='Real')
    plt.plot(y_pred, label='Forecast')
    plt.title('Прогноз vs Реальні значення')
    plt.legend()
    plt.show()

    # Boxplot помилок
    plt.figure(figsize=(6,4))
    sns.boxplot(errors)
    plt.title('Розподіл похибок прогнозу')
    plt.show()

    # Heatmap кореляцій
    df = pd.DataFrame({'Real': y_true, 'Forecast': y_pred})
    corr = df.corr()
    plt.figure(figsize=(5,4))
    sns.heatmap(corr, annot=True, cmap='coolwarm')
    plt.title('Кореляція фактичних та прогнозованих значень')
    plt.show()
    return {'MAE': mae, 'RMSE': rmse, 'MAPE': mape, 'R2': r2}
```

```
# ----------------------------
# Dash дашборд для інтерактивного аналізу
# ----------------------------
def run_dashboard(y_true, y_pred, dates=None):
    app = Dash(__name__)
    if dates is None:
        dates = np.arange(len(y_true))

    df = pd.DataFrame({'Date': dates, 'Real': y_true, 'Forecast': y_pred, 'Error':
y_true - y_pred})

    app.layout = html.Div([
        html.H1("Прогноз навантаження на вебсервер"),
        dcc.Graph(id='line-plot', figure={
            'data': [
                {'x': df['Date'], 'y': df['Real'], 'type': 'line', 'name':
'Real'},
                {'x': df['Date'], 'y': df['Forecast'], 'type': 'line', 'name':
'Forecast'}
            ],
            'layout': {'title': 'Фактичні та прогнозовані значення'}
        }),
        dcc.Graph(id='error-box', figure={
            'data': [
                {'y': df['Error'], 'type': 'box', 'name': 'Error'}
            ],
            'layout': {'title': 'Boxplot похибок'}
        }),
        dcc.Graph(id='heatmap-corr', figure={
            'data': [{
                'z': df[['Real','Forecast']].corr().values,
                'x': ['Real','Forecast'],
                'y': ['Real','Forecast'],
                'type': 'heatmap',
                'colorscale': 'Viridis',
                'zmin': -1,
                'zmax': 1
            }],
            'layout': {'title': 'Heatmap кореляцій'}
        })
    ])

    fig = go.Figure()
    fig.add_trace(go.Scatter(y=y_true, mode='lines', name='Actual'))
    fig.add_trace(go.Scatter(y=y_pred, mode='lines', name='Predicted'))

    app.layout = html.Div([
        html.H1("Прогноз навантаження вебсервера"),
        dcc.Graph(figure=fig)
    ])
    app.run(debug=True, port=8050)

# ----------------------------
# Демонстрація генерації звіту
# ----------------------------
if   name   == "  main  ":
    # Використаємо прикладні дані з попередньої частини (WNN+LSTM)
    metrics = generate_report(y_true, y_pred)
    print(metrics)
    # Запуск інтерактивного дашборду
    run_dashboard(y_true, y_pred)
```

ДОДАТОК Е.

Статистика роботи досліджуваного сервера

Пропускна здатність протягом місяця за типами протоколів

День	All Traffic	HTTP Traffic	FTP Traffic	IMAP Traffic	POP3 Traffic	SMTP Traffic
1	10,97 GB	10,84 GB	18,36 MB	2,94 MB	0	107,85 MB
2	10,69 GB	10,64 GB	4,07 MB	7,17 MB	0	40,04 MB
3	8,76 GB	8,69 GB	33,25 MB	38,14 MB	0	4,32 MB
4	5,47 GB	5,47 GB	5,01 MB	588,01 KB	0	40,62 KB
5	6,46 GB	6,46 GB	0	354,11 KB	0	46,36 KB
6	10,82 GB	10,79 GB	13,65 MB	21,62 MB	0	2,22 MB
7	10,02 GB	9,89 GB	31,85 MB	32,65 MB	0	66,28 MB
8	11,26 GB	10,97 GB	27,77 MB	45,6 MB	0	226,15 MB
9	12,25 GB	12,18 GB	24,63 MB	47,06 MB	0	985,92 KB
10	8,41 GB	8,18 GB	8,25 MB	31,74 MB	0	201,07 MB
11	5,34 GB	5,34 GB	0	191,73 KB	0	23,33 KB
12	5,72 GB	5,71 GB	8,39 MB	634,93 KB	0	64,08 KB
13	9,44 GB	9,33 GB	72,23 MB	45,8 MB	0	1,64 MB
14	9,67 GB	9,62 GB	38,26 MB	11,75 MB	0	5,03 MB
15	12,77 GB	12,38 GB	66,92 MB	17,94 MB	0	310,85 MB

16	9,87 GB	9,76 GB	92,04 MB	8,71 MB	0	8,14 MB
17	9,34 GB	9,24 GB	47,89 MB	49,71 MB	0	1,55 MB
18	4,75 GB	4,75 GB	0	175,59 KB	0	30,1 KB
19	5,71 GB	5,54 GB	173,13 MB	341,81 KB	0	80,5 KB
20	9,3 GB	8,96 GB	307,92 MB	34,4 MB	0	4,31 MB
21	9,83 GB	9,63 GB	23,33 MB	179,2 MB	0	9,14 MB
22	9,74 GB	9,57 GB	36,59 MB	3,57 MB	0	135,56 MB
23	10,71 GB	10,41 GB	295,58 MB	2,13 MB	0	8,99 MB
24	12,42 GB	12,09 GB	302,36 MB	37,68 MB	0	5,8 MB
25	6,08 GB	6,08 GB	0	477,78 KB	0	47,48 KB
26	7,1 GB	7,09 GB	6,3 MB	4,53 KB	0	48,87 KB
27	9,16 GB	8,97 GB	181,74 MB	9,8 MB	0	1,22 MB
28	8,49 GB	8,44 GB	10,9 MB	18,25 MB	0	20,32 MB
29	10,29 GB	10,24 GB	24,89 MB	26,46 MB	0	5,45 MB
30	9,15 GB	9,12 GB	8,81 MB	22,06 MB	0	807,51 KB
31	10,68 GB	10,51 GB	165,8 MB	5,14 MB	0	166,21 KB
Усього	280,69 GB	276,88 GB	1,98 GB	702,24 MB	0	1,14 GB

ДОДАТОК Ж. Публікації за темою дисертації

1. Radchenko K. Applying Wavelet Transforms for Web Server Load Forecasting / Z. Hu, I. Tereykovskiy, L. Tereykovska, M. Tsiutsiura, K. Radchenko // *In: Hu, Z., Petoukhov, S., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer, Cham*. https://doi.org/10.1007/978-3-030-16621-2_2 (Закордонне видання, реферується наукометричною базою «Scopus»). *Здобувач обґрунтував переваги та перспективи використання вейвлет-перетворень при прогнозуванні навантаження на вебсервер, виконав аналітичний огляд методів і підготував експериментальні приклади застосування; Z. Hu долучився до формування загальної структури статті; I. Tereykovskiy брав участь у перевірці коректності обраних базисних вейвлетів; L. Tereykovska проаналізувала властивості вебтрафіку та їх відповідність модельним припущенням; M. Tsiutsiura допоміг у формуванні архітектури програмного експерименту та узагальненні результатів.*

2. Радченко К. О. Концептуальна модель забезпечення ефективності прогнозування навантаження на вебсервер / К. О. Радченко // *Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія : Технічні науки*. – 2020. – Т. 31 (70), № 6. – С. 135–141. https://doi.org/10.32838/TNU-2663-5941/2020.6-1/23 (Фахове наукове видання України категорії Б). *Здобувач представив концептуальну модель ефективного прогнозування навантаження на вебсервер, визначив модель параметрів вебтрафіку.*

3. Радченко К.О. Застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер комп’ютерних мереж загального призначення / К.О. Радченко // *Комп’ютерно-інтегровані технології: освіта, наука, виробництво*. – Луцьк. – № 45. – 2021. – С. 90 – 96. https://doi.org/10.36910/6775-2524-0560-2021-45-13 (Фахове наукове видання України категорії Б). *Здобувач дослідив можливості застосування дискретних вейвлет-перетворень для прогнозування рівня навантаження на вебсервер*.

4. Радченко К.О. Особливості прогнозування рівня вебтрафіку у

комп'ютерних мережах загального призначення / К.О. Радченко // *Проблеми інформатизації та управління*. – Національний авіаційний університет – № 3(71). – 2022. – С. 41 – 50. https://doi.org/10.18372/2073-4751.71.17002 (Фахове наукове видання України категорії Б). *Здобувач проаналізував особливості прогнозування рівня вебтрафіку*.

5. Дичка, І., Радченко, К., Терейковський, І., & Терейковська, Л. (2024). Концептуальна модель процесу прогнозування навантаження на вебсервер. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (54), 74-83. https://doi.org/10.36910/6775-2524-0560-2024-54-09 (Фахове наукове видання України категорії Б). *Здобувач сформулював основи концептуальної моделі прогнозування навантаження на вебсервер, визначив функціональну структуру системи, узагальнив методи обробки часових рядів; І. Дичка забезпечив загальне методологічне керівництво дослідженням; І. Терейковський перевірив логічну узгодженість моделі; Л. Терейковська оцінила відповідність моделі реальним умовам функціонування вебсервера*.

6. Радченко, К., & Романенко, М. (2024). Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (56), 261-268. https://doi.org/10.36910/6775-2524-0560-2024-56-33 (Фахове наукове видання України категорії Б). *Здобувач дослідив особливості попередньої обробки часових рядів за допомогою дискретного вейвлет-перетворення, розробив структуру гібридної моделі; М. Романенко виконала аналіз отриманих результатів*.

7. K. Radchenko and I. Tereykovskyi, "Integrated Neural Network and Wavelet-Based Model for Web Server Load Forecasting", *SISIOT (Security of Infocommunication Systems and Internet of Things)*, vol. 2, no. 2, p. 02006, Dec. 2024, https://doi.org/10.31861/sisiot2024.2.02006 (Фахове наукове видання України категорії Б). *Здобувач розробив інтегральну модель прогнозування навантаження на вебсервер, провів експериментальне моделювання; І. Терейковський допоміг у валідації результатів та їх формальному описі*.

8. Радченко, К. О., & Терейковський, І. А. (2025). Метод прогнозування

навантаження на вебсервер з урахуванням вейвлет-аналізу веблогів та вебтрафіку. *КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ: ОСВІТА, НАУКА, ВИРОБНИЦТВО*, (60), 247-259. https://doi.org/10.36910/6775-2524-0560-2025-60-27 (Фахове наукове видання України категорії Б). *Здобувач презентував удосконалений метод прогнозування навантаження на вебсервер, виконав експериментальну оцінку точності; І. Терейковський долучився до аналізу результатів.*

9. Radchenko K. O. The application of discrete wavelet transform for forecasting the operational load of the Web server of the distance education system / L. A. Tereikovska, K. O. Radchenko, K. O. Kharlamov. // *The Eighth World Congress "Aviation in the XXI-st Century" - «Safety in Aviation And Space Technologies»*. – Kyiv Aviation University 10.10 – 12.10.2018. – Pp. 3.2.13 – 3.2.17. *Здобувач презентував обґрунтування використання дискретного вейвлет-перетворення при обробці часових рядів навантаженості вебсервера*.

10. Радченко К. О. Особливості архітектури нейромережевої моделі розпізнавання кібератак / К. О. Радченко. // *Фізико-технологічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах: Матеріали VII Міжнародної науково-практична конференції*. – Чернівці: «Місто». – 2018. С. 149 – 150. *Здобувач презентував особливості архітектури нейромережевої моделі під час обробки часових рядів навантаження на вебсервер*.

11. Радченко К.О. Концепція прогнозування навантаженості Вебсерверів за допомогою вейвлет-перетворень / К.О. Радченко, О.І. Терейковський, К.О. Харламов. // *Матеріали науково-практичної конференції «Сучасні інформаційні технології та кібербезпека»*. – К.: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. – С. 205 – 207. *Здобувач презентував концепцію прогнозування навантаженості вебсерверу з використанням вейвлет-перетворень, виконав підбір методів; Терейковський О.І. здійснив практичний аналіз стійкості та придатності вейвлет-базисів для задач обробки вебтрафіку; Харламов К.О. виконав порівняння характеристик різних типів вебнавантаження та їх впливу на роботу серверів*.

12. Radchenko K. Wavelet transforms for web server load calculating / Kostiantyn

Radchenko, Oleh Tereikovskyi // *ITSec: Безпека інформаційних технологій: IX міжнародна науково-технічна конференція*, 22-27 березня 2019 р. – К.: НАУ, 2019. – pp. 28 – 29. *Здобувач презентував особливості застосування вейвлет-перетворень під час аналізу рівня навантаженості вебсервера; О. Tereikovskyi здійснив математичний аналіз точності та стійкості перетворення*.

13. Радченко К.О. Аналіз самоподібності рівня вебтрафіку у комп'ютерних мережах загального призначення / К.О. Радченко, Л. О. Терейковська // *Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 25 листопада 2022 [Електронний ресурс]. – К: НУХТ, 2022. – С. 112 – 113. – Режим доступу: https://nuft.edu.ua/naukova-diyalnist/naukovi-konferencii. *Здобувач проаналізував рівень самоподібності вебтрафіку; Терейковська Л.О. надала методичний супровід щодо методів оцінювання самоподібності, виконала оцінювання коректності висновків.*

14. Стіренко С.Г., Радченко К.О., Апанасенко В.П. Прогнозування рівня навантаження на вебсервер комп'ютерних мереж загального призначення // *Кібербезпека державних інституцій та подолання кризових станів: матеріали І Міжнародної науково-практичної конференції* (Київ – Вроцлав, 26 травня 2022 р.). Київ, 2022. – С. 59 – 60. *Здобувач презентував концепцію використання вейвлет-перетворень під час прогнозування навантаження на вебсервер; Стіренко С.Г. визначив системні вимоги до моделі; Апанасенко В.П. виконав експериментальні дослідження*.

15. Радченко К.О. Прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень Добеші // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІРТК*-2023). *Шістнадцята міжнародна науково-практична конференція* 23 – 24 *травня* 2023 *р.,* Київ, Україна. – К.: НАУ, 2023. – С. 335 – 336. *Здобувач презентував прогнозування пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень*.

16. Радченко К.О. Виявлення різких змін рівня вебтрафіку за допомогою

вейвлет-перетворень Добеші // *Матеріали II Міжнародної науково-практичної конференції "Кібербезпека державних інституцій та подолання кризових станів" : в 2 т.* Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2023. Т. 1. С. 190. *Здобувач презентував визначення пікових значень рівня вебтрафіку за допомогою вейвлет-перетворень*.

17. Kostiantyn Radchenko, Xing Tao. Design and implementation of an automated Big Data analysis module // *Матеріали XI Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, 27 листопада 2024 [Електронний ресурс]. – К: НУХТ, 2024. С. 40 – 41. *Здобувач презентував особливості впровадження модульного підходу під час обробки великих масивів даних часових рядів; Xing Tao розробив архітектуру автоматизованого модуля Big Data analytics*.

18. K. Radchenko, I. Tereykovskyi, Xing Tao. Automated Big Data Analysis Module // *XVII Науково-практична конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг"* (*ПМК*-2024) (м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 20-22 листопада 2024 р.), С. 239 – 245. *Здобувач презентував архітектуру модульного підходу під час обробки великих масивів даних часових рядів; I. Tereykovskyi розробив логіку даних та оптимізацію методів фільтрації; Xing Tao здійснив технічну реалізацію прототипу*.

19. Романкевич В.О., Радченко К.О., Романенко М.В. Прогнозування цін акцій з використанням вейвлет-перетворення та нейронних мереж // *XVII Науково-практична конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг"* (*ПМК*-2024) (м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 20-22 листопада 2024 р.), С. 424 – 428. *Здобувач реалізував гібридний підхід до прогнозування часових рядів з використанням вейвлет-перетворення та нейронних мереж; Романкевич В.О. розробив постановку задачі; Романенко М.В. виконала навчання моделі*.

20. Радченко К.О., Статечний С.В. Адаптивні інтелектуальні системи для

динамічного керування навантаженням вебсерверів // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІІРТК*-2025). *Вісімнадцята міжнародна науково-практична конференція* 20-21 *травня* 2025 *р*., Київ, Україна. – К.: КАІ, 2025. – С. 375 – 377. *Здобувач розробив адаптивну інформаційну систему для динамічного керування навантаженням вебсерверів; Статечний С.В. допоміг з валідацією роботи системи у прикладних сценаріях.*

21. Радченко К.О., Черненький А. О. Прогнозування вебнавантаження для підвищення надійності та доступності обчислювальних систем // *Інтегровані інтелектуальні робототехнічні комплекси* (*ІІРТК*-2025). *Вісімнадцята міжнародна науково-практична конференція* 20-21 *травня* 2025 *р.*, Київ, Україна. – К.: КАІ, 2025. – С. 378 – 380. *Здобувач презентував метод прогнозування навантаження на вебсервери, що забезпечує превентивне балансування ресурсів для підтримки стабільної роботи обчислювальної інфраструктури; Черненький А.О. виконав експериментальне моделювання.*

22. Радченко К.О. Комп'ютерна програма «Wavelet analysis» / Терейковський Ігор Анатолійович; Радченко Костянтин Олександрович; Дичка Іван Андрійович; Романкевич Віталій Олексійович; Тарасенко-Клятченко Оксана Володимирівна – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» // Свідоцтво про реєстрацію авторського права на твір №7089 від 15.04.2024р. https://sis.nipo.gov.ua/en/search/detail/1803492/ Опубліковано 31.05.2024, бюл. №81. *Здобувач розробив модулі, що забезпечують автоматизоване прогнозування навантаження на вебсервери за допомогою гібридної моделі Wavelet Neural Network (WNN) та Long Short-Term Memory (LSTM), забезпечив тестування; Терейковський І. А. надав методичні рекомендації щодо побудови архітектури програмного забезпечення; Дичка І. А. структурував методи обробки сигналів та результати тестування; Романкевич В. О. структурував технічні характеристики, алгоритми роботи програми; Тарасенко-Клятченко О. В. здійснювала технічне та методичне редагування документації.*

ДОДАТОК З.

Акти впроваджень результатів дисертаційного дослідження

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного технічного університету
України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Сергій СТІРЕНКО
«12» 11 2025 р.

А К Т

впровадження результатів дисертаційного дослідження старшого викладача кафедри Системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «КПІ імені Ігоря Сікорського» Радченка Костянтина Олександровича на тему «Методи, моделі та засоби прогнозування навантаження на вебсервер» на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – декан факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», д.т.н., проф. Дичка І.А.; члени комісії – професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Романкевич О.М.; професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Терейковський І.А. цим Актом засвідчує, що результати дисертаційної роботи Радченка Костянтина використані співробітниками кафедри СПіСКС КПІ ім. Ігоря Сікорського при проведенні та викладанні курсу лекцій «Цифрова обробка сигналів та зображень».

Зокрема, впроваджено визначення найбільш ефективного типу базисного вейвлету, що включає новий метод вибору типу базисного вейвлету; розроблена лабораторна робота з вивчення пошуку періодичних коливань у часовому ряді вебнавантаження.

Голова комісії

д.т.н., проф. Іван ДИЧКА

Члени комісії

д.т.н., проф. Олексій РОМАНКЕВИЧ

д.т.н., проф. Ігор ТЕРЕЙКОВСЬКИЙ

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного технічного університету
України
«Київський політехнічний інститут
імені Ігоря Сікорського»

_____Сергій СТІРЕНКО
«7» ____11_____2025 р.

А К Т

про використання результатів дисертаційного дослідження старшого викладача кафедри Системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «КПІ імені Ігоря Сікорського» Радченка Костянтина Олександровича на тему «Методи, моделі та засоби прогнозування навантаження на вебсервер» на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – декан факультету прикладної математики «КПІ ім. Ігоря Сікорського», д.т.н., проф. Дичка І.А.; члени комісії – професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Романкевич О.М.; професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Терейковський І.А. цим Актом засвідчує, що результати дисертаційної роботи Радченка Костянтина отримані ним особисто та використані у:

- НДР «Методи, моделі та комп'ютерні засоби виявлення деструктивного впливу в медиапросторі», що виконується за ініціативою авторів, № держреєстрації 0121U110662, закінчення у 2023 р.
- НДР «Методи, моделі та засоби моніторингу закликів до тероризму у онлайн соціальних мережах», що виконується за ініціативою авторів, № держреєстрації 0124U003866, закінчення у 2026 р.

В НДР використані наступні результати:

- метод вибору типу базисного вейвлету, який за рахунок застосування запропонованих принципів та критеріїв оцінки ефективності типів базисного вейвлету дозволяє уникнути довготривалих експериментів, пов'язаних з визначенням найбільш ефективного типу базисного вейвлету.
- метод застосування вейвлет-перетворень для прогнозування навантаження на вебсервер в комп'ютерних мережах загального призначення, що дозволяє зменшити похибку прогнозування навантаження на вебсервер.

Голова комісії
д.т.н., проф. _____ Іван ДИЧКА

Члени комісії
д.т.н., проф. _____ Олексій РОМАНКЕВИЧ

д.т.н., проф. _____ Ігор ТЕРЕЙКОВСЬКИЙ

ЗАТВЕРДЖЕНО

Рішенням Ради

Федерації роботодавців ЖКГ України

№ 5/21 від «08» листопада 2021 р.

Голова Федерації роботодавців ЖКГ України Адамов О.І.

АКТ

впровадження результатів дисертаційної роботи

Даний акт підтверджує те, що результати наукових досліджень дисертаційної роботи Радченка Костянтина Олександровича використовуються у практичній діяльності Всеукраїнського об'єднання обласних організацій роботодавців підприємств житлово-комунального господарства «Федерація роботодавців ЖКГ України».

У функціонуванні Федерації роботодавців ЖКГ України було використано наступні результати дисертаційної роботи Радченка К.О. з тематики прогнозування навантаження на вебсервер:

- реалізований метод вибору найбільш ефективного типу базисного вейвлету при прогнозуванні навантаження на вебсервери об'єднання, дозволив приблизно в 1,5 рази зменшити обчислювальні витрати, пов'язані з визначенням типу базисного вейвлету;
- розроблене на базі запропонованого методу застосування вейвлет-перетворень спеціалізоване програмне забезпечення дозволило передбачити функціональні параметри вебсерверів і ефективно використати його для прогнозування з достатньою точністю навантаження на вебсервери об'єднання.

Результати дисертаційної роботи Радченка К.О. також можуть бути поширені та успішно використані в різноманітних організаціях, що займаються проєктуванням, експлуатацією та оцінкою показників надійності і захищеності комп'ютерних мереж загального призначення, у тому числі бюджетних установах, зокрема, на підприємствах житлово-комунального господарства.

Голова Федерації Адамов О.І.