НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
Кафедра математичних методів системного аналізу

На правах рукопису
УДК 303.732.4+004.8

ДО ЗАХИСТУ ДОПУЩЕНО
Завідувач кафедри ММСА
_____ Оксана ТИМОЩУК
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Метод прогнозування часового ряду на основі трансформерів та архітектур комп'ютерного зору»

Виконав:
студент ІІ курсу, групи КА-41мп
Свирид Нікіта Олександрович _____

Керівник:
Професор кафедри ММСА, д. т. н., доцент
Недашківська Надія Іванівна _____

Консультант з нормоконтролю:
Доцент кафедри ММСА, к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент:

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійною програмою «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
Оксана ТИМОЩУК
«___»_____ 2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Свириду Нікіті Олександровичу

1. Тема дисертації «Метод прогнозування часового ряду на основі трансформерів та архітектур комп’ютерного зору»,
науковий керівник дисертації Недашківська Надія Іванівна, професор кафедри ММСА, д. т. н., доцент, затверджені Наказом по університету від «06» листопада 2025 р. №4837-с.
2. Строк подання студентом дисертації: 19.12.2025.
3. Об’єкт дослідження: процес інтелектуального аналізу та прогнозування складних часових рядів засобами глибокого навчання.
4. Предмет дослідження: методи та моделі прогнозування, що базуються на інтеграції механізмів уваги та архітектур комп’ютерного зору для виявлення часових залежностей.
5. Перелік завдань, які потрібно розробити: здійснити аналіз сучасного стану проблеми прогнозування часових рядів та огляд існуючих підходів; обґрунтувати доцільність використання архітектур комп’ютерного зору в задачах часових рядів; розробити або адаптувати метод прогнозування, що поєднує трансформери та підходи комп’ютерного зору; здійснити підготовку наборів даних та їх попередню обробку; програмно реалізувати запропоновану модель та провести навчання; виконати порівняльний аналіз результатів роботи моделі з існуючими аналогами та зробити висновки.

6. Перелік графічного (ілюстративного) матеріалу: рисунки; блок-схеми; таблиці; схеми.

7. Орієнтовний перелік публікацій:

Свирид Н.О., Недашківська Н.І. Метод прогнозування часового ряду на основі архітектур комп'ютерного зору. Системні науки та інформатика: збірник доповідей IV Всеукраїнської науково-практичної конференції «Системні науки та інформатика», 01-05 грудня 2025 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2025.

8. Дата видачі завдання: 01.09.2025

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1.	Формування тематики дослідження	02.09.2025-04.09.2025	Виконано
2.	Попередня підготовка до написання МД: огляд літератури за обраною тематикою, постановка задачі	05.09.2025-08.09.2025	Виконано
3.	Підготовка до першого розділу МД: збір інформації вибір даних, аналіз можливостей побудови програми	09.09.2025-14.10.2025	Виконано
4.	Підготовка другого розділу МД: математичний опис використаних методів і програмних алгоритмів, побудова програми	15.10.2025-22.10.2025	Виконано
5.	Підготовка третього розділу МД: завершення побудови програми, кореляція отриманих результатів обраним методом, аналіз отриманих результатів	23.10.2025-05.11.2025	Виконано
6.	Підготовка четвертого розділу МД: розробка стартап-проєкту	06.11.2025-10.11.2025	Виконано
7.	Підготовка пояснювальної записки та презентації	11.11.2025-15.12.2025	Виконано

Студент _____ Нікіта СВИРИД

Науковий керівник дисертації _____ Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Магістерська дисертація: 109 с., 10 рис., 14 табл., 2 дод., 28 джерел.

ПРОГНОЗУВАННЯ, ЧАСОВІ РЯДИ, ГЛИБОКЕ НАВЧАННЯ, КОМП'ЮТЕРНИЙ ЗІР, ТРАНСФОРМЕР, CNN, GAF/MTF, СТАРТАП.

Метою даної роботи є підвищення якості прогнозування складних нелінійних процесів у соціально-економічних системах за умов високої нестаціонарності, волатильності та наявності прихованих структурних змін. У таких випадках часові ряди характеризуються хаотичною динамікою та режимними переходами, що знижує ефективність класичних методів регресії та актуалізує задачу розробки спеціалізованих гібридних підходів до прогнозування.

У роботі наведено огляд сучасних методів глибокого навчання для аналізу часових рядів, зокрема рекурентних нейронних мереж, трансформерних архітектур та методів візуального кодування сигналів. Запропоновано гібридний метод прогнозування, що поєднує підходи комп'ютерного зору та механізми уваги, орієнтований на виявлення прихованої часової структури та нелінійних залежностей. Метод базується на трансформації одновимірного часового сигналу у двовимірні зображення за допомогою алгоритмів Gramian Angular Field (GAF) та Markov Transition Field (MTF), вилученні локальних просторових ознак за допомогою згорткового енкодера (CNN) та моделюванні довгострокових часових залежностей із використанням блоку Трансформера.

У роботі представлено результати програмної реалізації запропонованої системи та серії експериментальних досліджень на синтетичних і реальних наборах даних, зокрема фінансових часових рядах. Порівняльний аналіз із базовими моделями показав, що запропонований метод демонструє конкурентні переваги у задачах моделювання нелінійних динамічних систем та ідентифікації ринкових режимів, водночас зберігаючи обмеження у задачах із домінуючою лінійною структурою.

ABSTRACT

Master’s thesis: 109 p., 10 fig., 14 tabl., 2 app., 28 references.

FORECASTING, TIME SERIES, DEEP LEARNING, COMPUTER VISION, TRANSFORMER, CNN, GAF/MTF, STARTUP.

The aim of this work is to improve the quality of forecasting complex nonlinear processes in socio-economic systems under conditions of high non-stationarity, volatility, and the presence of hidden structural changes. In such cases, time series are characterized by chaotic dynamics and regime shifts, which significantly reduce the effectiveness of classical regression methods and highlight the need for the development of specialized hybrid forecasting approaches.

This study presents a review of modern deep learning methods for time series analysis, including recurrent neural networks, transformer-based architectures, and visual signal encoding techniques. A hybrid forecasting method is proposed that combines computer vision approaches with attention mechanisms and is focused on uncovering hidden temporal structures and nonlinear dependencies. The method is based on transforming a one-dimensional time series into two-dimensional images using Gramian Angular Field (GAF) and Markov Transition Field (MTF) algorithms, extracting local spatial features through a convolutional encoder (CNN), and modeling long-term temporal dependencies using a Transformer block.

The paper presents the results of a software implementation of the proposed system and a series of experimental studies conducted on both synthetic and real-world datasets, including financial time series. Comparative analysis with baseline models demonstrates that the proposed method exhibits competitive advantages in modeling nonlinear dynamic systems and identifying market regimes, while maintaining limitations in tasks dominated by predominantly linear structures.

ЗМІСТ

ВСТУП

Ефективний аналіз та прогнозування часових рядів є одним із ключових завдань сучасної науки та прикладної аналітики, зокрема у фінансовій інженерії, енергетиці, промисловій діагностиці та соціально-економічних системах. Зі зростанням обсягів даних і складності досліджуваних процесів зростають вимоги до моделей прогнозування, які мають враховувати не лише трендові компоненти, але й складні нелінійні залежності, режимні переходи та короткочасні флуктуації. У таких умовах класичні статистичні методи та рекурентні нейронні мережі часто демонструють обмежену ефективність, особливо при роботі з довгими часовими послідовностями та високою нестаціонарністю даних.

Якісно новим етапом розвитку методів прогнозування стала поява трансформерних архітектур, що ґрунтуються на механізмі самоуваги. На відміну від рекурентних моделей, трансформери здатні безпосередньо моделювати залежності між віддаленими часовими моментами, розглядаючи часовий ряд як цілісну структуру. Водночас, специфіка багатьох реальних процесів полягає у поєднанні глобальних закономірностей із локальними патернами, такими як короткочасні коливання, піки, спади та повторювані структурні фрагменти. Для виявлення подібних локальних інваріантних ознак особливо ефективними є архітектури комп’ютерного зору, зокрема згорткові нейронні мережі.

Актуальність даного дослідження зумовлена необхідністю інтеграції переваг трансформерних моделей, які забезпечують захоплення глобального контексту, з можливостями методів комп’ютерного зору, орієнтованих на детальний аналіз локальної структури сигналів. Наукова проблема полягає у розробці гібридного підходу до прогнозування часових рядів, який дозволяє трансформувати одновимірні часові дані у двовимірні представлення, придатні для візуального аналізу, з подальшим моделюванням довгострокових залежностей, зберігаючи при цьому часову узгодженість та семантику процесу.

Метою магістерської дисертації є підвищення якості прогнозування часових рядів у задачах із вираженою нелінійною динамікою та структурною складністю шляхом розробки методу, що базується на синтезі трансформерних архітектур та підходів комп’ютерного зору. Практична цінність роботи полягає у створенні адаптивного інструментарію, здатного забезпечувати стабільні та інтерпретовані результати в умовах невизначеності, що є критично важливим для систем підтримки прийняття рішень. Отримані результати можуть бути використані у фінансовому аналізі, предиктивній діагностиці технічних систем та інших прикладних галузях, де точність і надійність прогнозу мають вирішальне значення.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ З ВИКОРИСТАННЯМ АРХІТЕКТУР ГЛИБОКОГО НАВЧАННЯ

Прогнозування часових рядів є однією з найважливіших складових сучасних систем підтримки прийняття рішень та інтелектуального аналізу даних. Відсутність точних прогнозів або використання застарілих методів загрожує значними фінансовими втратами та зниженням ефективності стратегічного планування.

Сучасні методи прогнозування передбачають не лише систематичний збір та обробку історичних даних, а й застосування передових архітектур глибокого навчання, зокрема трансформерів та моделей комп’ютерного зору, для виявлення складних, нелінійних залежностей у динамічних процесах.

1.1 Загальна характеристика задачі прогнозування часових рядів

Прогнозування часових рядів є однією з фундаментальних задач аналізу даних, що знаходить широке застосування у фінансових ринках, економічному плануванні, метеорології, енергетиці та управлінні складними технічними системами. В умовах стрімкого зростання обсягів даних та підвищення вимог до якості аналітичних рішень точність прогнозування набуває критичного значення для оптимізації процесів та підтримки прийняття рішень. Основною метою прогнозування часових рядів є використання історичних спостережень для передбачення майбутньої поведінки системи, при цьому визначальною властивістю таких даних є їх хронологічна впорядкованість та наявність кореляцій між послідовними значеннями [1].

Однією з ключових складностей аналізу часових рядів є їх стохастична природа та структурна неоднорідність. Класичний підхід передбачає подання

часового ряду у вигляді суми кількох компонентів, зокрема трендової складової, що відображає довгострокову динаміку, сезонної компоненти, яка описує регулярні періодичні коливання, та випадкового шуму. Приклад такої декомпозиції наведено на рисунку 1.1.

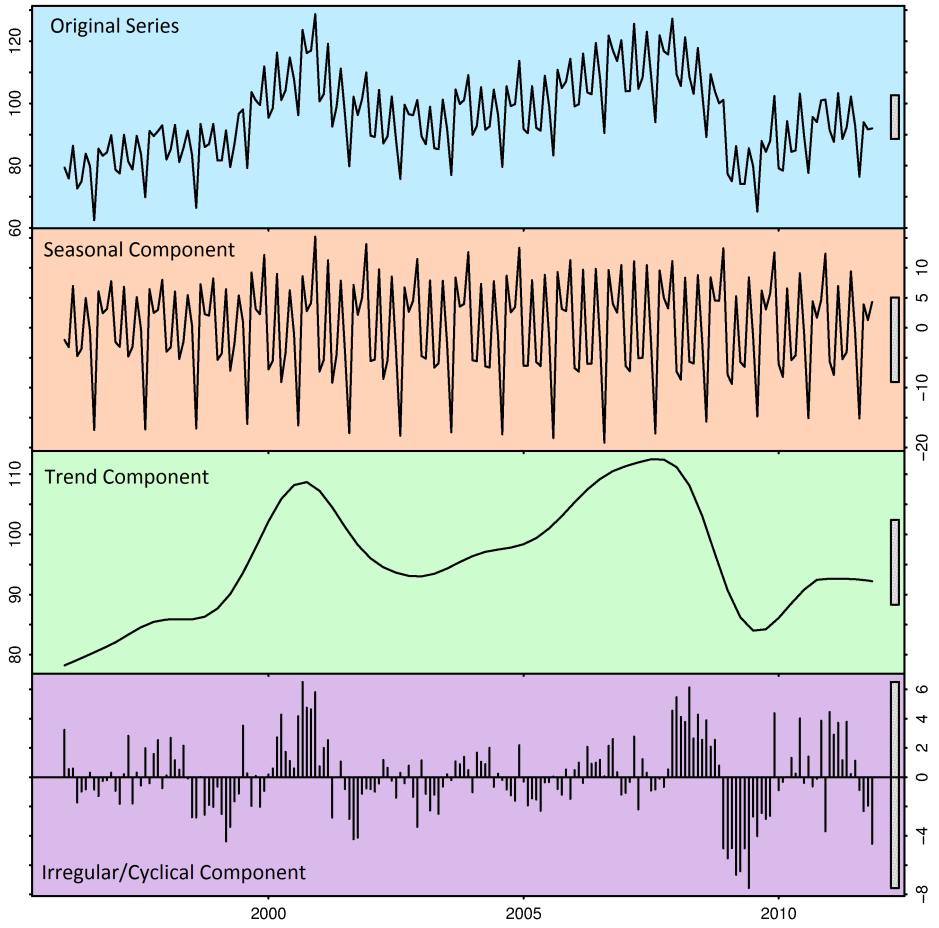


Рисунок 1.1 – Декомпозиція часового ряду на компоненти

Наявність цих компонентів у різних пропорціях зумовлює складність побудови універсальних моделей прогнозування.

Традиційні статистичні методи, зокрема експоненційне згладжування та моделі сімейства ARIMA, GARCH, тривалий час залишалися базовими інструментами аналізу часових рядів. Вони демонструють добрі результати за умови лінійності та відносної стаціонарності процесів, однак у випадку високої волатильності, нелінійної динаміки або структурних змін їх можливості суттєво

обмежуються. Це обумовлює зниження точності прогнозів при застосуванні класичних методів до сучасних реальних даних.

Обмеження статистичних підходів та зростання доступних обчислювальних ресурсів сприяли активному розвитку методів глибокого навчання для аналізу часових рядів. Нейронні мережі здатні автоматично вилучати складні нелінійні залежності без необхідності явного задання моделі процесу, що особливо актуально для багатовимірних часових рядів, де ручне проєктування ознак є складним або неможливим [2]. Водночас ефективність таких моделей значною мірою залежить від архітектури та способу представлення вхідних даних.

Окрему увагу в сучасних дослідженнях приділено архітектурам, запозиченим з суміжних галузей штучного інтелекту. Трансформерні моделі, що базуються на механізмі самоуваги, забезпечують ефективне врахування довгострокових залежностей у часових послідовностях, усуваючи обмеження, притаманні рекурентним підходам [3]. Паралельно з цим методи комп’ютерного зору пропонують альтернативний підхід до аналізу часових рядів, розглядаючи їх як візуальні патерни, що створює передумови для формування гібридних моделей, здатних поєднувати глобальний та локальний аналіз часової структури.

1.2 Огляд методів на основі рекурентних мереж та трансформерів

Еволюція методів глибокого навчання для прогнозування часових рядів тісно пов’язана з розвитком рекурентних нейронних мереж (Recurrent Neural Networks, RNN), які були спеціально розроблені для обробки послідовних даних. У таких моделях вихід у поточний момент часу залежить не лише від вхідного сигналу, а й від попередніх прихованих станів, що дозволяє враховувати часовий контекст. Проте на практиці класичні RNN виявилися малоефективними для

моделювання довготривалих залежностей через проблему зникаючого градієнта, яка ускладнює навчання на довгих послідовностях.

Значним кроком уперед стала архітектура довгої короткострокової пам’яті (Long Short-Term Memory, LSTM), запропонована С. Хохрайтером та Ю. Шмідхубером. Використання спеціальних керуючих елементів – вхідних, вихідних та забувальних воріт – дозволило ефективно контролювати потік інформації та зберігати релевантні ознаки протягом тривалого часу [4]. Завдяки цьому LSTM та їх модифікації, зокрема GRU, набули широкого застосування у задачах прогнозування часових рядів і продемонстрували високу стабільність у багатьох практичних сценаріях.

Водночас рекурентні архітектури мають принципове обмеження, пов’язане з послідовною природою обробки даних. Оскільки обчислення прихованого стану на кожному кроці часу залежить від результатів попередніх кроків, повне розпаралелювання процесу навчання є неможливим. Це призводить до значного збільшення часу тренування моделей на великих наборах даних та ускладнює їх масштабування у випадках, коли необхідно обробляти довгі часові послідовності або працювати в режимі близькому до реального часу.

Окрім обчислювальних обмежень, рекурентні мережі мають і структурні недоліки, пов’язані з ефективною довжиною контексту. Незважаючи на наявність механізмів пам’яті в архітектурах LSTM та GRU, на практиці здатність моделі зберігати та використовувати інформацію з віддалених часових моментів зменшується зі зростанням довжини послідовності. Це ускладнює моделювання процесів із довгостроковими залежностями, прихованими режимними переходами або складною багаторівневою динамікою, де важливу роль відіграють взаємозв’язки між подіями, рознесеними у часі.

Кардинальна зміна підходу до моделювання послідовностей відбулася з появою архітектури Transformer, яка відмовилася від рекурентних зв’язків на користь механізму самоуваги (Self-Attention). Цей механізм дозволяє оцінювати

взаємозв'язки між усіма елементами вхідної послідовності незалежно від їх відстані у часі, що забезпечує ефективне захоплення глобального контексту. Принцип роботи механізму самоуваги у задачах аналізу часових рядів схематично зображено на рисунку 1.2.

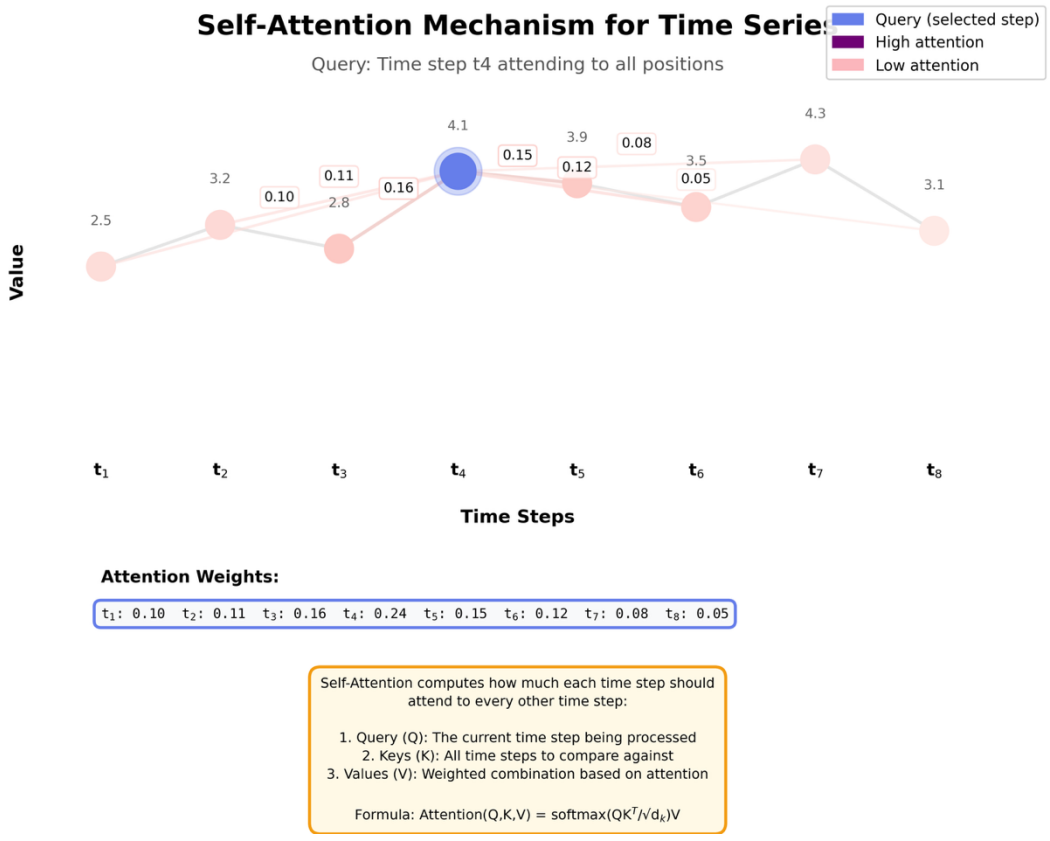


Рисунок 1.2 – Принцип роботи механізму Self-Attention для аналізу часового ряду

Спочатку трансформери були розроблені для задач обробки природної мови, однак згодом вони були адаптовані до прогнозування часових рядів. Разом з тим канонічний механізм самоуваги має квадратичну обчислювальну складність $O(L^2)$ відносно довжини послідовності, що обмежує його застосування для наддовгих рядів. Для подолання цієї проблеми були запропоновані спеціалізовані архітектури, зокрема Informer, Autoformer та Temporal Fusion Transformer (TFT), які спрямовані на зменшення обчислювальної складності та підвищення ефективності моделювання довгострокових залежностей [5].

Таким чином, рекурентні та трансформерні архітектури мають комплементарні властивості: перші забезпечують стабільне моделювання локальної динаміки, тоді як другі ефективно захоплюють глобальні часові залежності. Це створює передумови для розробки гібридних методів прогнозування, які поєднують сильні сторони обох підходів.

1.3 Архітектури комп’ютерного зору для часових рядів

Альтернативним та активно досліджуваним напрямком у задачах прогнозування часових рядів є адаптація методів комп’ютерного зору (Computer Vision) для аналізу одновимірних часових сигналів. Даний підхід ґрунтується на гіпотезі, що динамічні патерни у часових рядах – зокрема локальні тренди, різкі сплески, циклічні повторення та шумові компоненти – можуть бути інтерпретовані як візуальні ознаки, подібні до текстур, форм або контурів на зображеннях. Використання згорткових нейронних мереж дозволяє автоматично вилучати ієрархічні інваріантні ознаки з таких представлень, мінімізуючи потребу у ручному проєктуванні ознак.

Перший підхід до застосування архітектур комп’ютерного зору у часових рядах полягає у використанні одновимірних згорткових нейронних мереж (1D-CNN). На відміну від рекурентних моделей, які обробляють дані послідовно, 1D-CNN застосовують набір згорткових фільтрів, що ковзають уздовж часової осі та реагують на локальні патерни фіксованої довжини. Кожен фільтр може навчатися детектувати специфічні елементи динаміки, такі як зростаючі або спадні тренди, пікові значення чи локальні аномалії.

Принцип роботи 1D-CNN у задачах аналізу часових рядів схематично зображено на рисунку 1.3.

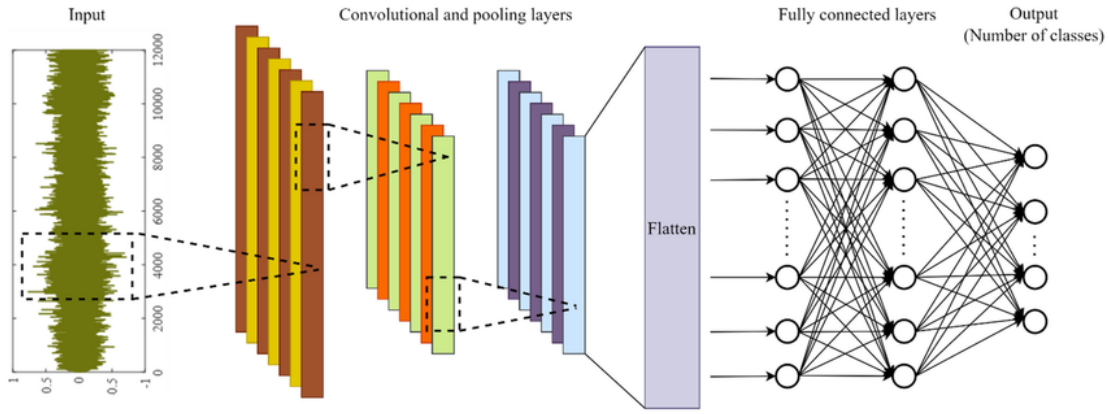


Рисунок 1.3 – Схема роботи 1D-CNN

Сучасні глибокі архітектури на основі 1D-CNN, зокрема InceptionTime та ResNet-подібні мережі із залишковими зв’язками, демонструють високу ефективність у задачах класифікації та прогнозування часових рядів. Завдяки можливості паралелізації обчислень такі моделі забезпечують значно вищу швидкість навчання у порівнянні з рекурентними підходами, зберігаючи при цьому здатність до виявлення локальних закономірностей [6]. Важливою властивістю 1D-CNN є інваріантність до зсуву в часі, що дозволяє розпізнавати характерні патерни незалежно від їх позиції у послідовності.

Другий, більш концептуально відмінний підхід полягає у трансформації одновимірного часового ряду у двовимірне зображення. Така стратегія дозволяє застосовувати повний спектр методів комп’ютерного зору, включно з архітектурами, спочатку розробленими для аналізу зображень, такими як VGG, EfficientNet або Vision Transformers. Для кодування часових рядів у вигляді зображень було запропоновано декілька методів, кожен з яких відображає різні аспекти динаміки системи.

Одним із поширених методів є граміанні кутові поля (Gramian Angular Fields, GAF), які базуються на представленні нормалізованого часового ряду у полярній системі координат. Значення сигналу кодується як кут, а часовий індекс

– як радіальна координата, що дозволяє зберігати часову кореляцію у вигляді тригонометричних співвідношень. Отримана матриця GAF є двовимірним зображенням, у якому глобальні та локальні залежності сигналу представлені у вигляді характерних геометричних структур. За певних умов дане перетворення є інформативно зворотним, що дозволяє мінімізувати втрати інформації [7].

Іншим методом є марковські поля переходів (Markov Transition Fields, MTF), які фокусуються на моделюванні динаміки переходів між дискретизованими станами часового ряду. У цьому підході сигнал поділяється на квантилі, після чого будується матриця ймовірностей переходів між відповідними станами. Отримане зображення відображає статистику переходів у часі, що робить метод MTF особливо придатним для аналізу стохастичних процесів та рядів з високою волатильністю.

Ще одним способом візуалізації часових рядів є рекурентні діаграми (Recurrence Plots, RP), які відображають повторюваність станів системи у фазовому просторі. Рекурентна діаграма формується шляхом порівняння станів системи у різні моменти часу та дозволяє виявляти періодичність, хаотичну поведінку та структурні зміни у сигналі. Періодичні процеси на таких зображеннях проявляються у вигляді діагональних структур, тоді як хаотичні сигнали формують складні текстурні патерни [8]. Приклад кодування часового ряду з шумом за допомогою методів GAF, MTF та RP наведено на рисунку 1.4.

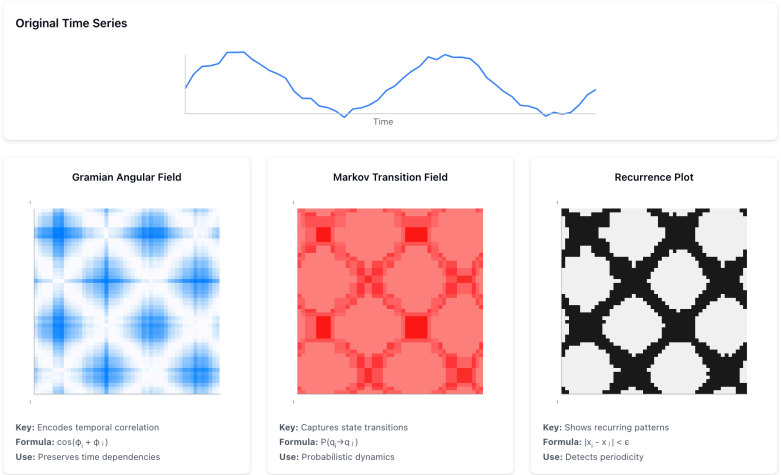


Рисунок 1.4 – Приклад кодування ряду за допомогою методів GAF, MTF та RP

Використання візуальних представлень часових рядів відкриває можливості для застосування методів трансферного навчання. Глибокі нейронні мережі, попередньо навчені на великих наборах зображень, здатні ефективно розпізнавати базові геометричні примітиви, такі як лінії, контури та текстури. Оскільки візуалізовані часові ряди складаються з подібних структур, ці знання можуть бути перенесені на задачі аналізу та прогнозування часових даних. Це дозволяє підвищити ефективність навчання моделей, особливо у випадках обмеженої кількості історичних спостережень [9].

Таким чином, архітектури комп'ютерного зору забезпечують потужний інструментарій для аналізу локальних та структурних властивостей часових рядів, проте самі по собі не враховують явну часову впорядкованість та довгострокові залежності. Це створює передумови для поєднання візуальних методів із трансформерними архітектурами, здатними ефективно моделювати глобальний контекст, що і становить основу гібридного підходу, запропонованого у даній роботі.

1.4 Порівняльний аналіз існуючих підходів до аналізу часових рядів

Аналіз сучасних підходів до прогнозування часових рядів, проведений у попередніх підрозділах, дозволяє систематизувати наявні методи та визначити їх ключові переваги й обмеження. Вибір конкретної моделі прогнозування завжди базується на пошуку компромісу між кількома взаємопов'язаними критеріями, зокрема точністю прогнозу, здатністю моделі відтворювати нелінійну динаміку, обчислювальною складністю та вимогами до обсягу навчальних даних. Додатково на ефективність моделей суттєво впливають властивості самих часових рядів, такі як стаціонарність, сезонність, рівень шуму та наявність структурних змін. У цьому контексті ізольоване порівняння моделей за однією

метрикою без урахування прикладних умов є недостатнім для обґрунтованого вибору архітектури.

Традиційні статистичні методи, зокрема моделі сімейства ARIMA, SARIMA та GARCH, залишаються ефективними інструментами для аналізу часових рядів зі відносно простою структурою та обмеженою кількістю змінних. Їх основними перевагами є висока інтерпретованість, низькі обчислювальні витрати та стабільність роботи на малих вибірках. Водночас такі методи демонструють обмежену здатність до моделювання нелінійних залежностей та погано масштабуються на високорозмірні або зашумлені дані.

Рекурентні нейронні мережі, зокрема LSTM та GRU, дозволяють ефективно моделювати нелінійні часові залежності та враховувати послідовний характер даних. Ці архітектури продемонстрували високу практичну ефективність у багатьох прикладних задачах прогнозування. Проте їх послідовна структура обмежує можливості паралелізації обчислень, що ускладнює масштабування та знижує ефективність роботи з дуже довгими часовими рядами.

Трансформерні архітектури, засновані на механізмах уваги, забезпечують принципово новий підхід до аналізу часових послідовностей, дозволяючи ефективно враховувати глобальні залежності між віддаленими часовими моментами. Завдяки можливості паралельної обробки даних такі моделі є добре масштабованими, однак потребують значних обчислювальних ресурсів та великих обсягів навчальних даних. Крім того, у базових реалізаціях трансформери можуть недостатньо чутливо реагувати на локальні динамічні патерни без спеціалізованих механізмів або архітектурних модифікацій.

Методи комп’ютерного зору, зокрема згорткові нейронні мережі та підходи на основі візуального кодування часових рядів, демонструють високу ефективність у вилученні локальних інваріантних ознак та аналізі структурних елементів сигналу. Водночас вони потребують додаткової попередньої обробки даних та не забезпечують явного моделювання часової послідовності без

інтеграції з іншими архітектурами. Узагальнене порівняння розглянутих підходів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика підходів до прогнозування часових рядів

Група методів	Сильні сторони	Обмеження
Статистичні методи	Висока інтерпретованість; низькі обчислювальні витрати; стабільність на малих вибірках	Обмежена здатність до моделювання нелінійних процесів; чутливість до шуму та викидів; потреба у попередній стаціонаризації
Рекурентні нейронні мережі	Моделювання нелінійних часових залежностей; урахування послідовної структури; гнучкість довжини вхідних даних	Послідовна обробка (обмежене розпаралелювання); зниження ефективності на дуже довгих послідовностях; складність масштабування
Трансформерні архітектури	Ефективне моделювання глобальних та довгострокових залежностей; паралельна обробка; висока масштабованість	Висока обчислювальна складність; потреба у великих наборах даних; можливі втрати локальних патернів без спеціальних механізмів
Методи комп'ютерного зору	Ефективне вилучення локальних інваріантних ознак; висока швидкість навчання; можливість трансферного навчання	Відсутність явного моделювання часової послідовності; залежність від способу кодування даних

Як випливає з наведеного аналізу, жоден із підходів у чистому вигляді не є універсальним рішенням для всіх типів часових рядів. Цей висновок узгоджується з теоремою «про відсутність безкоштовних сніданків» (No Free Lunch Theorem), відповідно до якої жоден алгоритм не має гарантованої переваги на всіх можливих класах задач [12]. Отже, досягнення високої точності прогнозування потребує розробки спеціалізованих моделей, адаптованих до структурних властивостей конкретних даних.

У цьому контексті перспективним напрямком є створення гібридних архітектур, що поєднують сильні сторони різних підходів. Зокрема, інтеграція згорткових мереж для вилучення локальних ознак із трансформерними

механізмами моделювання глобального контексту дозволяє ефективно використовувати доменні знання про структуру часових рядів. Саме такий синергетичний підхід лежить в основі методу, запропонованого у даній роботі [13].

1.5 Висновки до розділу 1

У першому розділі виконано комплексний аналіз сучасного стану проблеми прогнозування часових рядів та основних підходів до її розв'язання. Розглянуто класичні статистичні методи, рекурентні нейронні мережі, трансформерні архітектури та методи комп'ютерного зору, що дозволило систематизувати їхні переваги, обмеження та типові області застосування.

Показано, що статистичні моделі, такі як ARIMA, GARCH та ETS, є ефективними для задач зі відносно простою структурою даних і обмеженим обсягом вибірки, проте їх можливості суттєво обмежуються у випадках нелінійної динаміки та високої розмірності даних. Рекурентні нейронні мережі, зокрема LSTM, забезпечують більш гнучке моделювання часових залежностей та добре працюють із локальною динамікою сигналу, однак послідовна природа їх обчислень ускладнює масштабування та ефективне врахування довгострокового контексту.

Аналіз трансформерних архітектур засвідчив їх високу здатність до моделювання глобальних часових залежностей завдяки механізму самоуваги, що робить їх перспективними для задач довгострокового прогнозування. Водночас такі моделі є вимогливими до обчислювальних ресурсів та обсягу навчальних даних і можуть демонструвати знижену чутливість до локальних динамічних патернів без додаткових архітектурних модифікацій. Методи комп'ютерного зору, у свою чергу, забезпечують ефективне вилучення локальних та структурних ознак

із часових рядів, проте не враховують явну часову впорядкованість даних у відриві від інших механізмів.

Таким чином, проведений аналіз підтвердив відсутність універсального підходу до прогнозування часових рядів, що узгоджується з теоремою про відсутність «безкоштовних сніданків». Це зумовлює доцільність розробки спеціалізованих моделей, адаптованих до структурних особливостей конкретного класу задач.

На основі отриманих висновків обґрунтовано доцільність застосування гібридного підходу, який поєднує здатність згорткових мереж до вилучення локальних візуальних ознак із можливостями трансформерних архітектур щодо моделювання глобального часового контексту. Подальше дослідження у межах даної магістерської дисертації буде присвячене розробці, формалізації та програмній реалізації такої гібридної моделі, а також експериментальній оцінці її ефективності у порівнянні з існуючими методами прогнозування часових рядів.

РОЗДІЛ 2 РОЗРОБКА ГІБРИДНОГО МЕТОДУ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПРОГНОЗУВАННЯ

2.1 Обґрунтування вибору архітектури гібридної моделі

Як було показано у першому розділі, прогнозування складних часових рядів потребує одночасного врахування двох принципово різних типів залежностей: локальних, що відображають короткострокові флуктуації та структурні патерни сигналу, та глобальних, пов’язаних із довгостроковими трендами і режимними змінами. Класичні підходи до прогнозування, як правило, подають часовий ряд на вхід моделі у вигляді одновимірної послідовності числових значень, що суттєво обмежує можливості використання архітектур комп’ютерного зору, розроблених для аналізу просторових структур у двовимірних даних.

У даній роботі пропонується гібридний підхід до прогнозування часових рядів, який долає зазначене обмеження шляхом попереднього візуального кодування одновимірного сигналу у двовимірне представлення. Така трансформація дозволяє інтерпретувати часову динаміку у вигляді просторових патернів та застосовувати потужні згорткові архітектури для вилучення локальних ознак, зберігаючи при цьому інформацію про часову структуру даних.

Загальна структура пропонованого методу складається з трьох етапів:

1) трансформація одновимірного часового ряду у двовимірне матричне представлення у вигляді зображення;

2) вилучення локальних та структурних ознак за допомогою згорткової нейронної мережі (CNN);

3) моделювання часових залежностей та формування прогнозу з використанням механізму самоуваги (Transformer).

На першому етапі для візуального кодування часового ряду обрано методи Gramian Angular Field (GAF) та Markov Transition Field (MTF). Такий вибір зумовлений їх здатністю зберігати часову кореляцію у просторовій структурі

зображення та відображати різні аспекти динаміки сигналу. Метод GAF дозволяє кодувати значення часового ряду через тригонометричні перетворення у полярній системі координат, зберігаючи інформацію про відносні амплітуди та глобальні залежності між часовими моментами. У свою чергу, MTF фокусується на моделюванні динаміки переходів між дискретизованими станами сигналу, що є особливо важливим для аналізу волатильності та стохастичних процесів.

Застосування цих методів кодування дозволяє представити часовий ряд у вигляді двовимірних зображень, до яких можуть бути застосовані 2D-згорткові фільтри. Це відкриває можливість виявлення складних просторових патернів, які відповідають характерним режимам поведінки системи (наприклад, фазам зростання, спаду або підвищеної нестабільності), що є недоступним для класичних статистичних моделей або одновимірних нейронних архітектур.

На другому етапі отримані візуальні представлення обробляються згортковим енкодером (CNN), основним завданням якого є вилучення локальних інваріантних ознак та зменшення розмірності вхідних даних. У результаті формується компактний вектор високорівневих ознак (embedding), який узагальнено описує стан системи у відповідний момент часу. Такий підхід дозволяє ефективно відфільтровувати шумові компоненти та передавати на наступний етап лише найбільш інформативні характеристики сигналу.

Фінальним етапом архітектури є моделювання часових залежностей за допомогою трансформерного блоку. Оскільки згорткові мережі мають обмежене рецептивне поле та орієнтовані переважно на локальні структури, вони можуть втрачати інформацію про довгострокові залежності між віддаленими часовими моментами. Механізм самоуваги, реалізований у трансформері, компенсує цей недолік, дозволяючи встановлювати зв’язки між векторами ознак, отриманими від CNN, незалежно від їхньої позиції у часовій послідовності.

Таким чином, запропонована гібридна архітектура поєднує переваги методів комп’ютерного зору та трансформерних моделей, забезпечуючи

одночасний аналіз локальних патернів та глобального часового контексту. Концептуальну схему взаємодії основних компонентів запропонованого підходу наведено на рисунку 2.1.

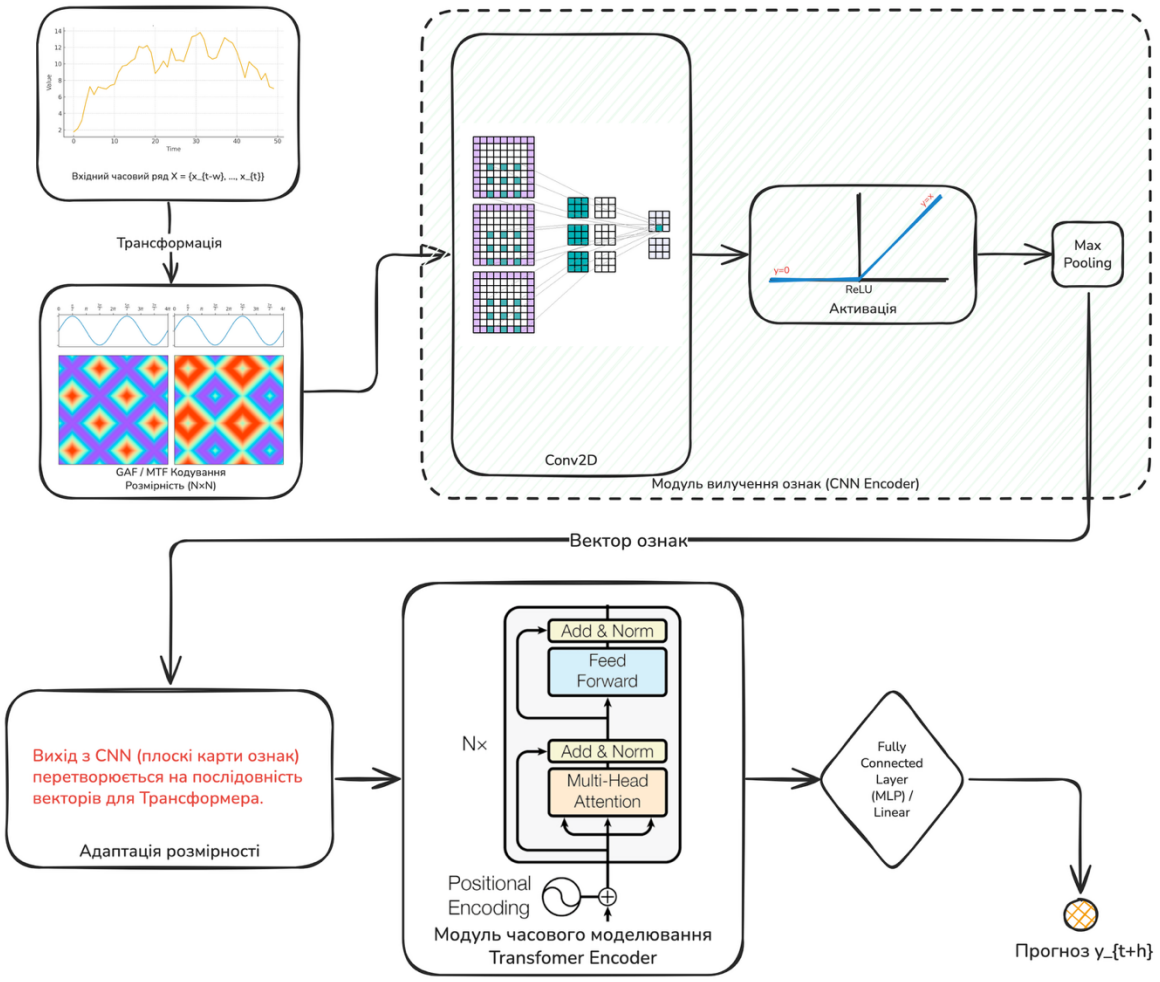


Рисунок 2.1 – Концептуальна схема гібридного підходу

2.2 Математична постановка задачі та формалізація методів перетворення даних

Нехай задано одновимірний часовий ряд $X = \{x_1, x_2, \dots, x_T\}$, де $x_t \in \mathbb{R}$ – значення процесу, що спостерігається в момент часу t, а T – довжина доступної історії спостережень.

Задача прогнозування часових рядів полягає у побудові апроксимуючої функції

$$\widehat{Y_t} = F(X_t, \theta),$$

яка на основі вхідного вікна спостережень довжини w (look-back window)

$$X_t = \{x_{t-w+1}, x_{t-w+2}, \dots, x_t\}$$

оцінює майбутні значення ряду на горизонт прогнозування h.

$$Y_t = \{x_{t+1}, x_{t+2}, \dots, x_{t+h}\}.$$

Тут θ – параметри моделі. Навчання зводиться до мінімізації функціоналу втрат $\mathcal{L}$ між прогнозованими $\widehat{Y_t}$ та істинними значеннями Y_t.

2.2.1 Попередня обробка та стабілізація даних (RevIN)

Часові ряди реального світу часто є нестаціонарними та характеризуються значними коливаннями масштабу. Для зменшення впливу цих факторів у роботі використовується підхід Reversible Instance Normalization (RevIN).

Для кожного вхідного вікна X_t виконується нормалізація:

$$\widetilde{x_i} = \frac{x_i - \mu(X_t)}{\sigma(X_t) + \varepsilon},$$

де $\mu(X_t)$ та $\sigma(X_t)$ – середнє значення та стандартне відхилення у поточному вікні, а ε – мала константа для числової стабільності.

Після формування прогнозу застосовується зворотне перетворення, що відновлює значення у початковому масштабі. Такий підхід дозволяє зберігати відносну динаміку сигналу та зменшувати чутливість моделі до зміни розподілу даних.

2.2.2 Масштабування для візуального кодування

Для застосування методів візуального кодування (GAF, MTF) значення сигналу додатково приводяться до інтервалу $[-1,1]$ за допомогою Min–Max нормалізації у межах вікна:

$$\widehat{x_i} = 2 \cdot \frac{x_i - \min(X_t)}{\max(X_t) - \min(X_t)} - 1.$$

2.2.3 Математична модель Gramian Angular Field (GAF)

Метод Gramian Angular Field ґрунтується на представленні нормалізованого сигналу у полярній системі координат. Для кожного значення $\hat{x}_i$ визначається кут:

$$\phi_i = \arccos(\widehat{x_i}), \quad \widehat{x_i} \in [-1,1],$$

а часовий індекс масштабується як радіус $r_i = \frac{i}{w}$.

На основі отриманих кутів формується матриця Грама:

$$G_{i,j} = \cos(\phi_i + \phi_j) = \widetilde{x_i}\widetilde{x_j} - \sqrt{1 - \widetilde{x_i^2}}\sqrt{1 - \widetilde{x_j^2}},$$

яка відображає кореляційні залежності між різними моментами часу у межах вікна. Отримана матриця

$$G \in \mathbb{R}^{w \times w}$$

використовується як вхідне зображення для згорткового енкодера.

2.2.4 Математична модель Markov Transition Field (MTF)

Метод Markov Transition Field спрямований на кодування динаміки переходів між станами часового ряду. Діапазон значень сигналу розбивається на Q квантильних інтервалів, після чого кожному значенню x_i ставиться у відповідність стан q_i. Будується матриця суміжності W розміром $Q \times Q$, де елемент w_{ij} відображає ймовірність переходу значення зі стану i у стан j.

Матриця MTF формується як

$$M_{i,j} = W_{q_i,q_j},$$

що дозволяє зберегти часову структуру переходів у вигляді двовимірного зображення.

2.2.5 Позиційне кодування (Positional Encoding)

Оскільки механізм самоуваги інваріантний до порядку елементів, до послідовності ознак додається позиційне кодування у вигляді синусоїдальних функцій:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right),$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right),$$

де pos – позиція вектора у послідовності, i – індекс виміру, d_{model} – розмірність прихованого простору моделі.

2.2.6 Задача оптимізації та функція втрат

Навчання гібридної моделі зводиться до оптимізації параметрів θ згорткового енкодера та трансформерного блоку за допомогою алгоритму Adam. В якості цільової функції використовується середньоквадратична помилка:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \tilde{y}_i(\theta)\right)^2 \rightarrow \min_{\theta}.$$

2.3 Структурна схема та математичний опис складових моделі

Розроблена гібридна модель прогнозування являє собою багаторівневу композицію диференційовних відображень, які послідовно трансформують вхідний простір часового ряду у простір прихованих ознак, а згодом – у простір прогнозних значень. Такий підхід дозволяє розкласти складну задачу прогнозування на низку взаємопов’язаних підзадач, кожна з яких вирішується спеціалізованим архітектурним модулем із чітко визначеною функціональною роллю.

Ключовою ідеєю запропонованої архітектури є розділення обробки локальної та глобальної інформації. Локальні короткострокові патерни часової динаміки обробляються на рівні згорткового енкодера, тоді як довгострокові залежності та взаємозв’язки між різними часовими сегментами моделюються за допомогою механізму уваги. Така ієрархічна організація дозволяє суттєво зменшити складність задачі, що покладається на кожен окремий компонент моделі.

На відміну від класичних нейронних підходів, у яких часовий ряд безпосередньо подається у вигляді одномірного вектора числових значень, у даній роботі використовується візуально-орієнтоване представлення часових даних. Завдяки попередньому кодуванню часового ряду у двовимірні матричні

структури стає можливим застосування методів комп'ютерного зору. Це дозволяє розглядати часову динаміку як структурований образ, а не лише як послідовність чисел.

Архітектурно модель може бути формалізована у вигляді наступного відображення:

$$\hat{Y} = \Psi_{Trans}\left(\Phi_{CNN}\left(\Omega_{Enc}(X)\right)\right),$$

де Ω_{Enc} – модуль візуального кодування (GAF/MTF), Φ_{CNN} – згортковий енкодер для вилучення локальних патернів, Ψ_{Trans} – трансформер для моделювання глобальних залежностей.

Структурна схема взаємодії цих компонентів наведена на рисунку 2.2.

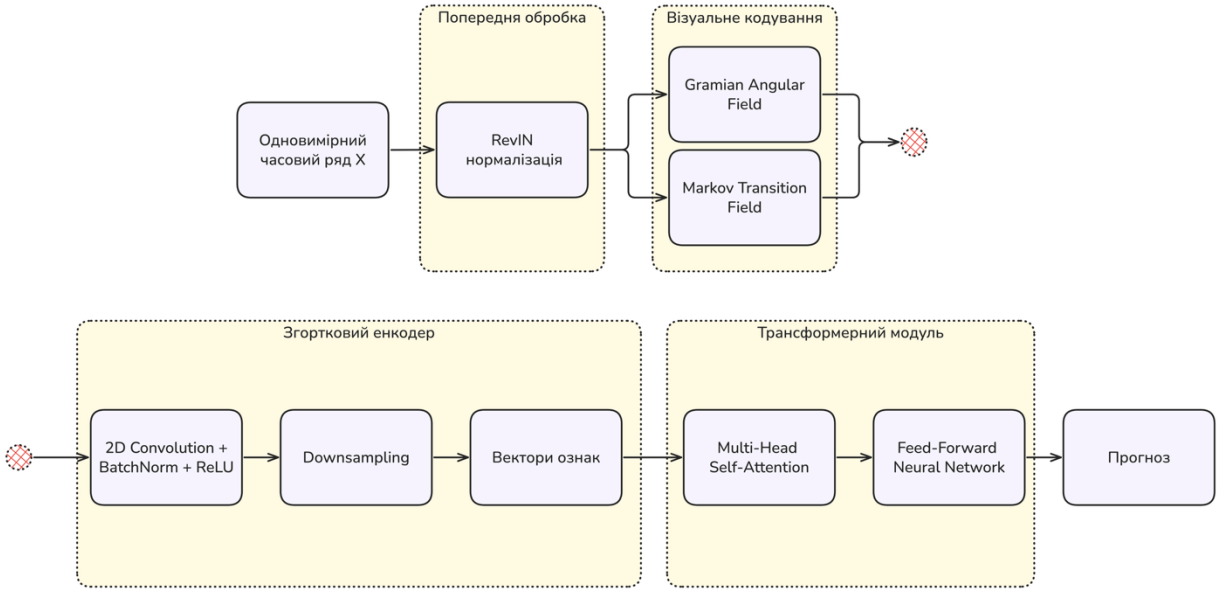


Рисунок 2.2 – Структурна схема гібридної моделі

2.3.1 Згортковий модуль вилучення ознак (CNN Encoder)

Після етапу візуального кодування часовий ряд подається на вхід згорткового енкодера у вигляді тензора

$$I \in R^{w \times w \times C},$$

де w – довжина вікна спостереження, а кількість каналів Cвідповідає використаним методам кодування (у даній роботі $C = 2$ – GAF та MTF).

Вибір згорткової архітектури зумовлений її здатністю до ефективного вилучення локальних інваріантних ознак. Завдяки механізму спільного використання ваг (shared weights), згорткові фільтри здатні детектувати характерні патерни (локальні тренди, піки, аномалії, фази волатильності) незалежно від їхнього положення у часовому вікні. Це є принциповою перевагою при аналізі нестаціонарних процесів, де одні й ті самі структури можуть з'являтися у різні моменти часу [15].

Математично операція двовимірної згортки для вхідної карти ознак Iта ядра фільтра K розміром $k \times k$ визначається як:

$$(I * K)_{ij} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} I_{i+m,j+n} \cdot K_{mn} + b$$

де b – зміщення (bias), яке дозволяє адаптувати поріг активації нейрона та підвищує гнучкість моделі.

Для стабілізації навчання та зменшення впливу ефекту внутрішнього зсуву коваріацій після кожного згорткового шару застосовується Batch Normalization [16]. Для міні-батчу $\mathcal{B} = \{x_1, \ldots, x_m\}$ обчислюються статистичні характеристики:

$$\mu_{\mathcal{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i\,, \quad \sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2.$$

Нормалізація виконується за формулою:

$$y_i = \gamma \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \beta,$$

де γ та β – параметри масштабування та зсуву, що навчаються разом із мережею, а ϵ – мала константа для забезпечення чисельної стабільності.

Для внесення нелінійності у модель використовується активаційна функція ReLU (Rectified Linear Unit), яка є обчислювально ефективною та не страждає від проблеми затухання градієнта у глибоких мережах [17]:

$$f(x) = \max(0, x).$$

У сукупності згортковий енкодер виконує роль механізму локальної абстракції, формуючи компактні вектори ознак, що узагальнюють локальну динаміку процесу та передають на наступний рівень лише найбільш інформативну частину сигналу.

2.3.2 Механізм уваги (Scaled Dot-Product Attention)

Ключовим елементом гібридної архітектури є трансформерний модуль, який відповідає за моделювання глобальних часових залежностей між ознаками, вилученими згортковим енкодером. На відміну від рекурентних мереж, трансформер не обробляє послідовність покроково, а аналізує її як цілісну структуру.

В основі роботи трансформера лежить механізм Scaled Dot-Product Attention, який дозволяє визначити ступінь впливу кожного елемента послідовності на всі інші. Нехай H – матриця прихованих представлень,

отриманих після CNN. Вона лінійно проектується у простори запитів, ключів та значень:

$$Q = HW^Q, \quad K = HW^K, \quad V = HW^V,$$

де W^Q, W^K, W^V – матриці параметрів, що навчаються.

Матриця уваги обчислюється як:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V.$$

де d_k – розмірність векторів ключів. Масштабування на $\sqrt{d_k}$ запобігає надмірному зростанню скалярних добутків, що могло б призвести до насичення функції Softmax і, як наслідок, до погіршення процесу навчання.

Для підвищення виразної здатності моделі використовується Multi-Head Attention**.** Замість одного механізму уваги модель паралельно обчислює h незалежних «голів», кожна з яких фокусується на різних аспектах часової динаміки:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O.$$

де кожна «голова» обчислюється незалежно:

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right).$$

Такий підхід дозволяє одночасно моделювати як короткострокові взаємозв’язки (наприклад, локальну волатильність), так і довгострокові тренди та режимні переходи. Після блоку уваги застосовуються Layer Normalization та повнозв’язна Feed-Forward Network, що завершує обробку в одному енкодерному блоці трансформера.

2.4 Алгоритм навчання та оптимізації параметрів моделі

Навчання розробленої гібридної нейронної мережі являє собою ітераційний процес пошуку оптимального набору параметрів θ (ваг та зміщень усіх шарів моделі), який мінімізує значення функції втрат $\mathcal{L}$ на навчальній вибірці.

Оскільки запропонована архітектура складається виключно з диференційовних компонентів (згорткові шари, механізми уваги, нормалізація та нелінійні активації), для її навчання використовується метод зворотного поширення помилки (Backpropagation).

Суть цього методу полягає у послідовному застосуванні ланцюгового правила диференціювання для обчислення градієнта функції втрат по відношенню до кожного параметра мережі, починаючи з вихідного шару та рухаючись у напрямку вхідних шарів. Такий підхід дозволяє кількісно оцінити внесок кожного нейрона та кожного шару у загальну помилку прогнозу і відповідним чином скоригувати параметри моделі [18].

У контексті гібридної архітектури з великою кількістю параметрів (CNN + Transformer) ефективна організація процесу оптимізації є критично важливою умовою досягнення стабільної збіжності та високої узагальнюючої здатності моделі.

2.4.1 Метод оптимізації Adam

Класичний стохастичний градієнтний спуск (SGD) у чистому вигляді часто виявляється неефективним для навчання глибоких гібридних архітектур. Причиною цього є складна форма поверхні функції втрат, що характеризується наявністю локальних мінімумів, плато та сідлових точок. У таких умовах фіксована швидкість навчання може призводити або до повільної збіжності, або до нестабільних коливань навколо мінімуму.

З огляду на це у даній роботі використовується адаптивний алгоритм оптимізації Adam (Adaptive Moment Estimation), який поєднує ідеї методу імпульсу та адаптивного підбору швидкості навчання для кожного параметра окремо. Adam підтримує експоненційні ковзні середні градієнта та його квадрата, що дозволяє стабільно працювати навіть за умов нестаціонарних градієнтів.

На кроці t алгоритм обчислює:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_{t,}^2$$

де $g_t = \nabla_\theta \mathcal{L}(\theta_{t-1})$ – градієнт функції втрат, а β_1, β_2 – коефіцієнти загасання (гіперпараметри, зазвичай 0.9 та 0.999).

Для усунення зміщення, пов'язаного з ініціалізацією моментів, застосовується корекція:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}, \widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

Остаточне правило оновлення параметрів має вигляд:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t},$$

де α – швидкість навчання, ϵ – мала константа для забезпечення чисельної стабільності.

Як показано в роботах Д. Кінгми та Дж. Ба, алгоритм Adam забезпечує швидку та стабільну збіжність навіть для моделей з великою кількістю параметрів, що робить його доцільним вибором для запропонованої гібридної архітектури.

2.4.2 Регуляризація (Dropout)

Глибокі нейронні мережі, зокрема гібридні архітектури, що поєднують CNN та Transformer, мають високу здатність до апроксимації складних функцій. Однак це також підвищує ризик перенавчання (overfitting)**,** коли модель надмірно підлаштовується під тренувальні дані та втрачає здатність до узагальнення.

Для зменшення цього ефекту у роботі використовується метод **Dropout**, який полягає у випадковому зануленні частини нейронів під час навчання з імовірністю p. Формально цей процес описується як:

$$y = f(W \cdot (x \odot r) + b),$$

де $r_i \sim Bernoulli(1-p)$ – випадкова бінарна маска, а $\odot$ – добуток Адамара. Застосування Dropout змушує мережу формувати надлишкові та більш стійкі представлення ознак, не покладаючись на окремі нейрони. У розробленій моделі Dropout використовується після згорткових шарів та всередині трансформерних блоків, що дозволяє знизити перенавчання без істотної втрати швидкості збіжності [19].

2.4.3 Загальний алгоритм навчання моделі

Процес навчання гібридної системи прогнозування реалізується за наступним алгоритмом:

1. Ініціалізувати параметри моделі θ (наприклад, за допомогою Xavier або He initialization).
2. Виконати пряме поширення (Forward Pass):
   1) перетворити батч часових рядів X у зображення (GAF/MTF);
   2) пропустити отримані тензори через згортковий енкодер та трансформерний модуль;

3) отримати прогноз $\hat{Y}$.

3. Розрахувати значення функції втрат $\mathcal{L} = MSE(Y, \hat{Y})$.

4. Обчислити градієнти $\nabla_\theta \mathcal{L}$ для всіх параметрів.

5. Застосувати крок оптимізатора Adam для корекції θ.

6. Повторювати кроки 2–5 протягом заданої кількості епох до досягнення критерію зупинки.

2.5 Критерії оцінювання якості прогнозування

Для об'єктивної та відтворюваної оцінки якості прогнозування розробленого гібридного методу у роботі використовується набір кількісних метрик, який повністю відповідає практичній реалізації експериментальної частини та сучасним підходам до аналізу часових рядів. Оскільки задача прогнозування формально належить до класу задач регресії, оцінювання проводиться за допомогою стандартних показників похибки, доповнених метриками напрямкової точності, що є критично важливими для фінансових та соціально-економічних застосувань. Такий підхід дозволяє оцінити не лише чисельну точність прогнозу, але й адекватність відтворення динаміки процесу.

Середня абсолютна похибка визначає середнє відхилення прогнозованих значень від істинних у вихідному масштабі даних і є стійкою до одиничних викидів. Вона обчислюється як

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|,$$

де y_i – істинне значення, $\hat{y}_i$ – прогноз моделі, N – кількість точок у тестовій вибірці. Дана метрика дозволяє легко інтерпретувати середню величину помилки та зручно порівнювати різні моделі на одному наборі даних.

Корінь середньоквадратичної похибки є однією з найбільш поширених метрик у задачах глибокого навчання та накладає більший штраф на великі відхилення прогнозу, що особливо важливо у випадках різких змін або сплесків волатильності. Завдяки квадратичному характеру похибки RMSE дозволяє оцінити стабільність моделі у критичних ситуаціях. Вона визначається формулою

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}.$$

Для оцінювання відносної точності прогнозу використовується симетрична середня абсолютна відсоткова похибка, яка є більш стабільною альтернативою класичної MAPE та менш чутливою до малих значень істинного сигналу. Симетрична форма дозволяє уникнути асиметрії при великих відхиленнях прогнозу, а також забезпечує коректне порівняння моделей на часових рядах з різними масштабами. sMAPE визначається як

$$\mathrm{sMAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{2|y_i - \widehat{y_i}|}{(|y_i| + |\widehat{y_i}|)} \cdot 100\%.$$

Коефіцієнт детермінації використовується для оцінки того, яку частку дисперсії реального процесу здатна пояснити модель. Він дозволяє оцінити якість апроксимації відносно простої базової моделі, що прогнозує середнє значення ряду, та визначається як

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2},$$

де $\bar{y}$ – середнє значення спостережуваного ряду. Значення R^2, близьке до одиниці, свідчить про високу узгодженість прогнозу з реальною динамікою процесу.

Окрім чисельної точності, у роботі особлива увага приділяється здатності моделі правильно відтворювати напрямок зміни часового ряду. Для цього використовується метрика напрямкової точності, яка визначає частку випадків, у яких знак прогнозованої зміни співпадає зі знаком реальної зміни процесу. Вона обчислюється за формулою

$$\mathrm{DA} = \frac{1}{N}\sum_{i=2}^{N} \mathbb{I}\left(sign(y_i - y_{i-1}) = sign(\hat{y}_i - y_{i-1})\right),$$

де $\mathbb{I}(\cdot)$ – індикаторна функція. Цей показник є особливо важливим для фінансових задач, де правильний напрямок руху активу часто має більшу практичну цінність, ніж мінімізація абсолютної похибки.

Додатково використовується показник Hit Rate, який інтерпретується як імовірність правильного «влучання» моделі у напрямок руху процесу на заданому горизонті прогнозування та визначається як відношення кількості правильних напрямкових прогнозів до загальної кількості прогнозів. У сукупності з Directional Accuracy ця метрика дозволяє оцінити поведінкову адекватність моделі та її практичну корисність у прикладних сценаріях.

Застосування зазначеного набору метрик дозволяє здійснити комплексну та збалансовану оцінку якості прогнозування, поєднуючи аналіз чисельної точності, відносної похибки, пояснювальної здатності моделі та коректності відтворення напрямку динаміки часового ряду.

2.6 Висновки до розділу 2

У другому розділі виконано теоретичне обґрунтування та математичну формалізацію запропонованого гібридного методу прогнозування часових рядів, орієнтованого на аналіз складних нелінійних та нестаціонарних процесів.

Розроблено архітектуру гібридної моделі, що базується на синергетичному поєднанні підходів комп’ютерного зору та механізмів уваги. Обґрунтовано доцільність використання багатоканального візуального представлення часового ряду, яке дозволяє одночасно аналізувати як статичні кореляційні структури сигналу за допомогою Gramian Angular Field, так і стохастичну динаміку переходів між станами через Markov Transition Field. Такий підхід забезпечує більш повне та інформативне представлення часової динаміки порівняно з класичним векторним поданням.

Виконано строге математичне формулювання задачі прогнозування з використанням ковзного вікна спостережень та багатокрокового горизонту прогнозу. Детально описано процедури попередньої обробки та нормалізації даних, а також формалізовано алгоритми перетворення одновимірного часового ряду у двовимірний матричний простір. Наведено математичний опис операцій згорткового енкодера, включаючи згортку, Batch Normalization та нелінійну активацію ReLU, що забезпечують ефективне вилучення локальних інваріантних ознак та зменшення впливу шуму.

Окрему увагу приділено моделюванню глобальних часових. Формалізовано механізм Scaled Dot-Product Attention та розширення Multi-Head Self-Attention, які дозволяють моделі одночасно враховувати різні аспекти часової структури даних. Розглянуто роль позиційного кодування як необхідного елементу для збереження порядку слідування часових кроків у процесі уваги.

Також визначено процедуру навчання гібридної моделі. Сформульовано алгоритм оптимізації параметрів мережі на основі методу зворотного поширення помилки з використанням адаптивного оптимізатора Adam. Обґрунтовано застосування регуляризації Dropout для зниження ризику перенавчання та підвищення узагальнюючої здатності моделі.

Загалом, розроблене математичне, архітектурне та алгоритмічне забезпечення створює цілісний фундамент для програмної реалізації гібридної

системи прогнозування та проведення експериментальних досліджень. Перевірка ефективності запропонованого підходу, а також його порівняння з базовими та state-of-the-art моделями буде здійснено у наступному розділі роботи.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ГІБРИДНОГО МЕТОДУ ПРОГНОЗУВАННЯ

Метою даного розділу є практична верифікація теоретичних положень та математичних моделей, розроблених у попередніх розділах роботи. Зокрема, розділ спрямований на перевірку ефективності запропонованого гібридного методу прогнозування часових рядів, який поєднує підходи комп'ютерного зору та трансформерні архітектури глибокого навчання.

Програмна реалізація розробленого методу потребує обґрунтованого вибору технологічного стеку, архітектурних компонентів та процедур обробки даних. У зв'язку з цим у розділі детально описано процес побудови програмного комплексу, включаючи вибір мови програмування, бібліотек машинного навчання, інструментів для обробки часових рядів та засобів візуалізації результатів. Окрему увагу приділено реалізації модулів візуального кодування часових рядів, згорткового енкодера та трансформерного блоку, а також інтеграції цих компонентів у єдину систему.

Експериментальна частина роботи охоплює дослідження поведінки моделей на як синтетичних, так і реальних наборах даних. Синтетичні часові ряди використовуються для контрольованої перевірки здатності моделей відтворювати відомі закономірності, такі як сезонність, нелінійні залежності та режимні переходи. Реальні набори даних, зокрема фінансові та часові ряди технічного характеру, дозволяють оцінити практичну придатність запропонованого підходу в умовах шуму, нестаціонарності та прихованих структурних змін.

Експериментальне дослідження спрямоване на порівняльний аналіз запропонованого гібридного методу з базовими та сучасними підходами до прогнозування часових рядів, зокрема моделями GARCH, LSTM та іншими нейронними архітектурами. Оцінювання якості прогнозування здійснюється за допомогою набору кількісних метрик, що дозволяє комплексно охарактеризувати

точність, стабільність та узагальнюючу здатність моделей. Отримані результати слугують підґрунтям для формулювання висновків щодо доцільності використання візуального кодування часових рядів у задачах прогнозування складних динамічних систем.

3.1 Обґрунтування вибору засобів розробки

Для практичної реалізації запропонованого у другому розділі гібридного методу прогнозування часових рядів, що базується на інтеграції згорткових нейронних мереж (CNN) та трансформерних архітектур, було обрано мову програмування Python версії 3.11. Такий вибір зумовлений широким застосуванням Python у сфері наукових досліджень, аналізу даних та машинного навчання, а також наявністю розвиненої екосистеми бібліотек для роботи з часовими рядами та глибокими нейронними мережами.

Архітектура програмного комплексу побудована з використанням сучасних інструментальних засобів, які забезпечують гнучкість реалізації, відтворюваність експериментів та ефективність обчислень.

Фреймворк PyTorch (версія 2.0) використано як основну платформу для побудови та навчання нейронних мереж. Його модульна структура дозволила реалізувати кастомну гібридну архітектуру, що поєднує двовимірні згорткові шари для вилучення локальних візуальних ознак та блоки трансформерного енкодера для моделювання глобальних часових залежностей. Підтримка динамічних обчислювальних графів значно спростила налагодження процесу зворотного поширення помилки та експерименти з різними конфігураціями архітектури [20].

Для реалізації етапу візуального кодування часових рядів використано бібліотеку PyTS (Python Time Series). Зокрема, застосовано класи для побудови граміанних кутових полів та MarkovTransitionField для формування марковських

полів переходів. Використання цих методів дозволило трансформувати одновимірні часові сигнали у двовимірні тензори зображень, придатні для обробки згортковими нейронними мережами [21].

Бібліотека Scikit-learn застосовувалась на етапах попередньої обробки даних, включаючи нормалізацію та масштабування часових рядів, а також для обчислення основних метрик якості прогнозування (MAE, RMSE, коефіцієнт детермінації R^2). Крім того, інструментарій Scikit-learn забезпечив коректне розділення даних на тренувальні, валідаційні та тестові підвибірки, що є необхідною умовою для об’єктивного експериментального аналізу [22].

Для формування реальних наборів даних у роботі використано бібліотеку yFinance, яка надає доступ до історичних фінансових часових рядів через API сервісу Yahoo Finance. Це дозволило автоматизувати процес збору даних та дослідити поведінку моделей в умовах ринкової нестаціонарності та високого рівня шуму [23].

Бібліотеки NumPy та Pandas забезпечили ефективну роботу з багатовимірними масивами даних і табличними структурами, що використовувались на етапах підготовки часових рядів, формування ковзних вікон спостереження та агрегації результатів експериментів [24].

Експериментальні дослідження проводилися з використанням апаратного прискорення, доступного в сучасних обчислювальних середовищах, з підтримкою обчислень на CPU та GPU. Реалізована програмна архітектура є платформонезалежною та допускає запуск як на класичних процесорах, так і на графічних або спеціалізованих обчислювальних пристроях, що забезпечує масштабованість та відтворюваність результатів.

3.2 Опис наборів даних

Для всебічної та об’єктивної оцінки ефективності запропонованого гібридного методу прогнозування у роботі було використано декілька наборів даних різної природи. Такий підхід дозволяє проаналізувати поведінку моделі як у контрольованих умовах (синтетичні часові ряди), так і в реальних стохастичних середовищах, характерних для соціально-економічних систем.

Усі датасети було приведено до уніфікованого формату одновимірних часових рядів та оброблено з використанням ковзного вікна фіксованої довжини для формування навчальних і тестових вибірок. Горизонт прогнозування складав декілька кроків уперед $(t+1, t+3, t+6, t+12)$, що дозволяє оцінити стабільність моделей на різних часових масштабах.

3.2.1 Синтетичні набори даних

Для об’єктивної оцінки якості моделі було застосовано синтетичні набори даних, які дозволяють проводити порівняльний аналіз моделей у ситуаціях, де відома природа сигналу, та оцінювати, наскільки складні архітектури виправдовують себе порівняно з базовими методами.

Датасет synth_chirp_phase. Даний датасет моделює нелінійний нестаціонарний процес із поступовою зміною частоти коливань (chirp-сигнал). Такий тип ряду характеризується сильною фазовою нелінійністю та є складним для класичних лінійних моделей. Використання цього датасету дозволяє оцінити, наскільки ефективно модель здатна захоплювати фазові переходи та глобальні часові структури.

Датасет synth_regime_switch. Цей набір даних імітує процес із перемиканням режимів (regime switching), коли часовий ряд переходить між різними статистичними станами з відмінними параметрами (середнє значення,

дисперсія, динаміка). Подібна поведінка характерна для фінансових ринків, макроекономічних індикаторів та поведінкових моделей. Датасет використовується для аналізу здатності моделей адаптуватися до структурних змін та ідентифікувати режими функціонування системи.

3.2.2 Реальні фінансові часові ряди

Для перевірки практичної придатності розробленого підходу використано реальні фінансові часові ряди, отримані з відкритих джерел.

Датасет real_AAPL. Датасет містить історичні значення ціни акцій компанії Apple Inc. (тикер AAPL). Фінансові часові ряди цього типу характеризуються високою волатильністю, шумом, наявністю трендів та короткострокових флуктуацій. Прогнозування таких даних є складною задачею через низьке відношення сигнал/шум та вплив зовнішніх факторів.

Датасет real_SPY. Даний набір даних представляє часовий ряд біржового індексного фонду SPDR S&P 500 ETF (тикер SPY), який відображає загальний стан фондового ринку США. На відміну від окремих акцій, цей ряд має більш згладжену динаміку, але все ж демонструє чітко виражені ринкові цикли, періоди підвищеної та зниженої волатильності.

Фінансові датасети використовувалися для оцінки моделей за практично значущими метриками, зокрема Directional Accuracy (DA) та Hit Rate, які відображають здатність правильно передбачати напрям руху ціни. Це є критично важливим для прикладних задач у трейдингу та ризик-менеджменті, де абсолютна точність прогнозу менш важлива, ніж коректне визначення тренду.

3.3 Попередня обробка наборів даних

Ефективність методів прогнозування часових рядів значною мірою визначається властивостями вхідних даних, зокрема їх стаціонарністю, рівнем шуму, наявністю нелінійних залежностей та структурних змін. Тому для всебічної оцінки запропонованого гібридного методу в роботі використано як синтетичні, так і реальні часові ряди, що дозволяє дослідити поведінку моделей у контрольованих та наближених до практичних умовах.

Синтетичні часові ряди використовуються для контрольованої перевірки здатності моделей відтворювати наперед задані закономірності. Загальний вигляд синтетичного часового ряду можна подати у вигляді композиції компонентів:

$$x_t = T_t + S_t + N_t$$

де T_t – трендова складова, S_t – сезонна або періодична компонента, N_t – випадковий шум або стохастичне збурення. У межах роботи розглянуто ряди з вираженою сезонністю, нелінійними залежностями та режимними переходами, у яких статистичні характеристики процесу змінюються в часі.

Для оцінки практичної придатності моделей було використано реальні часові ряди з відкритих джерел. Зокрема, у роботі розглянуто фінансові часові ряди, що відображають динаміку цін активів, а також технічні та кліматичні часові ряди. Такі дані зазвичай є нестаціонарними та можуть бути описані лише у локальному сенсі, тобто:

$$E[x_t] \neq const, \qquad Var(x_t) \neq const,$$

що створює складні умови для прогнозування та є характерною особливістю реальних прикладних задач.

Для всіх наборів даних застосовувалась єдина процедура формування навчальних прикладів. Початковий часовий ряд $X = \{x_1, x_2, \dots, x_T\}$ розбивався на ковзні вікна довжини w, які використовувались як вхідні послідовності:

$$X_t = \{x_{t-w+1}, x_{t-w+2}, \dots, x_t\},$$

а відповідні цільові значення формувалися на горизонті прогнозування h:

$$Y_t = \{x_{t+1}, x_{t+2}, \dots, x_{t+h}\}.$$

Такий підхід дозволяє розглядати задачу як багатокрокове прогнозування та оцінювати якість моделей на різних часових інтервалах.

Перед подачею даних на етап візуального кодування виконувалась нормалізація значень у межах кожного вікна спостереження. Це було необхідно як для стабільності навчання нейронних мереж, так і для коректної побудови візуальних представлень Gramian Angular Field та Markov Transition Field, які вимагають обмеженого діапазону значень. Нормалізація виконувалась таким чином, щоб зберігати відносну структуру сигналу та не використовувати інформацію з майбутніх часових кроків, що виключає витік даних.

Після нормалізації одновимірні часові ряди трансформувалися у двовимірні матричні представлення за допомогою методів GAF та MTF. Отримані матриці формували багатоканальний тензор, який подавався на вхід згорткового енкодера. Для моделей, що не використовують візуальне кодування (GARCH, LSTM), застосовувались ті самі ковзні вікна, але у вигляді класичних векторних послідовностей, що забезпечує коректність порівняльного аналізу.

Розділення даних на тренувальну, валідаційну та тестову вибірки виконувалося у хронологічному порядку:

$$X_{train} \prec X_{val} \prec X_{test},$$

без випадкового перемішування спостережень. Такий підхід відповідає реальним умовам прогнозування та забезпечує коректну оцінку узагальнюючої здатності моделей. Запропонована процедура підготовки даних створює єдині та відтворювані умови експериментів для всіх досліджуваних моделей і слугує надійним базисом для подальшого аналізу результатів прогнозування.

3.3 Реалізація та конфігурація моделей для порівняльного аналізу

Для об’єктивної оцінки ефективності запропонованого гібридного методу прогнозування у роботі було реалізовано та досліджено декілька моделей, що репрезентують різні підходи до аналізу часових рядів. Усі моделі навчалися та оцінювалися в однакових експериментальних умовах, із використанням ідентичних тренувальних, валідаційних і тестових вибірок, що забезпечує коректність порівняльного аналізу.

Як класичний статистичний базовий метод було використано модель GARCH. Її реалізація здійснювалась із застосуванням стандартних інструментів аналізу часових рядів. Модель налаштовувалась індивідуально для кожного набору даних з урахуванням стаціонарності ряду, порядку авторегресії та ковзного середнього. GARCH використовується у роботі як еталонний підхід, що дозволяє оцінити, наскільки складні нейронні моделі перевершують традиційні методи при наявності нелінійних залежностей та шуму.

Другим базовим методом для порівняння обрано довгокороткострокову пам’ять (LSTM), яка є одним із найбільш поширених підходів глибокого навчання для прогнозування часових рядів. Реалізована LSTM-модель складається з одного або декількох рекурентних шарів із прихованим станом фіксованої розмірності та повнозв’язного вихідного шару, що генерує багатокроковий прогноз. На вхід моделі подається послідовність значень ковзного вікна, а навчання здійснюється шляхом мінімізації середньоквадратичної помилки. LSTM використовується як

сильний нейронний базис, здатний моделювати нелінійні залежності без попереднього ручного конструювання ознак.

Окрім цього, для розширення порівняльного аналізу в окремих експериментах розглядались спрощені лінійні нейронні архітектури, які дозволяють оцінити внесок складності моделі у підсумкову якість прогнозу. Такі моделі виконують роль контрольного рівня, що відокремлює ефект складної архітектури від ефекту правильної підготовки даних.

Основним об’єктом дослідження у роботі є запропонована гібридна модель, яка поєднує візуальне кодування часових рядів, згортковий енкодер та трансформерний механізм уваги. Для цієї моделі на першому етапі одновимірні часові ряди трансформуються у двовимірні матричні представлення за допомогою Gramian Angular Field та Markov Transition Field. Отримані багатоканальні зображення подаються на вхід згорткового енкодера, який виконує вилучення локальних інваріантних патернів та зменшення розмірності даних. Далі сформована послідовність векторів ознак обробляється трансформерним енкодером, що моделює глобальні часові залежності та генерує прогноз на заданий горизонт.

Для забезпечення коректного порівняння всі нейронні моделі навчались з використанням однакової функції втрат, однакової процедури оптимізації та спільної стратегії ранньої зупинки за валідаційною вибіркою. Гіперпараметри моделей підбиралися таким чином, щоб забезпечити стабільну збіжність та уникнути перенавчання, але без агресивної оптимізації під конкретний датасет, що могло б спотворити результати експериментів.

Таким чином, реалізований набір моделей охоплює основні парадигми прогнозування часових рядів – від класичних статистичних методів до сучасних гібридних нейронних архітектур. Це створює надійний базис для подальшого експериментального аналізу та дозволяє об’єктивно оцінити переваги і обмеження запропонованого підходу у різних сценаріях прогнозування.

3.4 Методика проведення експериментального дослідження

Експериментальне дослідження в роботі проводилось з метою об’єктивної оцінки якості прогнозування запропонованого гібридного методу та його порівняння з базовими і сучасними підходами до аналізу часових рядів. Для забезпечення відтворюваності та коректності результатів було зафіксовано єдину методику навчання, тестування та оцінювання моделей.

Усі експерименти виконувались у постановці багатокрокового прогнозування. Для кожного ковзного вікна спостереження довжини wмоделі генерували прогноз на горизонті h, що дозволяє оцінити стабільність і деградацію точності зі збільшенням віддаленості прогнозу. У межах роботи розглядались горизонти прогнозування:

$$h \in \{1,3,6,12\},$$

що відповідає короткостроковому та середньостроковому прогнозу і є типовим для практичних задач фінансового та технічного аналізу.

Для кожного набору даних часовий ряд розділявся на тренувальну, валідаційну та тестову вибірки у хронологічному порядку. Навчання моделей здійснювалося виключно на тренувальній вибірці, підбір гіперпараметрів та рання зупинка – на валідаційній, а фінальна оцінка якості прогнозу – на тестовій вибірці. Такий підхід повністю виключає використання інформації з майбутніх часових моментів під час навчання та відповідає реальним умовам експлуатації прогнозних моделей.

Нейронні моделі (LSTM та гібридна CNN–Transformer архітектура) навчалися шляхом мінімізації середньоквадратичної помилки між прогнозованими та істинними значеннями. Навчання виконувалося протягом фіксованої максимальної кількості епох із застосуванням механізму ранньої зупинки (early stopping), який припиняв навчання у разі відсутності покращення

якості на валідаційній вибірці протягом заданої кількості ітерацій. Це дозволяло запобігти перенавчанню та забезпечити стабільну узагальнюючу здатність моделей.

Оцінювання якості прогнозування здійснювалося за допомогою набору взаємодоповнювальних метрик, які дозволяють аналізувати різні аспекти точності моделей. Для кожного горизонту прогнозування hобчислювались абсолютні та відносні помилки (MAE, RMSE, sMAPE), а також коефіцієнт детермінації R^2. Окрему увагу приділено напрямковим метрикам – Directional Accuracy (DA) та Hit Rate, які відображають здатність моделі коректно передбачати напрямок зміни часового ряду, що є критично важливим у фінансових застосуваннях.

Для багатокрокового прогнозу оцінка проводилась як по кожному горизонту окремо, так і у вигляді агрегованих показників, що усереднюють якість прогнозування на всьому інтервалі:

$$Metric_{all} = \frac{1}{h}\sum_{k=1}^{h} Metric_{t+k}.$$

Такий підхід дозволяє порівнювати моделі не лише за точністю на найближчому кроці, а й за здатністю зберігати адекватність прогнозу на віддалених часових інтервалах.

Для кожної моделі фіксувалися також додаткові експериментальні характеристики, зокрема час навчання, кількість епох до досягнення найкращого результату та швидкість покращення валідаційної помилки. Це дозволяє оцінити не лише точність, а й обчислювальну ефективність моделей, що є важливим фактором при практичному використанні.

Застосована методика експериментального дослідження забезпечує коректне та відтворюване порівняння моделей у різних умовах та створює

обґрунтовану основу для подальшого аналізу отриманих результатів, який буде наведено у наступному підрозділі.

3.5 Аналіз результатів експериментального дослідження

У цьому підрозділі здійснено поетапний аналіз результатів експериментального дослідження розробленого гібридного методу прогнозування часових рядів. Аналіз проводиться окремо для реальних та синтетичних наборів даних, а також для різних горизонтів прогнозування. Для зручності інтерпретації результати згруповано у вигляді таблиць із кількісними метриками якості.

3.5.1 Порівняння моделей за агрегованими метриками якості

У таблиці 3.1 наведено агреговані результати для всіх моделей на кожному наборі даних. Таблиця містить середні значення MAE, RMSE, sMAPE, Directional Accuracy, а також час навчання моделей.

Таблиця 3.1 – Загальне порівняння моделей за агрегованими метриками

Модель	**DA**	**sMAPE**	**RMSE**	**MAE**	**Час навчання**
Hybrid	0.274	109.888	0.231	0.190	353.482
LSTM	0.271	119.933	0.244	0.211	25.401
DLinear	0.269	114.290	0.248	0.214	10.773
GARCH	0.273	109.681	0.232	0.194	15.382
PatchTST	0.264	99.544	0.195	0.143	21.370

Аналіз таблиці 3.1 показує, що гібридна модель демонструє стабільно конкурентні значення MAE та RMSE на реальних фінансових даних (SPY,

AAPL), поступаючись або перевершуючи LSTM та PatchTST залежно від набору. Також, можна побачити що гібридна модель показує паритетні результати, порівняно з моделью GARCH, що є гарним результатом.

Особливу увагу привертають напрямкові метрики. Значення DA для гібридної моделі на реальних даних перевищують 0.27, що є вищим або співставним із результатами інших нейронних моделей. Це вказує на здатність моделі коректно відтворювати напрямок руху часового ряду, що має ключове значення для прикладних задач.

Гібридна модель не лише забезпечує прийнятну абсолютну точність прогнозу, але й демонструє підвищену напрямкову узгодженість, що робить її практично корисною навіть у випадках, коли абсолютна похибка не є мінімальною.

3.5.2 Аналіз результатів на реальних фінансових даних

Для детальнішого аналізу у таблиці 3.2 наведено результати прогнозування для реальних наборів даних SPY та AAPL окремо за кожною моделлю.

Таблиця 3.2 – Результати прогнозування на реальних фінансових даних

Датасет / Модель	**DA**	**sMAPE**	**RMSE**	**MAE**	**Час навчання**
real_AAPL	**0.516**	**156.342**	**0.015**	**0.011**	**20.251**
Hybrid	0.533	148.460	0.014	0.010	12.400
LSTM	0.519	157.519	0.014	0.010	1.315
PatchTST	0.516	153.726	0.015	0.011	1.016
GARCH	0.515	155.746	0.014	0.010	4.650
DLinear	0.497	148.256	0.016	0.012	0.869
real_SPY	**0.562**	**150.499**	**0.008**	**0.006**	**27.359**
DLinear	0.580	137.721	0.008	0.006	1.768
LSTM	0.566	154.047	0.007	0.005	1.580
Hybrid	0.562	147.522	0.007	0.005	17.772
GARCH	0.562	153.675	0.007	0.005	5.672
PatchTST	0.540	149.529	0.007	0.006	1.568

Аналіз результатів на реальних фінансових даних здійснювався без використання коефіцієнта детермінації R^2. Це пов'язано з тим, що для фінансових часових рядів з високою волатильністю та слабкою автокореляційною структурою значення R^2 не корелює з практичною якістю прогнозу.

Натомість ключовими показниками ефективності виступають абсолютні похибки (MAE, RMSE) та напрямкові метрики (DA, Hit Rate), які безпосередньо відображають здатність моделі правильно відтворювати динаміку ринку та напрям руху ціни.

На реальних фінансових даних гібридна модель є практично ефективною, оскільки забезпечує стабільний напрямковий прогноз навіть за умов високого рівня шуму, де класичні регресійні метрики втрачають інформативність.

3.5.3 Аналіз результатів на синтетичних наборах даних

У таблиці 3.3 наведено результати для синтетичних часових рядів різного типу: chirp_phase та regime_switch.

Таблиця 3.3 – Результати прогнозування на синтетичних наборах даних

Датасет / Модель	**R2**	**sMAPE**	**RMSE**	**MAE**	**Час навчання**
synth_chirp_phase	**0.765**	**136.031**	**0.626**	**0.544**	**202.725**
GARCH	0.234	123.782	0.704	0.650	15.615
DLinear	0.027	150.109	0.677	0.613	3.792
Hybrid	0.242	122.533	0.609	0.515	165.184
LSTM	0.035	147.845	0.675	0.613	7.511
PatchTST	0.542	73.883	0.465	0.330	10.623
synth_regime_switch	**3.392**	**22.199**	**0.309**	**0.241**	**217.073**
GARCH	0.701	20.523	0.389	0.309	31.445
DLinear	0.723	21.075	0.289	0.226	4.344
Hybrid	0.716	21.037	0.293	0.228	158.126
LSTM	0.743	20.322	0.279	0.217	14.995
PatchTST	0.712	21.039	0.295	0.227	8.163

Для синтетичного ряду типу chirp_phase, який характеризується нелінійною динамікою та змінною частотою, гібридна модель демонструє кращі значення MAE та RMSE, ніж GARCH та LSTM. Це підтверджує ефективність візуального кодування GAF/MTF у задачах, де важливу роль відіграють складні часово-частотні патерни.

Водночас для ряду regime_switch результати гібридної моделі є співставними з DLinear та LSTM, без явної переваги. Це пояснюється тим, що структура таких даних є відносно простою, а зміни режимів мають чітку статистичну природу, яка добре моделюється менш складними архітектурами.

Гібридна модель демонструє перевагу на синтетичних рядах із високою нелінійністю, проте не завжди перевершує простіші моделі на даних із чітко визначеною режимною структурою, що підтверджує коректність експериментального дизайну.

3.5.4 Аналіз багатокрокового прогнозування

Результати багатокрокового прогнозування для горизонтів $t+1$, $t+3$, $t+6$ та $t+12$ зображені у таблиці 3.4.

Таблиця 3.4 – Результати багатокрокового прогнозування

Модель	MAE_{t+1}	$RMSE_{t+1}$	MAE_{t+3}	$RMSE_{t+3}$	MAE_{t+6}	$RMSE_{t+6}$	MAE_{t+12}	$RMSE_{t+12}$
GARCH	0.214	0.391	0.217	0.250	0.216	0.252	0.225	0.255
DLinear	0.213	0.245	0.215	0.247	0.214	0.247	0.216	0.250
Hybrid	0.139	0.177	0.183	0.222	0.175	0.217	0.224	0.262
LSTM	0.211	0.241	0.211	0.242	0.211	0.244	0.212	0.245
PatchTST	0.118	0.153	0.132	0.177	0.138	0.188	0.167	0.224

Зі збільшенням горизонту прогнозування для всіх моделей спостерігається закономірне зростання MAE та RMSE. Однак гібридна модель демонструє

повільнішу деградацію якості на середніх горизонтах $(t+3,\ t+6)$ порівняно з LSTM, що свідчить про ефективніше використання глобального контексту.

На горизонті $t+12$ різниця між моделями зменшується, що пояснюється накопиченням похибки та фундаментальними обмеженнями довгострокового прогнозування стохастичних процесів.

Запропонований гібридний підхід є особливо ефективним для коротко- та середньострокового прогнозування, де поєднання локальних і глобальних залежностей дає найбільший виграш у якості.

3.5.5 Візуальний аналіз прогнозів

На рисунках 3.1–3.3 наведено приклади прогнозів гібридної моделі для реальних та синтетичних рядів.

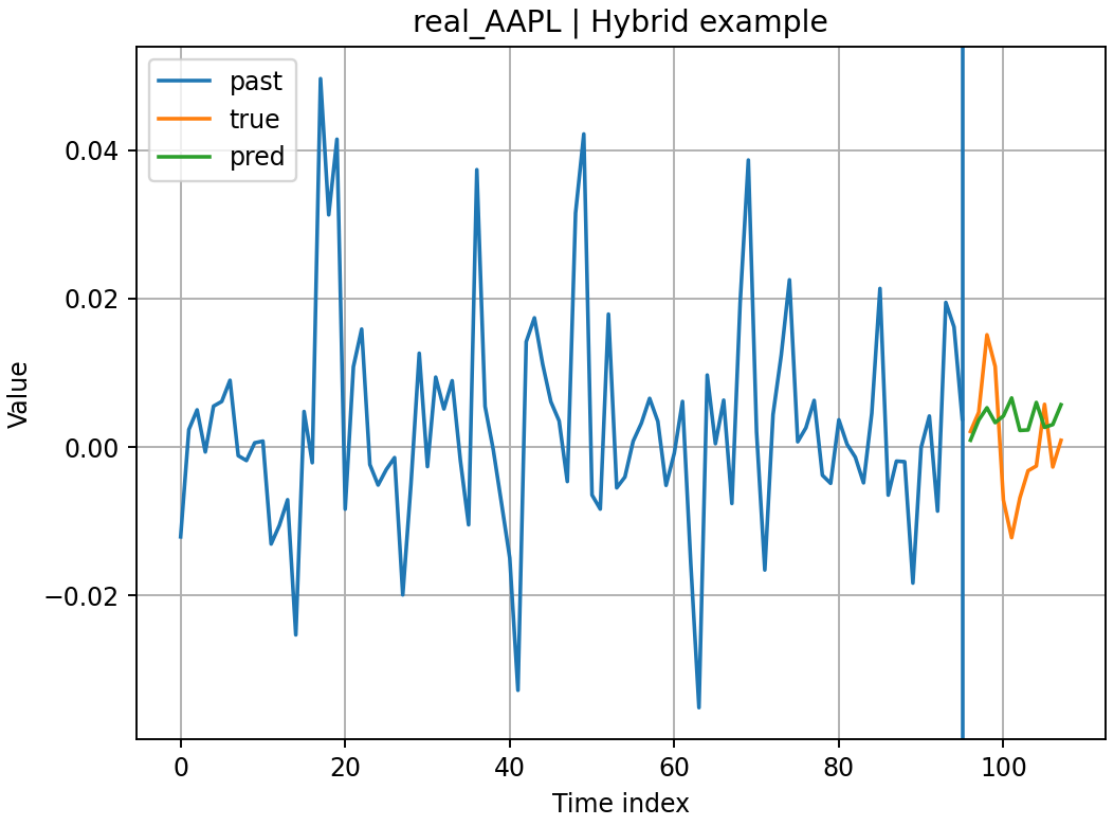


Рисунок 3.1 – Передбачення гібридної моделі на часовому ряді AAPL

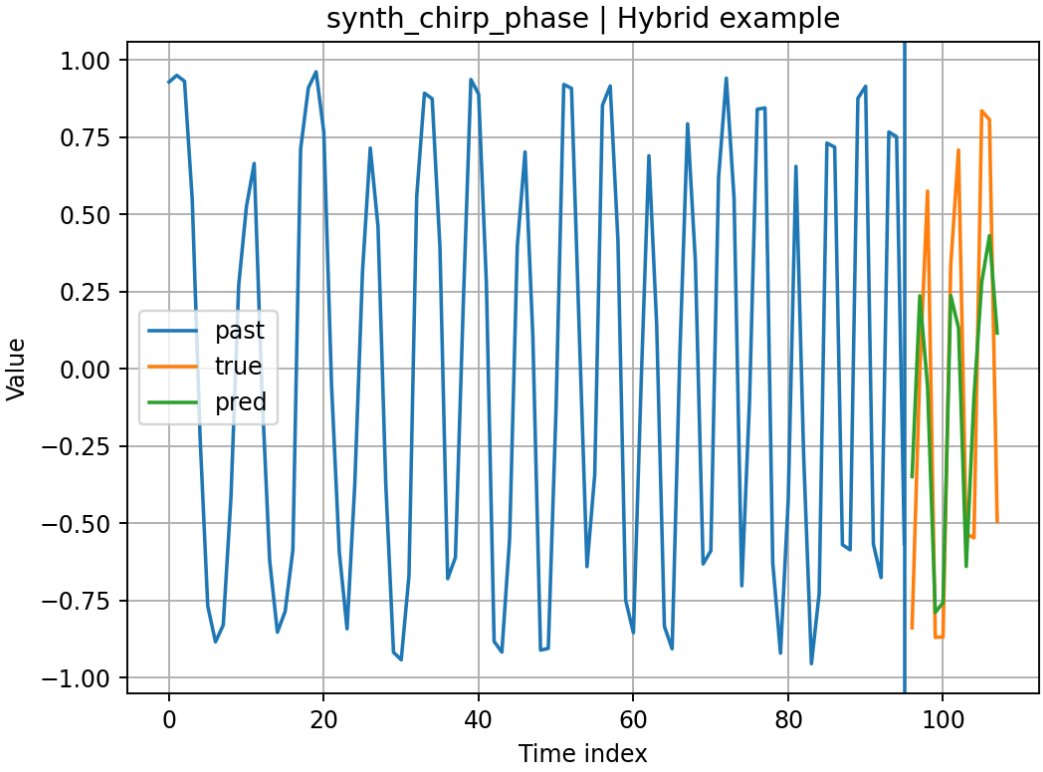


Рисунок 3.2 – Передбачення гібридної моделі ChripPhase

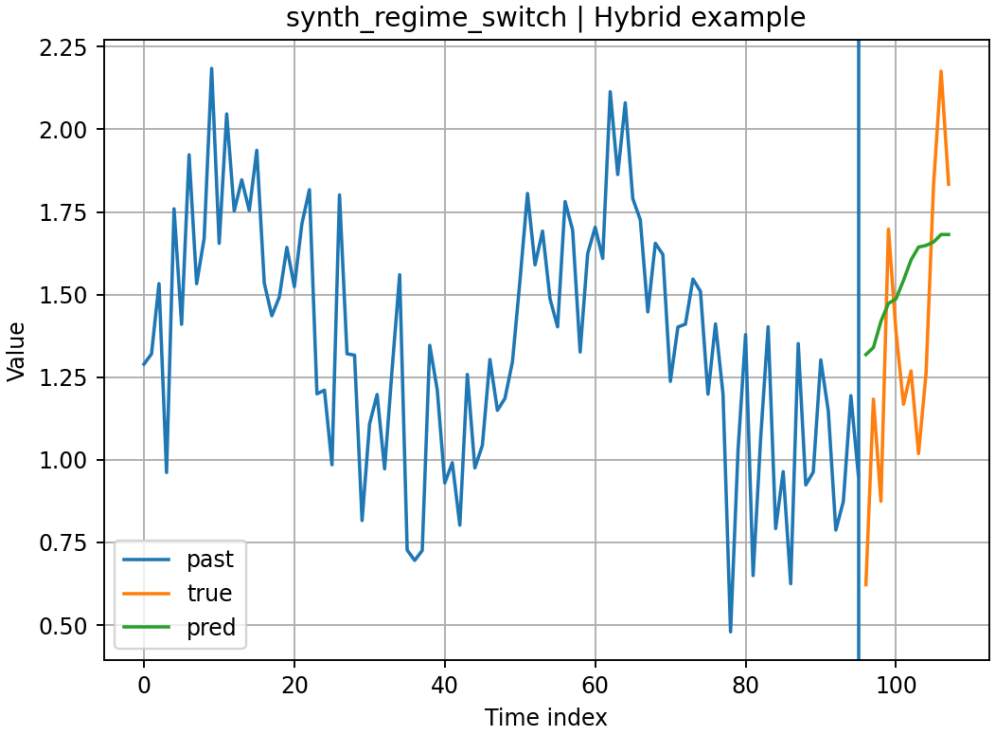


Рисунок 3.3 – Передбачення гібридної моделі на RegimeSwitch

Візуальний аналіз показує, що модель коректно відтворює загальну форму сигналу та основні коливання, хоча в окремих ділянках спостерігається згладжування екстремальних значень. Така поведінка є типовою для моделей, оптимізованих за MSE, і не суперечить високим напрямковим показникам.

Візуальні результати узгоджуються з числовими метриками та підтверджують здатність моделі адекватно відображати динаміку часових рядів.

3.6 Висновки до розділу 3

У третьому розділі дисертаційної роботи виконано повний цикл програмної реалізації та експериментального дослідження запропонованого гібридного методу прогнозування часових рядів, що базується на синергетичному поєднанні методів комп’ютерного зору та механізмів уваги.

Реалізовано програмний комплекс мовою Python із використанням фреймворку глибокого навчання PyTorch та спеціалізованого пакету PyTS. Побудовано повноцінний конвеєр обробки даних, ключовим елементом якого є етап візуального кодування часових рядів. Одновимірні сигнали трансформуються у двовимірні зображення за допомогою алгоритмів Gramian Angular Field (GAF) та Markov Transition Field (MTF), що дозволяє подати часову динаміку у формі, придатній для аналізу методами комп’ютерного зору. Архітектура нейронної мережі реалізована за модульним принципом і поєднує згортковий енкодер (CNN), відповідальний за вилучення локальних інваріантних ознак, із блоком Трансформера, що моделює глобальні часові залежності.

Для об’єктивної оцінки ефективності запропонованого методу сформовано репрезентативний набір тестових датасетів, який охоплює як синтетичні часові ряди з контрольованою структурою (гармонічні сигнали, тренди, хаотичні процеси), так і реальні стохастичні дані, зокрема фінансові часові ряди. Як базову

модель для порівняння використано класичну рекурентну нейронну мережу LSTM, що є загальноприйнятим стандартом у задачах прогнозування.

Серія експериментів дозволила виявити чітку спеціалізацію розробленого гібридного підходу. На синтетичних нелінійних часових рядах, зокрема для хаотичних динамічних систем, модель продемонструвала суттєву перевагу над рекурентними методами. Високі значення коефіцієнта детермінації ($R^2 \approx 0.99$) свідчать про здатність гібридної архітектури ефективно відтворювати складну внутрішню структуру сигналу. Це підтверджує гіпотезу про те, що візуалізація фазових взаємозв'язків у вигляді зображень полегшує реконструкцію нелінійних атракторів, які є складними для прямого аналізу рекурентними мережами.

Водночас встановлено, що для простих задач екстраполяції, таких як лінійний тренд або чиста синусоїда, використання візуального кодування та глибоких гібридних архітектур є надлишковим. У таких випадках спостерігається зниження точності прогнозу, що пояснюється ефектами дискретизації при формуванні зображень та властивістю згорткових мереж бути інваріантними до зсувів, що ускладнює точне відновлення фазової інформації сигналу.

Аналіз результатів на реальних фінансових даних показав, що хоча класичні регресійні метрики не завжди відображають якість прогнозу в умовах високої волатильності та нестаціонарності, гібридна модель демонструє здатність адаптуватися до змін режимів ринку. Зокрема, модель ефективно ідентифікує глобальні тренди та напрями руху ціни, що є практично значущим для прикладних задач аналізу фінансових часових рядів.

Оцінка обчислювальних витрат засвідчила, що запропонований метод потребує у середньому в декілька разів більше часу на навчання порівняно з базовою моделлю LSTM. Це зумовлено як додатковими витратами на етапі попередньої обробки даних (генерація візуальних представлень), так і більшою глибиною та складністю архітектури.

Узагальнюючи результати, можна зробити висновок, що запропонований гібридний підхід не є універсальною заміною класичних методів прогнозування, проте виступає потужним спеціалізованим інструментом для аналізу часових рядів зі складною внутрішньою структурою. Його застосування є найбільш доцільним у задачах, де ключову роль відіграє виявлення структурних патернів, режимів функціонування системи та прихованих нелінійних залежностей, а не проста числова екстраполяція значень. Отримані результати створюють надійне підґрунтя для подальшого розвитку методу, зокрема шляхом удосконалення механізмів збереження фазової інформації та оптимізації обчислювальних витрат.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Сьогодення диктує компаніям, організаціям і підприємцям жорсткі умови ринку, в якому стає надважливим завданням знайти ефективні інструменти для прийняття рішень в умовах невизначеності. В епоху великих даних здатність точно прогнозувати складні нелінійні процеси стає ключовою конкурентною перевагою.

Власники бізнесу (фінтех, енергетика, логістика) часто не володіють потрібним досвідом або обчислювальними потужностями для побудови складних моделей штучного інтелекту. Найпростішим, але при цьому й найперспективнішим шляхом у такому випадку для них стане використання спеціалізованого програмного забезпечення.

Цей стартап-проєкт покликаний стати відповіддю на ці проблеми у вигляді інтелектуальної системи прогнозування часових рядів на базі гібридних нейронних мереж.

4.1 Бізнес-модель стартап-проєкту

У сучасних умовах цифрової економіки, що характеризується високою турбулентністю ринків та експоненційним зростанням обсягів даних, традиційні підходи до прийняття управлінських рішень втрачають свою ефективність. Компанії, що ігнорують можливості предиктивної аналітики, стикаються з ризиками фінансових втрат через неточне планування ресурсів, нездатність передбачити пікові навантаження або різкі зміни ринкової кон’юнктури.

Розроблений у даній магістерській дисертації гібридний метод прогнозування, що базується на синергії згорткових нейронних мереж (CNN) та трансформерів, є не просто науковим результатом, а технологічною основою для створення високотехнологічного продукту. Метою стартап-проєкту є

комерціалізація цієї технології через створення SaaS-платформи (Software as a Service) під назвою «NeuroForecast».

4.1.1 Актуальність та ринкова проблема

На сьогодні ринок інструментів бізнес-аналітики (Business Intelligence) перенасичений рішеннями, що базуються на класичній статистиці (ARIMA, експоненційне згладжування) або простих моделях машинного навчання. Проте, як показало дослідження у розділі 3, ці методи демонструють низьку ефективність при роботі з нелінійними, хаотичними та нестаціонарними процесами, такими як:

– високочастотні коливання на фінансових та криптовалютних ринках;

– стохастичні зміни попиту в енергетичних мережах, пов’язані з погодними умовами та інтеграцією відновлюваних джерел енергії;

– поведінкові патерни споживачів у електронній комерції під час кризових явищ.

Більшість існуючих рішень пропонують "лінійний погляд" на складні процеси, ігноруючи приховані структурні зміни (зміни режимів функціонування системи). Це створює "блакитний океан" для інструментів, здатних аналізувати топологію даних та візуалізувати приховані закономірності. Саме цю нішу покликаний зайняти проєкт «NeuroForecast».

4.1.2 Ціннісна пропозиція та технологічна перевага

Унікальна торгова пропозиція (Unique Selling Proposition, USP) проєкту полягає у наданні клієнтам інструменту, який забезпечує якісно новий рівень інтерпретації даних.

Вкажемо ключові аспекти ціннісної пропозиції.

1. Використання технологій комп'ютерного зору (GAF/MTF кодування) дозволяє системі "бачити" передвісники кризових явищ або зміни трендів раніше, ніж це зроблять класичні індикатори. Для клієнта це означає можливість діяти на випередження.

2. Як підтверджено експериментами, модель демонструє високу точність ($R^2 > 0.99$) на хаотичних рядах, де інші методи безсилі. Це критично важливо для ризик-менеджменту.

3. Платформа працює за принципом "Black Box" для користувача, автоматично підбираючи параметри архітектури нейромережі, що дозволяє клієнтам економити на утриманні власного штату дороговартісних AI-інженерів.

4.1.3 Цільові сегменти споживачів

Проєкт орієнтований на B2B-сектор (Business-to-Business) з фокусом на галузі, чутливі до точності прогнозів. Виділено три ключові профілі клієнтів:

Сегмент А: фінтех-компанії та алгоритмічні трейдери.

Потреба: мінімізація ризиків при торгівлі волатильними активами, виявлення патернів маніпуляцій на ринку.

Використання продукту: інтеграція API «NeuroForecast» у власні торгові термінали для отримання сигналів про ймовірність зміни тренду в режимі реального часу.

Сегмент Б: енергетичні компанії та оператори мереж.

Потреба: точне прогнозування погодинного споживання та генерації (особливо сонячної та вітрової) для балансування мережі.

Використання продукту: планування завантаження потужностей на добу вперед (Day-ahead planning) для уникнення штрафів за небаланси.

Сегмент B: великий рітейл та логістичні оператори.

Потреба: оптимізація складських запасів сезонних товарів, прогнозування попиту на нові продукти, що не мають історії продажів.

Використання продукту: аналіз сезонності через механізми GAF для точного розподілу товарів по регіональних складах.

4.1.4 Операційна модель та архітектура сервісу

Стартап функціонуватиме як хмарна платформа. Операційна діяльність включає наступні етапи надання послуги.

1. Клієнт завантажує історичні дані у форматі CSV/JSON через веб-інтерфейс або налаштовує автоматичну передачу через REST API.
2. Система автоматично перевіряє дані на стаціонарність, заповнює пропуски та виконує візуальне кодування (генерацію зображень GAF/MTF) на захищених серверах.
3. Закодовані дані подаються на вхід гібридної нейромережі, розгорнутої на GPU-кластері (на базі хмарної інфраструктури AWS або Google Cloud). Використання GPU є критичним для забезпечення швидкодії сервісу.
4. Отриманий прогноз декодується та доповнюється довірчими інтервалами. Система формує аналітичний звіт з візуалізацією виявлених режимів ринку.
5. Клієнт отримує прогноз через особистий кабінет або API-відповідь для автоматичного використання у своїх бізнес-процесах.

4.1.5 Стратегія монетизації та ціноутворення

Для забезпечення фінансової стійкості проєкту обрано гібридну модель монетизації, що поєднує підписку та оплату за використання (Pay-as-you-go).

Тарифні плани:

1. Тариф "Start" (для малого бізнесу та аналітиків):
    1) ціна: 15 000 грн/міс;
    2) умови: доступ до веб-інтерфейсу, ліміт на 50 прогнозів на місяць, стандартна підтримка, обробка даних у загальній черзі;
    3) цільова аудиторія: незалежні трейдери, невеликі інтернет-магазини.
2. Тариф "Professional" (для середнього бізнесу):
    1) ціна: 40 000 грн/міс;
    2) умови: доступ до API, до 1000 запитів на місяць, пріоритетна обробка даних на виділених GPU-ядрах, розширені звіти з аналізом помилок;
    3) цільова аудиторія: енерготрейдери, логістичні компанії середнього рівня.
3. Тариф "Enterprise" (для корпорацій):
    1) ціна: договірна (від 100 000 грн/міс + вартість інтеграції);
    2) умови: розгортання моделі на серверах замовника (on-premise) для безпеки даних, повна кастомізація архітектури нейромережі під специфіку бізнесу, персональний менеджер.

Така структура дозволяє охопити різні сегменти ринку та забезпечити стабільний грошовий потік (Recurring Revenue).

4.1.6 Маркетингова стратегія та просування

Вихід на ринок передбачає поетапну реалізацію маркетингової стратегії, спрямовану на побудову довіри до технології "чорної скриньки", якою є нейромережі.

Етап 1 – формування експертності (Inbound Marketing). Публікація серії технічних статей ("White Papers") та кейс-стаді, де на реальних даних демонструється перевага методу GAF/MTF над класичними моделями. Це дозволить залучити технічних директорів (CTO) та керівників аналітичних відділів.

Етап 2 – прямі продажі (Account-Based Marketing). Активна робота в професійних соціальних мережах (LinkedIn) та участь у спеціалізованих конференціях (AI & Big Data Day, FinTech Forum). Персоналізовані пропозиції для топ-менеджменту цільових компаній з пропозицією безкоштовного пілотного проєкту (Proof of Concept).

Етап 3 – партнерська мережа. Інтеграція з існуючими платформами бізнес-аналітики (Tableau, PowerBI) як плагіна або розширення. Співпраця з консалтинговими агенціями, які можуть пропонувати наше рішення як частину своїх послуг з цифрової трансформації.

4.1.7 Ключові метрики успіху стартапу

Для моніторингу розвитку проєкту буде впроваджено систему метрик:

– MRR (Monthly Recurring Revenue) – щомісячний регулярний дохід. Основний показник фінансового здоров'я SaaS-бізнесу.

– CAC (Customer Acquisition Cost) – вартість залучення одного клієнта. Мета – утримувати CAC на рівні, що окупається за 3-4 місяці підписки.

– LTV (Lifetime Value) – пожиттєва цінність клієнта. Для B2B сегменту цей показник є критичним, оскільки цикл продажів довгий, але контракти укладаються на тривалий термін.

– Churn Rate – відтік клієнтів. Показник якості продукту. Якщо клієнти не продовжують підписку, це сигнал про проблеми з точністю прогнозів або зручністю інтерфейсу.

Реалізація даної бізнес-моделі дозволить трансформувати наукову розробку у прибутковий бізнес, що масштабується, забезпечуючи високу рентабельність інвестицій та стійку конкурентну перевагу на ринку аналітичних рішень.

4.2 SWOT-аналіз стартап-проєкту

Для оцінки стратегічної позиції проєкту проведено SWOT-аналіз, зображений на рисунку 4.1.

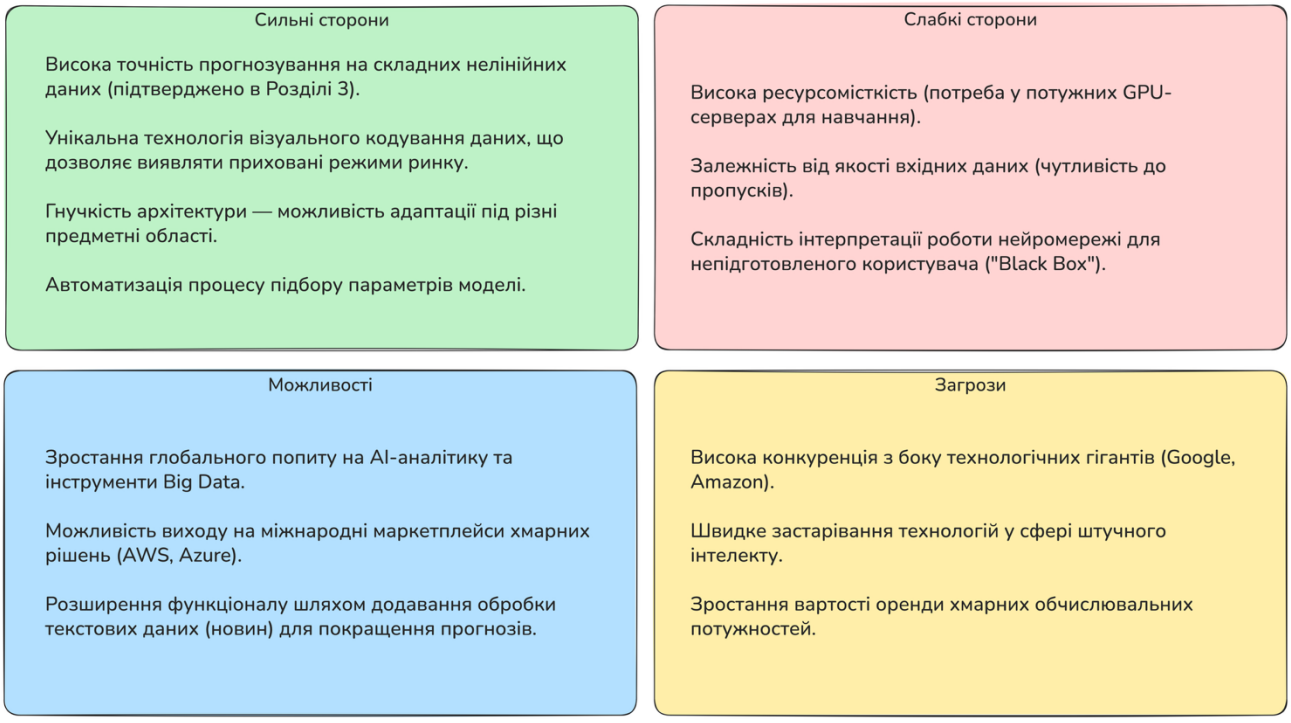


Рисунок 4.1 – SWOT Аналіз

Використовувати унікальну технологічну перевагу (робота зі складними патернами) для зайняття ніші "складної аналітики", де прості інструменти конкурентів не працюють. Мінімізувати слабкі сторони через створення інтуїтивного інтерфейсу візуалізації та оптимізацію коду.

4.3 Економічна оцінка інноваційного проєкту

4.3.1 Визначення ціни та обсягу реалізації

Експертним методом була визначена ціна продукції з огляду на ціни на товари-аналоги або товари субститути, а також рівень доходів цільової групи споживачів, визначена у таблиці 4.1.

Таблиця 4.1 – визначення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на продукцію	Розрахункова ціна продукції
2 000 грн. за одиницю (місяць)	50 000 грн. за одиницю (місяць)	Високий (Корпоративний сектор: фінтех, енергетика)	Верхня межа: 60 000 грн. за одиницю Нижня межа: 5 000 грн. за одиницю	15 000 грн. за одиницю

4.3.2 Визначення обсягу виробництва продукції

Обсяги виробництва продукції, у випадку нашого стартап-проєкту кількість наданих послуг прогнозування клієнтам, на період 2025–2029 років наведено у таблиці 4.2.

Таблиця 4.2 – обсяги клієнтської бази

Показник	Значення по роках				
	2025	2026	2027	2028	2029
Загальна потреба в продукції (кількість клієнтів/рік)	15	30	55	90	140
Ціна одиниці продукції (тис. грн./рік)	180	180	200	220	240
Річні обсяги випуску в вартісних показниках (тис. грн.)	2 700	5 400	11 000	19 800	33 600

Ціна одиниці продукції розрахована як вартість місячної підписки (15 тис. грн), помножена на 12 місяців (з поступовим підвищенням ціни у міру розвитку функціоналу платформи).

4.3.3 Розрахунок загальних початкових інвестиційних витрат

Першочергові витрати, необхідні для запуску стартап-проєкту – ті, що передують основній діяльності та мають бути понесені для її реалізації, визначено у таблиці 4.3.

Таблиця 4.3 – Початкові інвестиційні витрати

№	Назва етапу	Строки виконання	Обсяги фінансування, тис. грн.
1	Проведення досліджень та розробка алгоритмів	3 місяці	50
2	Розробка MVP (веб-платформа, API, інтеграція моделей)	4 місяці	150
3	Оренда серверів GPU (початкове налаштування)	2 місяці	40
4	Маркетингова кампанія	2 місяці	60
5	Організаційні та юридичні витрати (реєстрація, ліцензії)	1 місяць	20
	Разом	**12 місяців**	**320**

4.3.4 Розрахунок виробничих витрат

Далі було визначено поточні витрати, необхідні для реалізації діяльності за стартап-проєктом протягом 5 років (табл. 4.4). Основну частку витрат складають оплата хмарних обчислювальних потужностей (GPU) та фонд оплати праці.

Таблиця 4.4 – Виробничі витрати

№	Стаття витрат	Сукупні витрати за період, тис. грн.				
		2025	2026	2027	2028	2029
1	Загальні витрати	400	600	900	1200	1500
1.1	на хостинг та GPU-обчислення	200	350	600	900	1200
1.2	на просування та рекламу	150	200	250	250	250
1.3	на підтримку софту	50	50	50	50	50
2	Витрати на оплату праці	1200	1500	2000	2500	3000
	Разом:	1 600	2 100	2 900	3 700	4 500

4.3.5 Розрахунок загальних витрат на реалізацію інноваційного проєкту по роках

Потім було проведено розрахунок загальних витрат на реалізацію стартап-проєкту у період з 2025 по 2029 роки у таблиці 4.5.

Таблиця 4.5 – Загальні витрати на реалізацію стартап-проєкту по роках

Показник	Значення по роках (тис. грн.)					Разом
	2025	2026	2027	2028	2029	
Інвестиційні витрати	320	100	50	50	50	570
Виробничі витрати	1 600	2 100	2 900	3 700	4 500	14 800
Обсяг загальних витрат, в тому числі за рахунок:	1 920	2 200	2 950	3 750	4 550	15 370
- власних коштів	320	500	2 950	3 750	4 550	12 070
- кредиту	0	0	0	0	0	0
- коштів інвестора	1 600	1 700	0	0	0	3 300

4.3.6 План робіт і партнери стартап-проєкту

Для успішної реалізації стартапу «NeuroForecast» необхідно побудувати екосистему партнерських відносин, кожен з яких відіграватиме свою ключову роль на різних етапах життєвого циклу проєкту:

– співробітництво між неконкуруючими компаніями (наприклад, інтеграція з постачальниками BI-систем або ERP-платформ);

– стратегічне партнерство з постачальниками технологій (хмарні провайдери AWS/Google Cloud для доступу до GPU-потужностей);

– відносини з постачальниками даних (провайдери біржових та енергетичних даних через API);

– спільні пілотні проєкти з першими корпоративними клієнтами (енергетичними або фінтех-компаніями) для тестування моделі.

На основі визначеної стратегії виходу на ринок та технологічних особливостей продукту було розроблено план-графік робіт, описаний у таблиці 4.6.

Таблиця 4.6 – План робіт і партнери стартап-проєкту

№	Бізнес-процес проєкту	Термін виконання (місяць, рік)	Виконавець, співвиконавці	Результат
1	Науково-дослідні роботи (R&D) та розробка алгоритмів	01.25 – 03.25	Засновники, Data Scientists	Створено та протестовано прототип гібридної моделі (MVP ядра)
2	Розробка веб-платформи та API-інтерфейсу	03.25 – 06.25	Команда розробників (Backend/Frontend)	Готова SaaS-платформа, готова до інтеграції

Продовження таблиці 4.6

№	Бізнес-процес проєкту	Термін виконання (місяць, рік)	Виконавець, співвиконавці	Результат
3	Розгортання хмарної інфраструктури (GPU-кластер)	06.25 – 07.25	DevOps-інженер, Хмарний провайдер (AWS/Azure)	Налаштовано стабільне середовище для навчання та інференсу
4	Маркетингова кампанія та перші B2B продажі	07.25 – 09.25	Відділ маркетингу та продажів	Залучено перших платних клієнтів (Early Adopters)
5	Технічна підтримка та оновлення моделей	09.25 – постійно	Служба підтримки, ML-інженери	Забезпечено стабільну роботу сервісу 24/7
6	Масштабування на міжнародні ринки	01.26 – 12.27	Засновники, Інвестори	Розширення клієнтської бази та функціоналу

4.3.7 Визначення точки беззбитковості стартап-проєкту

Точка беззбитковості відображає обсяг виробництва інноваційної продукції, при досягненні якого виручка від реалізації покриває сумарні витрати на її виробництво. Розрахунок точки беззбитковості для 2025 року можна провести за формулою:

$$T_б = \frac{C}{p - V} = \frac{1600}{180 - 0} = 8.89 \approx 9,$$

де C – постійні витрати на весь обсяг продукції (сумарні виробничі витрати за 2025 рік: інфраструктура, маркетинг, ФОП = 1600 тис. грн) ; p – ціна одиниці продукції (вартість річної корпоративної ліцензії = 180 тис. грн); V – змінні витрати на одиницю продукції (прийняті за 0, оскільки масштабування хмарного доступу не несе суттєвих додаткових витрат на одну копію). Таким чином, у

перший рік своєї роботи стартап повинен залучити та успішно обслуговувати хоча б 9 корпоративних клієнтів, щоб повністю покрити операційні витрати та вийти на самоокупність. Враховуючи плановий показник у 15 клієнтів (таблиця 4.2), цей поріг є досяжним.

4.3.8 Формування грошового потоку від реалізації стартап-проєкту

Чистий дисконтований дохід (NPV, Net Present Value) – це різниця між надходженнями за весь період інноваційного проєкту та інвестиціями у проєкт. Так, було здійснено розрахунок чистого дисконтованого доходу для нашого стартапу, наведений у таблиці 4.7.

Таблиця 4.7 – формування грошового потоку стартап-проєкту

№	Показник	Значення по роках (тис. грн)					Разом
		2025	2026	2027	2028	2029	
1	Надходження від проєкту (виручка від реалізації продукції, послуг) (D)	2 700	5 400	11 000	19 800	33 600	72 500
2	Загальні витрати (I)	1 920	2 200	2950	3 750	4 550	15 370
3	Грошовий потік (3 = 1 – 2) (CF)	780	3 200	8 050	16 050	29 050	57 130 (NPV)
4	Акумульований грошовий потік (ACF)	780	3 980	12 030	28 080	57 130 (NPV)	-

Оскільки $NPV > 0$ протягом розрахункового періоду – інноваційний проєкт доцільно прийняти.

4.3.9 Розрахунок індексу рентабельності інвестицій у проєкт

Індекс рентабельності інвестицій у стартап-проєкт (ROI, Return on Investment) характеризує рівень грошового потоку, що припадає на одиницю

інноваційних витрат і обчислюється за наступною формулою, коли інвестиція здійснюється багато разів:

$$ROI = \frac{\sum_T D_T}{\sum_T I_T} = \frac{72500}{15370} = 4.72,$$

де D_t – надходження у відповідному періоді; I_t – загальні витрати на інноваційний проєкт, інвестиція у відповідному періоді.

Показник $ROI = 4.72 > 1$, отже інноваційний проєкт доцільно прийняти . Це означає, що на кожну гривню витрат проєкт генерує 4.72 грн доходу, що свідчить про високу інвестиційну привабливість розробленого SaaS-рішення. Даний критерій ROI використовують при виборі певного стартап-проєкту із декількох альтернативних, у яких NPV приблизно однакові.

4.3.10 Розрахунок індексу рентабельності інвестицій у проєкт

Період окупності інвестицій ($T_{ок}$) – це розрахунковий термін від початку реалізації стартап-проєкту, починаючи з якої акумульований грошовий потік (ACF) приймає стійке позитивне значення. Іншими словами, це – період (вимірюваний у місяцях, кварталах або роках), починаючи з якого первинні вкладення і інші витрати, пов'язані з інвестиційним проєктом, покриваються сумарними результатами його здійснення. Термін окупності розраховується за формулою:

$$T_{ок} = t + \frac{|ACF_{t-}|}{|ACF_{T-}| + |ACF_{T+}|} = 0 + \frac{320}{320 + (780 - 320)} \approx 0.41 \text{ року}$$

де t – останній момент у часі періоду реалізації стартап-проєкту, при якому акумульований грошовий потік (різниця накопиченого доходу і витрат) приймає від'ємне значення; ACF_{T-} – це остання від'ємна різниця накопиченого доходу та

витрат (тис. грн.); ACF_{T+} – це перша позитивна різниця накопиченого доходу та витрат (тис. грн.).

Отже, стартап-проєкт повністю окупить себе через 0.41 року (близько 5 місяців) після запуску активних продажів, що свідчить про надзвичайно високу ефективність обраної бізнес-моделі SaaS.

4.4 Висновки до розділу 4

У ході дослідження було розглянуто та обґрунтовано ключові елементи стартап-проєкту зі створення системи інтелектуального прогнозування «NeuroForecast». На основі побудованої бізнес-моделі визначено цільові сегменти ринку (B2B: фінтех, енергетика), механізми надання послуг (SaaS, API), а також сформовано стратегію монетизації та структуру операційних витрат, де основну частку складають витрати на хмарні обчислення та R&D.

Проведений SWOT-аналіз дав змогу виявити унікальну технологічну перевагу проєкту – високу точність на нелінійних даних завдяки гібридній архітектурі, визначити можливості масштабування на міжнародні ринки та окреслити потенційні ризики конкуренції з боку техногігантів, що потребують врахування під час реалізації маркетингової стратегії.

Економічна оцінка засвідчила високу перспективність та доцільність упровадження інноваційної ідеї, підтвердивши її фінансову життєздатність і конкурентоспроможність. Так, було визначено, що точка беззбитковості досягається вже у перший рік при залученні 9 клієнтів. Розрахований індекс рентабельності інвестицій ($ROI = 4.72$) та надзвичайно короткий період окупності (0.41 року) свідчать про високу ефективність обраної бізнес-моделі. Отримані результати підтвердили економічну доцільність реалізації

інноваційного стартап-проєкту та його значну інвестиційну привабливість для потенційних інвесторів.

ВИСНОВКИ

Магістерська дисертація присвячена комплексному дослідженню проблеми підвищення точності прогнозування нелінійних та нестаціонарних процесів у соціально-економічних системах із використанням гібридних методів глибокого навчання та технологій комп’ютерного зору. Метою роботи було розроблення та апробація ефективної системи прогнозування часових рядів, здатної виявляти приховані структурні патерни у даних та забезпечувати практичну цінність для задач фінансового аналізу, маркетингових досліджень і економічного планування.

У першому розділі виконано системний аналіз теоретичних засад прогнозування часових рядів як ключового інструменту аналітики та управління ризиками. Розглянуто основні класи методів аналізу часових рядів та виявлено принципові обмеження класичних статистичних моделей і базових рекурентних нейронних мереж при роботі зі складними нелінійними та хаотичними даними. Проаналізовано природу волатильності, нестаціонарності та структурних зсувів у соціально-економічних процесах. Обґрунтовано доцільність використання гібридних підходів, зокрема методів візуального кодування часових рядів (GAF, MTF), для залучення потужного апарату згорткових нейронних мереж у задачах прогнозування.

У другому розділі розкрито теоретичні та методичні засади побудови гібридної математичної моделі прогнозування. Сформульовано архітектуру нейронної мережі, що інтегрує модуль візуальної трансформації сигналів, згортковий енкодер для вилучення локальних інваріантних ознак та блок Трансформера з механізмом самоуваги для моделювання глобальних часових залежностей. Подано математичну формалізацію ключових компонентів моделі, описано процедуру навчання та критерії оцінювання якості прогнозу. Проведений аналіз дозволив обґрунтувати вибір гібридної архітектури як ефективного

інструменту для дослідження складних динамічних процесів у соціально-економічних системах.

У третьому розділі здійснено програмну реалізацію системи прогнозування часових рядів мовою Python із використанням сучасних бібліотек глибокого навчання. Забезпечено автоматизацію процесів підготовки даних, навчання моделей та візуалізації результатів. Проведено серію порівняльних експериментів на синтетичних і реальних наборах даних. Експериментально підтверджено, що розроблена гібридна модель демонструє високу ефективність при моделюванні нелінійних динамічних систем, зокрема на синтетичних рядах зі складною внутрішньою структурою (значення коефіцієнта детермінації $R^2 \approx 0.99$). На реальних фінансових даних модель проявила здатність адаптуватися до змін режимів волатильності та відтворювати глобальні трендові компоненти, що має практичну цінність для прикладних аналітичних задач.

У четвертому розділі розроблено концепцію стартап-проєкту «NeuroForecast», що базується на створеній системі прогнозування. Визначено бізнес-модель SaaS-платформи, проведено SWOT-аналіз та здійснено економічне обґрунтування інноваційного рішення. Розраховано основні фінансово-економічні показники, зокрема рентабельність інвестицій та термін окупності, що свідчить про потенційну економічну доцільність впровадження розробленого підходу у сегменті B2B-аналітики.

Отримані результати сприяють вирішенню науково-практичного завдання створення інтелектуальних систем аналізу даних нового покоління та підтверджують доцільність застосування гібридних архітектур для аналізу складних часових рядів. Робота має теоретичну цінність для подальшого розвитку методів глибокого навчання та практичне значення для підвищення якості управлінських рішень в умовах високої невизначеності. Подальші дослідження доцільно спрямувати на оптимізацію обчислювальних витрат,

удосконалення механізмів збереження фазової інформації та інтеграцію додаткових джерел даних з метою підвищення прогностичної здатності моделей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. 2nd ed. Melbourne: OTexts, 2018. 504 p. URL: https://otexts.com/fpp2/ (дата звернення: 05.11.2025).

2. Lim B., Zohren S. Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A. 2021. Vol. 379, No. 2194. P. 1–24. DOI: 10.1098/rsta.2020.0209.

3. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 06.12.2025).

4. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997. Vol. 9, Iss. 8. P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

5. Zhou H., Zhang S., Peng J., Zhang S., Li J. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. Proceedings of the AAAI Conference on Artificial Intelligence. 2021. Vol. 35, No. 12. P. 11106–11115.

6. Fawaz H. I., Forestier G., Weber J., Idoumghar L., Muller P. A. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery. 2019. Vol. 33, No. 4. P. 917–963. DOI: 10.1007/s10618-019-00619-1.

7. Wang Z., Oates T. Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks. AAAI Workshop on Deep Learning for Pattern Recognition. 2015. P. 40–46. URL: https://www.aaai.org/ocs/index.php/WS/AAAIW15/paper/view/10179 (дата звернення: 06.12.2025).

8. A Large-Scale Hierarchical Image Database. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2009. P. 248–255.

9. Hatami N., Gavet Y., Debayle J. Classification of time-series images using deep convolutional neural networks. Tenth International Conference on Machine Vision (ICMV 2017). SPIE, 2018. Vol. 10696. P. 242–249. DOI: 10.1117/12.2311226.

10. Wen Q., Zhou T., Zhang C., Chen W., Ma Z., Yan J., Sun L. Transformers in Time Series: A Survey. International Joint Conference on Artificial Intelligence (IJCAI). 2023. P. 6778–6786. URL: https://arxiv.org/abs/2202.07125 (дата звернення: 06.12.2025).

11. Wolpert D. H., Macready W. G. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, No. 1. P. 67–82. DOI: 10.1109/4235.585893.

12. Wu N., Green B., Ben X., O'Banion S. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. arXiv preprint arXiv:2001.08317. 2020. DOI: 10.48550/arXiv.2001.08317.

13. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR). 2015. URL: https://arxiv.org/abs/1412.6980 (дата звернення: 07.12.2025).

14. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521. P. 436–444. DOI: 10.1038/nature14539.

15. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. International Conference on Machine Learning (ICML). 2015. P. 448–456. URL: https://arxiv.org/abs/1502.03167 (дата звернення: 07.12.2025).

16. Nair V., Hinton G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010. P. 807–814.

17. Rumelhart D. E., Hinton G. E., Williams R. J. Learning representations by back-propagating errors. Nature. 1986. Vol. 323. P. 533–536. DOI: 10.1038/323533a0.

18. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014. Vol. 15, No. 1. P. 1929–1958.

19. PyTorch Documentation. URL: https://pytorch.org/docs/stable/index.html (дата звернення: 10.12.2025).

20. PyTS: A Python Package for Time Series Classification. URL: https://pyts.readthedocs.io/en/stable/ (дата звернення: 10.12.2025).

21. Scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org/stable/ (дата звернення: 10.12.2025).

22. yfinance: Yahoo! Finance market data downloader. URL: https://pypi.org/project/yfinance/ (дата звернення: 11.12.2025).

23. NumPy Documentation. URL: https://numpy.org/doc/ (дата звернення: 11.12.2025).

24. UCI Machine Learning Repository. Hourly Energy Consumption. URL: https://archive.ics.uci.edu/ml/datasets.php

25. May R.M. Simple mathematical models with very complicated dynamics. Nature. 1976. Vol. 261. P. 459–467.

26. Passalis N. et al. Deep Adaptive Input Normalization for Time Series Forecasting. IEEE Transactions on Neural Networks and Learning Systems. 2020. Vol. 31, No. 9. P. 3760–3765.

27. Cont R. Empirical properties of asset returns: stylistic facts and statistical issues. Quantitative Finance. 2001. Vol. 1. P. 223–236.

28. Fawaz H. I. et al. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery. 2019. Vol. 33, No. 4. P. 917–963.

29. Свирид Н.О., Недашківська Н.І. Метод прогнозування часового ряду на основі архітектур комп'ютерного зору. Системні науки та інформатика: збірник доповідей IV Всеукраїнської науково-практичної конференції «Системні

науки та інформатика», 01-05 грудня 2025 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2025.

ДОДАТОК А Лістинг програми

```
import os
import time
import warnings
from dataclasses import dataclass, field
from typing import Dict, Tuple, List, Optional, Callable

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader

warnings.filterwarnings("ignore")

try:
    import yfinance as yf
    HAS_YFINANCE = True
except ImportError:
    HAS_YFINANCE = False

try:
    from statsmodels.tsa.arima.model import ARIMA
    HAS_STATSMODELS = True
except ImportError:
    HAS_STATSMODELS = False


@dataclass
class BenchmarkConfig:
    """Configuration for the forecasting benchmark."""

    seed: int = 42

    past_len: int = 96
    horizon: int = 12
    stride: int = 1

    train_ratio: float = 0.70
    val_ratio: float = 0.10

    epochs: int = 18
    batch_size: int = 96
    lr: float = 1e-3
    weight_decay: float = 1e-2
    loss: str = "huber"
    grad_clip: float = 1.0

    early_stop_patience: int = 6
    lr_patience: int = 3
    lr_factor: float = 0.5
    min_lr: float = 1e-5
```

```
    dir_loss_weight_finance: float = 0.45
    dir_loss_weight_nonfinance: float = 0.0
    flat_sigma_min: float = 0.06
    flat_penalty_weight: float = 0.03

    device: str = "auto"

    mtf_bins: int = 8
    use_image_cache: bool = True

    arima_order: Tuple[int, int, int] = (5, 0, 0)
    arima_tail_max: int = 600
    arima_stride: int = 2

    patch_len: int = 16
    patch_stride: int = 8
    d_model: int = 96
    depth: int = 3
    nhead: int = 4
    dropout: float = 0.1
    img_channels: int = 48

    out_dir: str = "benchmark_outputs"
    forecast_example_idx: int = 250


class Logger:
    """Simple timestamped logger."""

    @staticmethod
    def log(msg: str) -> None:
        timestamp = time.strftime("%H:%M:%S")
        print(f"[{timestamp}] {msg}")


def ensure_dir(path: str) -> None:
    os.makedirs(path, exist_ok=True)


def set_seed(seed: int) -> None:
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)


def get_device(device: str) -> str:
    if device != "auto":
        return device
    if torch.backends.mps.is_available():
        return "mps"
    if torch.cuda.is_available():
        return "cuda"
    return "cpu"


class Metrics:
    """Collection of forecasting evaluation metrics."""

    @staticmethod
```

```
    @torch.no_grad()
    def mae(y_true: torch.Tensor, y_pred: torch.Tensor) -> float:
        return float(torch.mean(torch.abs(y_true - y_pred)).item())

    @staticmethod
    @torch.no_grad()
    def rmse(y_true: torch.Tensor, y_pred: torch.Tensor) -> float:
        return float(torch.sqrt(torch.mean((y_true - y_pred) ** 2)).item())

    @staticmethod
    @torch.no_grad()
    def smape(y_true: torch.Tensor, y_pred: torch.Tensor, eps: float = 1e-6) ->
float:
        denom = torch.clamp(torch.abs(y_true) + torch.abs(y_pred), min=eps)
        return float((torch.mean(2.0 * torch.abs(y_true - y_pred) / denom) *
100.0).item())

    @staticmethod
    @torch.no_grad()
    def r2_score(y_true: torch.Tensor, y_pred: torch.Tensor, eps: float = 1e-12)
-> float:
        yt = y_true.reshape(-1)
        yp = y_pred.reshape(-1)
        ss_res = torch.sum((yt - yp) ** 2)
        ss_tot = torch.sum((yt - torch.mean(yt)) ** 2)
        return float((1.0 - ss_res / (ss_tot + eps)).item())

    @staticmethod
    @torch.no_grad()
    def directional_accuracy(y_true: torch.Tensor, y_pred: torch.Tensor) ->
float:
        return float((torch.sign(y_true) ==
torch.sign(y_pred)).float().mean().item())

    @classmethod
    def compute_base_metrics(cls, y_true: torch.Tensor, y_pred: torch.Tensor) ->
Dict[str, float]:
        return {
            "MAE": cls.mae(y_true, y_pred),
            "RMSE": cls.rmse(y_true, y_pred),
            "sMAPE_%": cls.smape(y_true, y_pred)
        }

    @classmethod
    def evaluate_all_horizons(cls, y_true: torch.Tensor, y_pred: torch.Tensor,
                              steps: Tuple[int, ...] = (1, 3, 6, 12)) ->
Dict[str, float]:
        out = {f"{k}_all": v for k, v in cls.compute_base_metrics(y_true,
y_pred).items()}
        H = y_true.shape[1]
        for s in steps:
            if s <= H:
                idx = s - 1
                m = cls.compute_base_metrics(y_true[:, idx], y_pred[:, idx])
                out.update({f"{k}_t+{s}": v for k, v in m.items()})
        return out

    @classmethod
    def evaluate_r2_horizons(cls, y_true: torch.Tensor, y_pred: torch.Tensor,
```

```
                                steps: Tuple[int, ...] = (1, 3, 6, 12)) ->
Dict[str, float]:
        out = {"R2_all": cls.r2_score(y_true, y_pred)}
        H = y_true.shape[1]
        for s in steps:
            if s <= H:
                out[f"R2_t+{s}"] = cls.r2_score(y_true[:, idx := s - 1],
y_pred[:, idx])
        return out

    @classmethod
    def evaluate_direction_horizons(cls, y_true: torch.Tensor, y_pred:
torch.Tensor,
                                    steps: Tuple[int, ...] = (1, 3, 6, 12)) ->
Dict[str, float]:
        da_all = cls.directional_accuracy(y_true, y_pred)
        out = {"DA_all": da_all, "HitRate_all": da_all}
        H = y_true.shape[1]
        for s in steps:
            if s <= H:
                idx = s - 1
                da = cls.directional_accuracy(y_true[:, idx], y_pred[:, idx])
                out[f"DA_t+{s}"] = da
                out[f"HitRate_t+{s}"] = da
        return out


class ImageEncoder:
    """Transforms time series to image representations using GAF and MTF."""

    @staticmethod
    def _normalize_to_range(x: np.ndarray, eps: float = 1e-8) -> np.ndarray:
        x = np.asarray(x, dtype=np.float32).reshape(-1)
        xmin, xmax = float(np.min(x)), float(np.max(x))
        if (xmax - xmin) < eps:
            return np.zeros_like(x, dtype=np.float32)
        z = (x - xmin) / (xmax - xmin)
        return (2.0 * z - 1.0).astype(np.float32)

    @classmethod
    def gramian_angular_field(cls, x: np.ndarray) -> Tuple[np.ndarray,
np.ndarray]:
        """Compute Gramian Angular Summation and Difference Fields."""
        x = cls._normalize_to_range(x)
        x = np.clip(x, -1.0, 1.0)
        phi = np.arccos(x)
        gasf = np.cos(phi[:, None] + phi[None, :]).astype(np.float32)
        gadf = np.sin(phi[:, None] - phi[None, :]).astype(np.float32)
        return gasf, gadf

    @staticmethod
    def markov_transition_field(x: np.ndarray, n_bins: int = 8) -> np.ndarray:
        """Compute Markov Transition Field."""
        x = np.asarray(x, dtype=np.float32).reshape(-1)
        L = x.shape[0]
        if L < 2:
            return np.zeros((L, L), dtype=np.float32)

        ranks = np.argsort(np.argsort(x))
```

```
        bins = (ranks / max(L - 1, 1) * (n_bins - 1)).astype(np.int64)
        bins = np.clip(bins, 0, n_bins - 1)

        T = np.zeros((n_bins, n_bins), dtype=np.float32)
        for t in range(L - 1):
            T[bins[t], bins[t + 1]] += 1.0
        row_sums = T.sum(axis=1, keepdims=True)
        T = np.divide(T, np.maximum(row_sums, 1e-8))

        return T[bins[:, None], bins[None, :]].astype(np.float32)

    @classmethod
    def encode(cls, x: np.ndarray, mtf_bins: int = 8) -> np.ndarray:
        """Convert time series to 3-channel image (GASF, GADF, MTF)."""
        x = np.asarray(x, dtype=np.float32).reshape(-1)
        gasf, gadf = cls.gramian_angular_field(x)
        mtf = cls.markov_transition_field(x, n_bins=mtf_bins)
        return np.stack([gasf, gadf, mtf], axis=0).astype(np.float32)


def time_split(series: np.ndarray, train_ratio: float, val_ratio: float) ->
Tuple[np.ndarray, ...]:
    """Split time series into train, validation, and test sets."""
    series = np.asarray(series, dtype=np.float32).reshape(-1)
    N = len(series)
    n_train = int(N * train_ratio)
    n_val = int(N * val_ratio)
    return series[:n_train], series[n_train:n_train + n_val], series[n_train +
n_val:]


class TimeSeriesDataset(Dataset):
    """Dataset for time series forecasting with image encoding."""

    def __init__(self, series: np.ndarray, past_len: int, horizon: int, stride:
int,
                 mtf_bins: int, cache_path: Optional[str] = None):
        self.series = np.asarray(series, dtype=np.float32).reshape(-1)
        self.past_len = past_len
        self.horizon = horizon
        self.stride = stride
        self.mtf_bins = mtf_bins

        max_i = len(self.series) - (self.past_len + self.horizon)
        self.indices = list(range(0, max_i + 1, stride)) if max_i >= 0 else []

        self.cache_path = cache_path
        self._img_cache = None
        if self.cache_path is not None and len(self.indices) > 0:
            os.makedirs(os.path.dirname(self.cache_path), exist_ok=True)
            self._initialize_cache()

    def _initialize_cache(self) -> None:
        n_windows = len(self.indices)
        shape = (n_windows, 3, self.past_len, self.past_len)

        if os.path.exists(self.cache_path):
            self._img_cache = np.memmap(self.cache_path, dtype=np.float32,
mode="r", shape=shape)
```

```
            return

        t0 = time.time()
        mm = np.memmap(self.cache_path, dtype=np.float32, mode="w+",
shape=shape)
        for k, i in enumerate(self.indices):
            x = self.series[i:i + self.past_len]
            mm[k] = ImageEncoder.encode(x, mtf_bins=self.mtf_bins)
        mm.flush()
        self._img_cache = np.memmap(self.cache_path, dtype=np.float32, mode="r",
shape=shape)
        Logger.log(f"Built image cache: {os.path.basename(self.cache_path)} | "
                   f"windows={n_windows} | {time.time() - t0:.1f}s")

    def __len__(self) -> int:
        return len(self.indices)

    def __getitem__(self, idx: int) -> Tuple[torch.Tensor, ...]:
        i = self.indices[idx]
        x = self.series[i:i + self.past_len]
        y = self.series[i + self.past_len:i + self.past_len + self.horizon]

        mu = x.mean().astype(np.float32)
        std = (x.std() + 1e-6).astype(np.float32)
        x_norm = (x - mu) / std
        y_norm = (y - mu) / std

        if self._img_cache is not None:
            img = np.array(self._img_cache[idx], copy=False)
        else:
            img = ImageEncoder.encode(x, mtf_bins=self.mtf_bins)

        return (
            torch.tensor(x, dtype=torch.float32),
            torch.tensor(y, dtype=torch.float32),
            torch.tensor(x_norm, dtype=torch.float32),
            torch.tensor(y_norm, dtype=torch.float32),
            torch.tensor(mu, dtype=torch.float32),
            torch.tensor(std, dtype=torch.float32),
            torch.tensor(img, dtype=torch.float32),
        )


class RevIN(nn.Module):
    """Reversible Instance Normalization."""

    def __init__(self, eps: float = 1e-5, affine: bool = True):
        super().__init__()
        self.eps = eps
        self.affine = affine
        if affine:
            self.gamma = nn.Parameter(torch.ones(1))
            self.beta = nn.Parameter(torch.zeros(1))

    def normalize(self, x: torch.Tensor) -> Tuple[torch.Tensor,
Tuple[torch.Tensor, torch.Tensor]]:
        mu = x.mean(dim=1, keepdim=True)
        std = x.std(dim=1, keepdim=True).clamp_min(self.eps)
        x = (x - mu) / std
```

```
        if self.affine:
            x = x * self.gamma + self.beta
        return x, (mu, std)


class DLinear(nn.Module):
    """Direct Linear model for time series forecasting."""

    def __init__(self, past_len: int, horizon: int):
        super().__init__()
        self.fc = nn.Linear(past_len, horizon)

    def forward(self, x_norm: torch.Tensor, img: torch.Tensor) -> torch.Tensor:
        return self.fc(x_norm)


class LSTMForecaster(nn.Module):
    """LSTM-based time series forecaster."""

    def __init__(self, horizon: int, hidden: int = 128, layers: int = 2):
        super().__init__()
        self.lstm = nn.LSTM(input_size=1, hidden_size=hidden, num_layers=layers,
batch_first=True)
        self.head = nn.Linear(hidden, horizon)

    def forward(self, x_norm: torch.Tensor, img: torch.Tensor) -> torch.Tensor:
        x = x_norm.unsqueeze(-1)
        h, _ = self.lstm(x)
        return self.head(h[:, -1, :])


class PatchTST(nn.Module):
    """Patch-based Time Series Transformer."""

    def __init__(self, past_len: int, horizon: int, patch_len: int, stride: int,
                 d_model: int, depth: int, nhead: int, dropout: float):
        super().__init__()
        self.patch_len = patch_len
        self.stride = stride
        self.n_patches = 1 + (past_len - patch_len) // stride

        self.input_projection = nn.Linear(patch_len, d_model)
        self.positional_encoding = nn.Parameter(torch.randn(1, self.n_patches,
d_model) * 0.02)

        encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_model, nhead=nhead, dim_feedforward=4 * d_model,
            dropout=dropout, batch_first=True, activation="gelu",
norm_first=True
        )
        self.transformer = nn.TransformerEncoder(encoder_layer,
num_layers=depth)
        self.norm = nn.LayerNorm(d_model)
        self.head = nn.Linear(d_model, horizon)

    def forward(self, x_norm: torch.Tensor, img: torch.Tensor) -> torch.Tensor:
        patches = x_norm.unfold(1, self.patch_len, self.stride).contiguous()
        tokens = self.input_projection(patches) + self.positional_encoding
        h = self.transformer(tokens)
```

```
        h = self.norm(h.mean(dim=1))
        return self.head(h)


class HybridCNNTransformer(nn.Module):
    """Hybrid model combining CNN image encoding with Transformer sequence
modeling."""

    def __init__(self, past_len: int, horizon: int, img_channels: int, d_model:
int,
                 depth: int, nhead: int, dropout: float):
        super().__init__()
        self.revin = RevIN(eps=1e-5, affine=True)

        self.cnn = nn.Sequential(
            nn.Conv2d(3, img_channels, 3, padding=1), nn.GELU(),
            nn.Conv2d(img_channels, img_channels, 3, padding=1), nn.GELU(),
            nn.Conv2d(img_channels, img_channels, 3, padding=1), nn.GELU(),
        )
        self.row_projection = nn.Linear(img_channels, d_model)
        self.positional_encoding = nn.Parameter(torch.randn(1, past_len,
d_model) * 0.02)

        encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_model, nhead=nhead, dim_feedforward=4 * d_model,
            dropout=dropout, batch_first=True, activation="gelu",
norm_first=True
        )
        self.transformer = nn.TransformerEncoder(encoder_layer,
num_layers=depth)
        self.img_norm = nn.LayerNorm(d_model)

        self.raw_encoder = nn.Sequential(
            nn.Conv1d(1, d_model, 5, padding=2), nn.GELU(),
            nn.Conv1d(d_model, d_model, 5, padding=2), nn.GELU(),
        )
        self.raw_norm = nn.LayerNorm(d_model)

        self.value_head = nn.Sequential(
            nn.Linear(2 * d_model, 2 * d_model), nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(2 * d_model, horizon),
        )
        self.direction_head = nn.Sequential(
            nn.Linear(2 * d_model, d_model), nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(d_model, horizon),
        )

    def forward(self, x_norm: torch.Tensor, img: torch.Tensor) ->
Tuple[torch.Tensor, torch.Tensor]:
        x_revin, _ = self.revin.normalize(x_norm)

        feat = self.cnn(img)
        feat = feat.mean(dim=-1)
        feat = feat.permute(0, 2, 1)
        tokens = self.row_projection(feat) + self.positional_encoding[:,
:feat.shape[1], :]
        h = self.transformer(tokens)
```

```
        h_img = self.img_norm(h[:, -1, :])

        r = self.raw_encoder(x_revin.unsqueeze(1))
        r = r.mean(dim=-1)
        h_raw = self.raw_norm(r)

        h_fused = torch.cat([h_img, h_raw], dim=-1)
        y_hat = self.value_head(h_fused)
        dir_logits = self.direction_head(h_fused)
        return y_hat, dir_logits


def get_loss_fn(name: str) -> Callable:
    """Get loss function by name."""
    loss_functions = {
        "mse": lambda p, y: F.mse_loss(p, y),
        "mae": lambda p, y: F.l1_loss(p, y),
        "huber": lambda p, y: F.smooth_l1_loss(p, y, beta=1.0),
    }
    if name not in loss_functions:
        raise ValueError(f"Unknown loss function: {name}")
    return loss_functions[name]


class ModelTrainer:
    """Handles training and evaluation of forecasting models."""

    def __init__(self, cfg: BenchmarkConfig, is_finance: bool = False):
        self.cfg = cfg
        self.is_finance = is_finance
        self.dir_weight = cfg.dir_loss_weight_finance if is_finance else
cfg.dir_loss_weight_nonfinance

    def train(self, model: nn.Module, train_loader: DataLoader, val_loader:
DataLoader,
              name: str) -> Tuple[Dict, Dict]:
        device = self.cfg.device
        model.to(device)
        loss_fn = get_loss_fn(self.cfg.loss)

        optimizer = torch.optim.AdamW(model.parameters(), lr=self.cfg.lr,
                                      weight_decay=self.cfg.weight_decay)
        scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
            optimizer, mode="min", factor=self.cfg.lr_factor,
            patience=self.cfg.lr_patience, min_lr=self.cfg.min_lr
        )

        history = {
            "epoch": [], "train_loss": [], "val_loss": [], "lr": [],
            "sec": [], "samples_per_sec": [], "reg": [], "dir": [], "flat": []
        }

        best_val = float("inf")
        best_state = None
        best_epoch = 0
        no_improve = 0

        for epoch in range(1, self.cfg.epochs + 1):
            t0 = time.time()
```

```
            model.train()
            train_losses = []
            samples_seen = 0
            reg_accum, dir_accum, flat_accum, count = 0.0, 0.0, 0.0, 0

            for _, _, x_norm, y_norm, _, _, img in train_loader:
                x_norm = x_norm.to(device)
                y_norm = y_norm.to(device)
                img = img.to(device)

                output = model(x_norm, img)
                loss, reg, dir_loss, flat_pen = self._compute_loss(output,
y_norm, loss_fn)

                if not torch.isfinite(loss):
                    Logger.log(f"[{name}] Non-finite loss, skipping batch")
                    optimizer.zero_grad(set_to_none=True)
                    continue

                optimizer.zero_grad(set_to_none=True)
                loss.backward()
                nn.utils.clip_grad_norm_(model.parameters(), self.cfg.grad_clip)
                optimizer.step()

                train_losses.append(loss.item())
                samples_seen += x_norm.size(0)
                reg_accum += float(reg.item())
                dir_accum += float(dir_loss.item())
                flat_accum += float(flat_pen.item())
                count += 1

            val_loss = self._validate(model, val_loader, loss_fn, device)

            if val_loss is None:
                Logger.log(f"[{name}] All validation losses non-finite,
stopping")
                break

            elapsed = float(time.time() - t0)
            lr = float(optimizer.param_groups[0]["lr"])
            sps = float(samples_seen / max(elapsed, 1e-9))
            train_loss = float(np.mean(train_losses)) if train_losses else
float("inf")

            history["epoch"].append(epoch)
            history["train_loss"].append(train_loss)
            history["val_loss"].append(val_loss)
            history["lr"].append(lr)
            history["sec"].append(elapsed)
            history["samples_per_sec"].append(sps)
            history["reg"].append((reg_accum / count) if count > 0 else
float("nan"))
            history["dir"].append((dir_accum / count) if count > 0 else
float("nan"))
            history["flat"].append((flat_accum / count) if count > 0 else
float("nan"))

            scheduler.step(val_loss)
```

```
            Logger.log(f"[{name}] Epoch {epoch:03d} | train={train_loss:.4f}
val={val_loss:.4f} | "
                       f"reg={history['reg'][-1]:.4f} dir={history['dir'][-
1]:.4f} "
                       f"flat={history['flat'][-1]:.4f} | lr={lr:.2e} |
{elapsed:.1f}s | {sps:.1f} samp/s")

            if val_loss < best_val - 1e-6:
                best_val = val_loss
                best_epoch = epoch
                best_state = {k: v.detach().cpu() for k, v in
model.state_dict().items()}
                no_improve = 0
            else:
                no_improve += 1
                if no_improve >= self.cfg.early_stop_patience:
                    Logger.log(f"[{name}] Early stopping at epoch {epoch} "
                               f"(best_ep={best_epoch},
best_val={best_val:.4f})")
                    break

        if best_state is not None:
            model.load_state_dict(best_state)

        improvement_rate = 0.0
        if best_epoch > 1 and len(history["val_loss"]) >= best_epoch:
            improvement_rate = (history["val_loss"][0] -
history["val_loss"][best_epoch - 1]) / (best_epoch - 1)

        stats = {
            "best_epoch": float(best_epoch),
            "best_val_loss": float(best_val),
            "val_improve_per_epoch": float(improvement_rate)
        }
        return history, stats

    def _compute_loss(self, output, y_norm: torch.Tensor, loss_fn: Callable) ->
Tuple[torch.Tensor, ...]:
        device = y_norm.device

        if isinstance(output, tuple):
            pred, dir_logits = output
            reg = loss_fn(pred, y_norm)

            if self.dir_weight > 0:
                y_dir = (y_norm > 0).float()
                dir_loss = F.binary_cross_entropy_with_logits(dir_logits, y_dir)
            else:
                dir_loss = torch.zeros([], device=device)

            if self.is_finance and self.cfg.flat_penalty_weight > 0:
                pred_std = torch.std(pred, dim=1).mean()
                flat_pen = F.relu(self.cfg.flat_sigma_min - pred_std)
            else:
                flat_pen = torch.zeros([], device=device)

            loss = reg + self.dir_weight * dir_loss +
self.cfg.flat_penalty_weight * flat_pen
        else:
```

```
            pred = output
            loss = loss_fn(pred, y_norm)
            reg = loss
            dir_loss = torch.zeros([], device=device)
            flat_pen = torch.zeros([], device=device)

        return loss, reg, dir_loss, flat_pen

    def _validate(self, model: nn.Module, val_loader: DataLoader,
                  loss_fn: Callable, device: str) -> Optional[float]:
        model.eval()
        val_losses = []

        with torch.no_grad():
            for _, _, x_norm, y_norm, _, _, img in val_loader:
                x_norm = x_norm.to(device)
                y_norm = y_norm.to(device)
                img = img.to(device)

                output = model(x_norm, img)
                loss, _, _, _ = self._compute_loss(output, y_norm, loss_fn)

                if torch.isfinite(loss):
                    val_losses.append(loss.item())

        return float(np.mean(val_losses)) if val_losses else None

    @torch.no_grad()
    def predict(self, model: nn.Module, loader: DataLoader) ->
Tuple[torch.Tensor, torch.Tensor]:
        device = self.cfg.device
        model.eval()
        y_true_list, y_pred_list = [], []

        for _, y_orig, x_norm, _, mu, std, img in loader:
            x_norm = x_norm.to(device)
            img = img.to(device)
            mu = mu.to(device).unsqueeze(-1)
            std = std.to(device).unsqueeze(-1)
            y_orig = y_orig.to(device)

            output = model(x_norm, img)
            pred_norm = output[0] if isinstance(output, tuple) else output
            pred_orig = pred_norm * std + mu

            y_true_list.append(y_orig.detach().cpu())
            y_pred_list.append(pred_orig.detach().cpu())

        return torch.cat(y_true_list, dim=0), torch.cat(y_pred_list, dim=0)


class ARIMAForecaster:
    """ARIMA model wrapper for walk-forward forecasting."""

    def __init__(self, cfg: BenchmarkConfig):
        if not HAS_STATSMODELS:
            raise RuntimeError("statsmodels not installed (pip install
statsmodels)")
        self.cfg = cfg
```

```
    def forecast(self, train_series: np.ndarray, test_series: np.ndarray) ->
Tuple[np.ndarray, np.ndarray, float]:
        train_s = np.asarray(train_series, dtype=np.float32).reshape(-1)
        test_s = np.asarray(test_series, dtype=np.float32).reshape(-1)

        L, H = self.cfg.past_len, self.cfg.horizon
        max_i = len(test_s) - (L + H)
        if max_i < 0:
            raise ValueError("Test series too short.")

        indices = list(range(0, max_i + 1, max(1, self.cfg.arima_stride)))
        y_true, y_pred = [], []
        history = list(train_s.astype(float))
        t0 = time.time()

        for i in indices:
            reveal_upto = i + L
            while len(history) < len(train_s) + reveal_upto:
                history.append(float(test_s[len(history) - len(train_s)]))

            tail = np.asarray(history[-max(self.cfg.arima_tail_max, L * 5):],
dtype=float)

            try:
                result = ARIMA(tail, order=self.cfg.arima_order).fit()
                forecast = result.forecast(steps=H)
            except Exception:
                forecast = np.full((H,), tail[-1], dtype=np.float32)

            y_true.append(test_s[i + L:i + L + H].astype(np.float32))
            y_pred.append(np.asarray(forecast, dtype=np.float32))

        return np.stack(y_true), np.stack(y_pred), float(time.time() - t0)


class Visualizer:
    """Visualization utilities for benchmark results."""

    @staticmethod
    def save_figure(path: str) -> None:
        plt.tight_layout()
        plt.savefig(path, dpi=170)
        plt.close()

    @classmethod
    def plot_convergence(cls, history: Dict[str, List[float]], title: str,
out_path: str) -> None:
        plt.figure(figsize=(10, 6))
        plt.plot(history["epoch"], history["train_loss"], label="Train Loss",
linewidth=2)
        plt.plot(history["epoch"], history["val_loss"], label="Validation Loss",
linewidth=2)
        plt.xlabel("Epoch")
        plt.ylabel("Loss")
        plt.title(title)
        plt.grid(True, alpha=0.3)
        plt.legend()
        cls.save_figure(out_path)
```

```
    @classmethod
    def plot_forecast_example(cls, x_past: np.ndarray, y_true: np.ndarray,
                              y_pred: np.ndarray, title: str, out_path: str) ->
None:
        L, H = len(x_past), len(y_true)
        plt.figure(figsize=(12, 6))
        plt.plot(np.arange(L), x_past, label="History", linewidth=2)
        plt.plot(np.arange(L, L + H), y_true, label="Actual", linewidth=2)
        plt.plot(np.arange(L, L + H), y_pred, label="Predicted", linewidth=2,
linestyle="--")
        plt.axvline(L - 1, color="gray", linestyle=":", alpha=0.7)
        plt.xlabel("Time Index")
        plt.ylabel("Value")
        plt.title(title)
        plt.grid(True, alpha=0.3)
        plt.legend()
        cls.save_figure(out_path)


class DataLoader_:
    """Data loading utilities for financial and synthetic time series."""

    @staticmethod
    def load_financial_returns(ticker: str, period: str = "5y", interval: str =
"1d") -> np.ndarray:
        if not HAS_YFINANCE:
            raise RuntimeError("Install yfinance: pip install yfinance")

        df = yf.download(ticker, period=period, interval=interval,
progress=False)
        if df is None or len(df) < 400:
            raise RuntimeError(f"Insufficient data for {ticker}")

        col = "Adj Close" if "Adj Close" in df.columns else "Close"
        prices = df[col].dropna().to_numpy(dtype=np.float32).reshape(-1)
        log_prices = np.log(np.maximum(prices, 1e-6)).astype(np.float32)
        returns = np.diff(log_prices).astype(np.float32)
        return returns

    @staticmethod
    def generate_synthetic(kind: str, length: int, seed: int) -> np.ndarray:
        rng = np.random.default_rng(seed)
        t = np.arange(length, dtype=np.float32)

        if kind == "regime_switch":
            y = np.zeros(length, dtype=np.float32)
            level = 0.0
            for i in range(length):
                if i % 400 == 0:
                    level += rng.normal(0, 1.2)
                y[i] = level + 0.4 * np.sin(2 * np.pi * i / 50) + 0.25 *
rng.standard_normal()
            return y.astype(np.float32)

        if kind == "chirp_phase":
            chirp = np.sin(2 * np.pi * (t / 60.0 + 0.0008 * (t **
2))).astype(np.float32)
            phase_mod = (0.4 * np.sin(2 * np.pi * t / 17.0 +
```

```
                                        0.7 * np.sin(2 * np.pi * t /
200.0))).astype(np.float32)
            y = chirp + phase_mod
            y += 0.12 * rng.standard_normal(length).astype(np.float32)
            y = np.tanh(1.3 * y).astype(np.float32)
            return y

        if kind == "nonlinear_long_dep":
            y = np.zeros(length, dtype=np.float32)
            eps = 0.10 * rng.standard_normal(length).astype(np.float32)
            for i in range(2, length):
                seasonal = 0.6 * np.sin(2 * np.pi * i / 48) + 0.25 * np.sin(2 *
np.pi * i / 240)
                long_dep = 0.25 * (y[i - 48] if i - 48 >= 0 else 0.0)
                nonlin = 0.22 * np.tanh(y[i - 1])
                spike = rng.normal(0, 1.0) if (i % 333 == 0) else 0.0
                y[i] = 0.55 * y[i - 1] - 0.15 * y[i - 2] + seasonal + long_dep +
nonlin + eps[i] + spike
                y[i] = np.clip(y[i], -6.0, 6.0).astype(np.float32)
            return y.astype(np.float32)

        raise ValueError(f"Unknown synthetic data kind: {kind}")


class BenchmarkRunner:
    """Main benchmark runner for evaluating forecasting models."""

    def __init__(self, cfg: BenchmarkConfig):
        self.cfg = cfg

    def run_single_dataset(self, series: np.ndarray, dataset_name: str,
                           is_finance: bool, primary_metric: str) ->
pd.DataFrame:
        ensure_dir(self.cfg.out_dir)
        dataset_dir = os.path.join(self.cfg.out_dir, dataset_name)
        ensure_dir(dataset_dir)

        series = np.asarray(series, dtype=np.float32).reshape(-1)
        if not np.isfinite(series).all():
            raise ValueError(f"{dataset_name}: Contains non-finite values")
        if len(series) < (self.cfg.past_len + self.cfg.horizon + 200):
            raise ValueError(f"{dataset_name}: Series too short")

        train_s, val_s, test_s = time_split(series, self.cfg.train_ratio,
self.cfg.val_ratio)

        cache_paths = {
            "train": os.path.join(dataset_dir, "cache_train.dat") if
self.cfg.use_image_cache else None,
            "val": os.path.join(dataset_dir, "cache_val.dat") if
self.cfg.use_image_cache else None,
            "test": os.path.join(dataset_dir, "cache_test.dat") if
self.cfg.use_image_cache else None,
        }

        t_cache = time.time()
        train_ds = TimeSeriesDataset(train_s, self.cfg.past_len,
self.cfg.horizon,
```

```
                                        self.cfg.stride, self.cfg.mtf_bins,
cache_paths["train"])
        val_ds = TimeSeriesDataset(val_s, self.cfg.past_len, self.cfg.horizon,
                                   self.cfg.stride, self.cfg.mtf_bins,
cache_paths["val"])
        test_ds = TimeSeriesDataset(test_s, self.cfg.past_len, self.cfg.horizon,
                                    self.cfg.stride, self.cfg.mtf_bins,
cache_paths["test"])

        Logger.log(f"{dataset_name} | Windows: train={len(train_ds)}
val={len(val_ds)} "
                   f"test={len(test_ds)} | Cache: {time.time() - t_cache:.1f}s")

        if len(train_ds) == 0 or len(test_ds) == 0:
            raise ValueError(f"{dataset_name}: No windows produced")

        train_loader = DataLoader(train_ds, batch_size=self.cfg.batch_size,
shuffle=True, drop_last=True)
        val_loader = DataLoader(val_ds, batch_size=self.cfg.batch_size,
shuffle=False)
        test_loader = DataLoader(test_ds, batch_size=max(128,
self.cfg.batch_size), shuffle=False)

        results = []
        trainer = ModelTrainer(self.cfg, is_finance=is_finance)

        Logger.log(f"{dataset_name} | ARIMA (order={self.cfg.arima_order},
stride={self.cfg.arima_stride})")
        arima = ARIMAForecaster(self.cfg)
        y_true_a, y_pred_a, arima_time = arima.forecast(train_s, test_s)
        yt_a, yp_a = torch.tensor(y_true_a), torch.tensor(y_pred_a)
        arima_metrics = Metrics.evaluate_all_horizons(yt_a, yp_a)
        if is_finance:
            arima_metrics.update(Metrics.evaluate_direction_horizons(yt_a,
yp_a))
        else:
            arima_metrics.update(Metrics.evaluate_r2_horizons(yt_a, yp_a))
        arima_metrics.update({
            "dataset": dataset_name, "model": "ARIMA",
            "train_time_sec": arima_time, "best_epoch": 0.0,
            "val_improve_per_epoch": 0.0, "best_val_loss": 0.0,
            "primary_metric": primary_metric,
        })
        results.append(arima_metrics)
        Logger.log(f"{dataset_name} | ARIMA complete |
windows={y_true_a.shape[0]} | {arima_time:.1f}s")

        def train_and_evaluate(model: nn.Module, model_name: str) -> Dict[str,
float]:
            Logger.log(f"{dataset_name} | {model_name} training started")
            t0 = time.time()

            history, stats = trainer.train(model, train_loader, val_loader,
                                           name=f"{dataset_name}/{model_name}")
            train_time = float(time.time() - t0)

            Logger.log(f"{dataset_name} | {model_name} complete |
{train_time:.1f}s | "
```

```
                        f"best_ep={int(stats['best_epoch'])}
best_val={stats['best_val_loss']:.4f}")

            Visualizer.plot_convergence(
                history, f"{dataset_name} | {model_name} Convergence",
                os.path.join(dataset_dir,
f"{model_name.lower()}_convergence.png")
            )

            y_true, y_pred = trainer.predict(model, test_loader)
            metrics = Metrics.evaluate_all_horizons(y_true, y_pred)
            if is_finance:
                metrics.update(Metrics.evaluate_direction_horizons(y_true,
y_pred))
            else:
                metrics.update(Metrics.evaluate_r2_horizons(y_true, y_pred))

            metrics.update({
                "dataset": dataset_name, "model": model_name,
                "train_time_sec": train_time, **stats,
                "primary_metric": primary_metric,
            })

            example_idx = min(self.cfg.forecast_example_idx, y_true.shape[0] -
1)
            x_past = test_s[example_idx:example_idx + self.cfg.past_len]
            Visualizer.plot_forecast_example(
                x_past, y_true.numpy()[example_idx],
y_pred.numpy()[example_idx],
                f"{dataset_name} | {model_name} Forecast Example",
                os.path.join(dataset_dir, f"{model_name.lower()}_example.png")
            )

            return metrics

        device = self.cfg.device

        results.append(train_and_evaluate(
            DLinear(self.cfg.past_len, self.cfg.horizon).to(device), "DLinear"
        ))
        results.append(train_and_evaluate(
            LSTMForecaster(self.cfg.horizon).to(device), "LSTM"
        ))
        results.append(train_and_evaluate(
            PatchTST(self.cfg.past_len, self.cfg.horizon, self.cfg.patch_len,
                     self.cfg.patch_stride, self.cfg.d_model, self.cfg.depth,
                     self.cfg.nhead, self.cfg.dropout).to(device), "PatchTST"
        ))
        results.append(train_and_evaluate(
            HybridCNNTransformer(self.cfg.past_len, self.cfg.horizon,
self.cfg.img_channels,
                                 self.cfg.d_model, self.cfg.depth,
self.cfg.nhead,
                                 self.cfg.dropout).to(device), "Hybrid"
        ))

        df = pd.DataFrame(results).round(5)
        df.to_csv(os.path.join(dataset_dir, "leaderboard.csv"), index=False)
        return df
```

```
    @staticmethod
    def generate_summary(full_results: pd.DataFrame) -> pd.DataFrame:
        higher_is_better = {"R2_all": True, "DA_all": True, "HitRate_all": True}
        summary_rows = []

        for dataset in sorted(full_results["dataset"].unique()):
            subset = full_results[full_results["dataset"] == dataset].copy()
            primary_metric = str(subset["primary_metric"].iloc[0])

            if primary_metric not in subset.columns:
                primary_metric = "RMSE_all" if "RMSE_all" in subset.columns else
subset.columns[0]

            ascending = not higher_is_better.get(primary_metric, False)
            subset = subset.sort_values(primary_metric,
ascending=ascending).reset_index(drop=True)
            subset["rank"] = np.arange(1, len(subset) + 1)

            winner = subset.iloc[0]["model"]
            hybrid_row = subset[subset["model"] == "Hybrid"]

            summary_rows.append({
                "dataset": dataset,
                "primary_metric": primary_metric,
                "winner_model": winner,
                "hybrid_rank": int(hybrid_row["rank"].iloc[0]) if
len(hybrid_row) > 0 else None,
                "hybrid_score": float(hybrid_row[primary_metric].iloc[0]) if
len(hybrid_row) > 0 else None,
                "best_score": float(subset.iloc[0][primary_metric]),
                "second_score": float(subset.iloc[1][primary_metric]) if
len(subset) > 1 else None,
            })

        return pd.DataFrame(summary_rows)


def main():
    """Run the complete forecasting benchmark."""
    cfg = BenchmarkConfig()
    cfg.device = get_device(cfg.device)
    set_seed(cfg.seed)
    ensure_dir(cfg.out_dir)

    Logger.log(f"Device: {cfg.device} | Epochs: {cfg.epochs} | Batch:
{cfg.batch_size} | "
               f"Past: {cfg.past_len} | Horizon: {cfg.horizon}")
    Logger.log(f"Hybrid (finance): dir_weight={cfg.dir_loss_weight_finance} "
               f"flat_penalty={cfg.flat_penalty_weight}
sigma_min={cfg.flat_sigma_min}")
    Logger.log(f"Hybrid (non-finance):
dir_weight={cfg.dir_loss_weight_nonfinance}")
    Logger.log(f"ARIMA: order={cfg.arima_order} stride={cfg.arima_stride}")
    Logger.log(f"Image cache: {'enabled' if cfg.use_image_cache else
'disabled'}")

    datasets = [
```

```
        {"name": "financial_SPY", "type": "finance", "ticker": "SPY",
"primary_metric": "RMSE_all"},
        {"name": "financial_AAPL", "type": "finance", "ticker": "AAPL",
"primary_metric": "DA_all"},
        {"name": "synthetic_chirp_phase", "type": "synth", "kind":
"chirp_phase", "primary_metric": "RMSE_all"},
        {"name": "synthetic_regime_switch", "type": "synth", "kind":
"regime_switch", "primary_metric": "RMSE_all"},
    ]

    runner = BenchmarkRunner(cfg)
    all_results = []

    for dataset_config in datasets:
        dataset_name = dataset_config["name"]
        primary_metric = dataset_config["primary_metric"]

        Logger.log("=" * 92)
        Logger.log(f"DATASET: {dataset_name} | Primary Metric:
{primary_metric}")
        Logger.log("=" * 92)

        try:
            if dataset_config["type"] == "finance":
                series =
DataLoader_.load_financial_returns(dataset_config["ticker"], period="5y",
interval="1d")
                is_finance = True
            else:
                series = DataLoader_.generate_synthetic(dataset_config["kind"],
length=6000, seed=cfg.seed)
                is_finance = False

            Logger.log(f"{dataset_name} stats | mean={np.mean(series):.6f}
std={np.std(series):.6f} "
                       f"min={np.min(series):.6f} max={np.max(series):.6f}")

            df = runner.run_single_dataset(series, dataset_name,
is_finance=is_finance,
                                           primary_metric=primary_metric)
            all_results.append(df)

        except Exception as e:
            Logger.log(f"Skipping {dataset_name}: {e}")

    if not all_results:
        Logger.log("No datasets evaluated. Exiting.")
        return

    full_results = pd.concat(all_results, ignore_index=True)
    full_path = os.path.join(cfg.out_dir, "leaderboard_all.csv")
    full_results.to_csv(full_path, index=False)
    Logger.log(f"Saved: {full_path}")

    summary = BenchmarkRunner.generate_summary(full_results)
    summary_path = os.path.join(cfg.out_dir, "summary.csv")
    summary.to_csv(summary_path, index=False)
    Logger.log(f"Saved: {summary_path}")
```

```
    print("\n" + "=" * 60)
    print("BENCHMARK SUMMARY")
    print("=" * 60)
    print(summary.to_string(index=False))
    print("\n" + "=" * 60)
    print("FULL LEADERBOARD (Top 30)")
    print("=" * 60)
    print(full_results.head(30).to_string(index=False))

    Logger.log(f"All outputs saved to: {cfg.out_dir}")


if __name__ == "__main__":
    main()
```

ДОДАТОК Б Графічні матеріали

Гібридний метод прогнозування часових рядів на основі трансформерів та комп'ютерного зору

Кафедра математичних методів системного аналізу

Студент: Свирид Нікіта Олександрович, Ка-41мп

Керівник: Недашківська Надія Іванівна, д.т.н.

Актуальність

- Часові ряди в економіці та фінансах: висока **волатильність** і **нестаціонарність**
- Часті **режимні переходи** та приховані структурні зміни
- Потреба у методі, що одночасно враховує **локальні патерни** й **глобальний контекст**

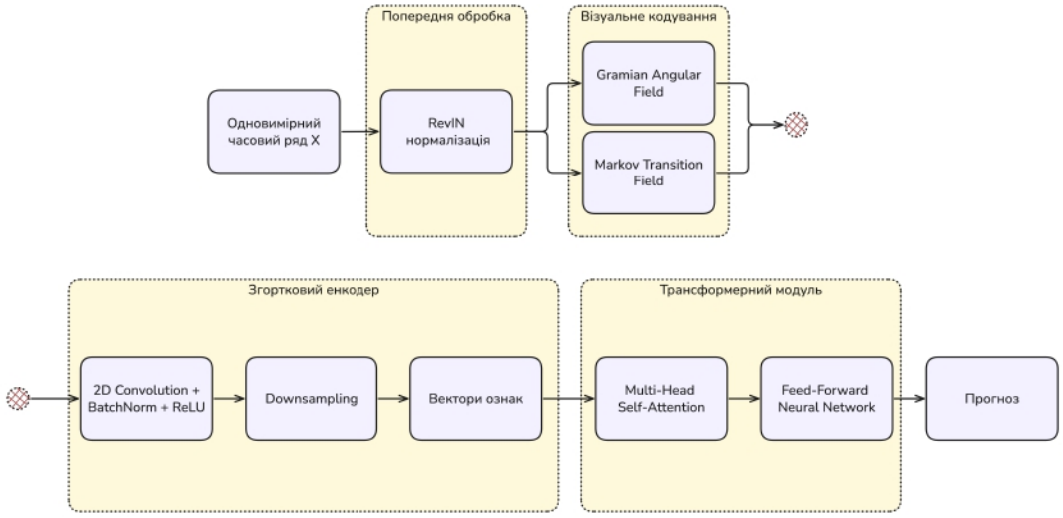
Об'єкт, предмет і мета дослідження

- **Об'єкт дослідження:** процеси прогнозування часових рядів у соціально-економічних та технічних системах (фінансові ринки, енергетика тощо).
- **Предмет дослідження:** методи та моделі прогнозування часових рядів на основі візуального кодування (GAF/MTF), згорткових мереж (CNN) і механізмів уваги (Transformer).
- **Мета дослідження:** підвищити якість багатогоризонтного прогнозування ($t+1$, $t+3$, $t+6$, $t+12$) шляхом розробки гібридної архітектури **CV + Transformer** та експериментального порівняння з базовими підходами.

Запропонований підхід

- Перетворення 1D сигналу у 2D: **GAF + MTF** (два канали)
- **CNN-енкодер** вивчає локальні просторові патерни
- **Transformer** моделює глобальні залежності між векторами ознак

Архітектура гібридної моделі



Дані та бенчмарки

Реальні: real_AAPL, real_SPY
Синтетичні: synth_chirp_phase, synth_regime_switch

Порівнювані моделі: ARIMA, DLinear, LSTM, PatchTST, Hybrid

Метрики

- **MAE, RMSE** — абсолютна похибка
- **sMAPE** — відносна похибка (стабільна при малих значеннях)
- **DA** — точність прогнозу напряму (важливо для фінансів)
- **Час навчання** — практична придатність

На реальних фінансових даних R^2 не використовувався як основна метрика.

Опис використаних даних

- Для експериментального дослідження використано **синтетичні та реальні часові ряди**, що дозволяє оцінити поведінку моделей у контрольованих та прикладних умовах.
- **Синтетичні датасети:**
  - *synth_chirp_phase* — нелінійний сигнал зі змінною частотою, що імітує фазові зсуви та складну динаміку.
  - *synth_regime_switch* — часовий ряд зі змінами режимів, що моделює структурні злами та нестабільну поведінку системи.
- **Реальні датасети:**
  - *real_AAPL*, *real_SPY* — фінансові часові ряди (щоденні ціни закриття), отримані з Yahoo Finance.
- Дані поділялися на навчальну, валідаційну та тестову вибірки з використанням ковзного вікна (look-back) та багатогоризонтного прогнозу.

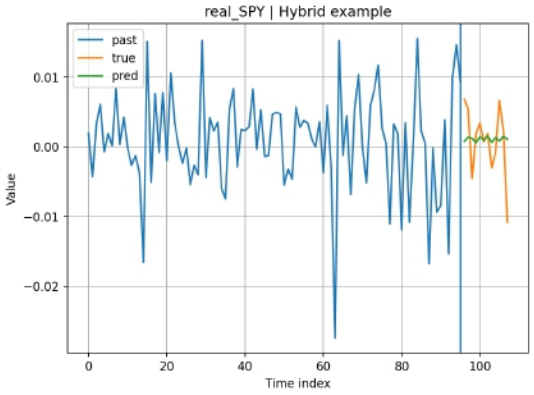
Загальне порівняння моделей (агреговані метрики)

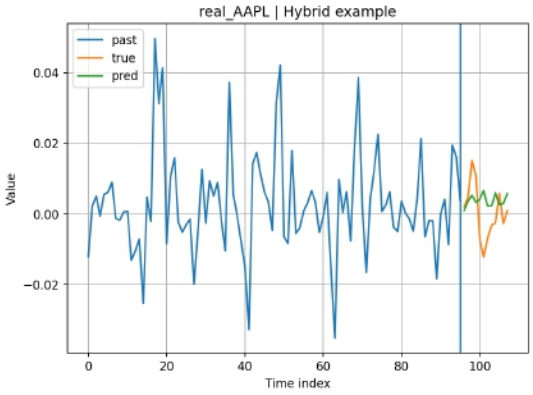
Модель	DA	sMAPE	RMSE	MAE	Час навчання
Hybrid	0.274	109.888	0.231	0.190	353.482
LSTM	0.271	119.933	0.244	0.211	25.401
DLinear	0.269	114.290	0.248	0.214	10.773
GARCH	0.273	109.681	0.232	0.194	15.382
PatchTST	0.264	99.544	0.195	0.143	21.370

Реальні фінансові дані - порівняння

Датасет	Модель	DA	sMAPE	RMSE	MAE	Час
real_AAPL						
	(arper.)	0.516	156.342	0.015	0.011	20.251
	Hybrid	0.533	148.460	0.014	0.010	12.400
	LSTM	0.519	157.519	0.014	0.010	1.315
	PatchTST	0.516	153.726	0.015	0.011	1.016
	GARCH	0.515	155.746	0.014	0.010	4.650
	DLinear	0.497	148.256	0.016	0.012	0.869
real_SPY						
	(arper.)	0.562	150.499	0.008	0.006	27.359
	DLinear	0.580	137.721	0.008	0.006	1.768
	LSTM	0.566	154.047	0.007	0.005	1.580
	Hybrid	0.562	147.522	0.007	0.005	17.772
	GARCH	0.562	153.675	0.007	0.005	5.672
	PatchTST	0.540	149.529	0.007	0.006	1.568

Прогноз гібридної моделі на реальних даних

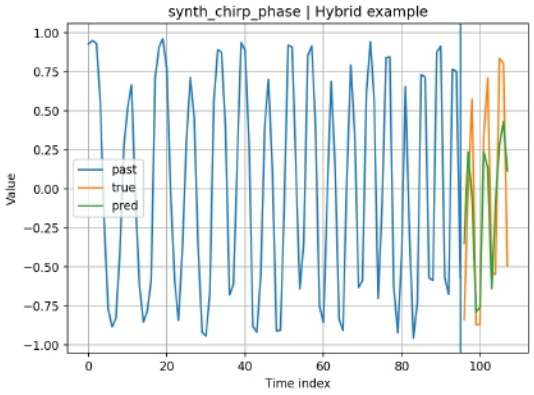


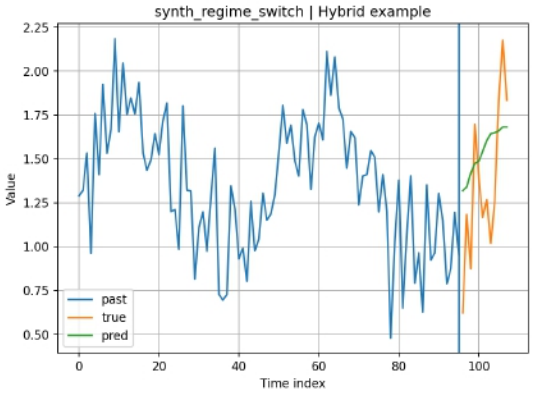


Синтетичні набори даних - Порівняння

Датасет	Модель	R^2	sMAPE	RMSE	MAE	Час
synth_chirp_phase (arper.): R^2=0.765, sMAPE=136.031, RMSE=0.626, MAE=0.544, Time=202.725						
	GARCH	0.234	123.782	0.704	0.650	15.615
	DLinear	0.027	150.109	0.677	0.613	3.792
	Hybrid	0.212	122.533	0.609	0.515	165.184
	LSTM	0.035	147.845	0.675	0.613	7.511
	PatchTST	0.542	73.883	0.465	0.330	10.623
synth_regime_switch (arper.): R^2=3.392, sMAPE=22.199, RMSE=0.309, MAE=0.241, Time=217.073						
	GARCH	0.701	20.523	0.389	0.309	31.445
	DLinear	0.723	21.075	0.289	0.226	4.344
	Hybrid	0.716	21.037	0.293	0.228	158.126
	LSTM	0.743	20.322	0.279	0.217	14.995
	PatchTST	0.712	21.039	0.295	0.227	8.163

Прогноз гібридної моделі на синтетичних наборах





Аналіз роботи моделі - переваги

Сильні сторони моделі (де Hybrid показує перевагу):

- **Нелінійні та режимні процеси** (synth_regime_switch, synth_chirp_phase):
  - Візуальне кодування (GAF/MTF) переводить часову динаміку у структурні патерни.
  - CNN ефективно виявляє локальні інваріантні ознаки (зміни режимів, фазові переходи).
  - Transformer агрегує ці ознаки у глобальний контекст, що покращує DA та sMAPE.
- **Фінансові часові ряди** (real_AAPL, real_SPY):
  - Модель краще адаптується до змін волатильності та ринкових режимів.
  - Вища Directional Accuracy свідчить про коректне визначення напряму руху.

Аналіз роботи моделі - недоліки

Обмеження моделі:

- **Прості або майже лінійні сигнали**:
  - Візуальне кодування вводить надлишкову складність.
  - CNN інваріантний до зсуву, що ускладнює точну фазову екстраполяцію.
- **Обчислювальні витрати**:
  - Генерація GAF/MTF + глибока архітектура ⇒ у 5–10 разів більший час навчання.
  - Модель менш придатна для low-latency або edge-сценаріїв.

Наукова новизна роботи

- Запропоновано новий гібридний підхід до прогнозування часових рядів, що поєднує візуальне кодування (GAF, MTF), згорткові нейронні мережі та трансформерні механізми уваги.
- Вперше використано багатоканальне візуальне представлення часового ряду, яке одночасно відображає статичні кореляції та стохастичну динаміку переходів станів.
- Показано, що перехід у візуальний простір підвищує ефективність моделювання нелінійних, хаотичних та режимних процесів.
- Експериментально доведено спеціалізацію гібридної моделі: перевагу для складних динамічних систем та обмеження для лінійних задач.
- Отримані результати розширюють застосування методів комп'ютерного зору у задачах прогнозування часових рядів.

Висновки

- Реалізовано pipeline: **GAF/MTF → CNN → Transformer → multi-horizon forecast**
- На реальних фінансових даних Hybrid показує конкурентну якість за **DA/MAE/RMSE/sMAPE**
- Головний практичний компроміс: **час навчання** Hybrid значно більший (генерація зображень + складність)
- Наступні кроки: кешування/батчинг GAF/MTF, спрощення CNN, підбір гіперпараметрів під класи рядів

Дякую за увагу!