

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.912

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

« ____ » _____ 2024 р.

Магістерська дисертація

на здобуття ступеня магістра

освітньо–науковою програмою

**«Інженерія програмного забезпечення мультимедійних та
інформаційно–пошукових систем»**

зі спеціальності 121 Інженерія програмного забезпечення

**на тему: «Програмний метод мелодійного викривлення
аудіоматеріалів за допомогою штучного інтелекту»**

Виконав:

студент II курсу, групи КП–21мн

Анікеєв Ігор Анатолійович _____

Керівник:

Доцент кафедри ПЗКС, к.ф.–м.н., доцент,

Нещадим Олександр Михайлович _____

Консультант з нормоконтролю:

Доцент кафедри ПЗКС, к.т.н., доцент,

Онай Микола Володимирович _____

Рецензент:

Доцент кафедри СПСКС, к.т.н., с.н.с.,

Боярінова Юлія Віталіївна _____

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

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення»

Освітньо–наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно–пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«__» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Анікєєву Ігорю Анатолійовичу

1. Тема дисертації « Програмний метод мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту », науковий керівник дисертації Нещадим Олександр Михайлович, к.ф.–м.н., доцент, затверджені наказом по університету від «27» березня 2024 р. № 1478–С.
2. Термін подання студентом дисертації «13» червня 2024 р.
3. Об'єкт дослідження: процес процес мелодійної обробки та покращення якості звуку в режимі реального часу за допомогою алгоритмів на основі ШІ.
4. Предмет дослідження: методи, алгоритми та програмні засоби CNN, методи обробки сигналів, WaveNet.
5. Перелік завдань, які потрібно розробити:
 - Дослідити та проаналізувати існуючі методи мелодійного викривлення та їхні обмеження;
 - Запропонувати новий метод, використовуючи модифіковані архітектури нейронних мереж для швидкого перетворення аудіоданих;
 - Оптимізувати алгоритм для обробки в реальному часі;
 - Експериментально дослідити ефективність розробленого методу шляхом оцінки роботи ПЗ за критеріями;
 - Провести аналіз отриманих результатів.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
 - Блок–схема запропонованого методу (плакат);
 - Діаграма розгалуження для попередньої обробки даних (плакат);
 - Демонстрація процесу нормалізації (плакат);
 - Робота фільтру FIR (плакат);
 - Діаграма компонентів системи мелодійного викривлення (плакат);

- Графік похибки стабільності (плакат);
- Порівняльний графік часу обробки сигналу (плакат);
- Графік функції втрати (плакат);

7. Орієнтовний перелік публікацій:

- Тези доповіді “Програмний метод мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту”
- Публікація статті у фаховому журналі за спеціальністю 121 «Інженерія програмного забезпечення»

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

9. Дата видачі завдання «11» жовтня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ознайомлення з предметною галуззю	15.12.2022	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.03.2023	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.05.2023	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення;	24.10.2023	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження; підготовка матеріалів доповіді на конференції ПМК–2023	15.12.2023	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	20.02.2024	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу	16.04.2024	
8.	Оформлення текстової та графічної частини магістерської дисертації	20.05.2024	

Студент

Ігор АНІКЄВ

Науковий керівник

Олександр НЕЩАДИМ

РЕФЕРАТ

Актуальність теми. Традиційні методи часто не забезпечують належний баланс між якістю звуку та зручністю використання, що стимулює необхідність нових підходів. Використання глибокого навчання, зокрема згорткових нейронних мереж, відкриває можливості для створення складних та реалістичних аудіоперетворень. Ці технології дозволяють значно підвищити ефективність обробки звуку в реальному часі, що є критично важливим для музикантів та звукоінженерів. Тема дослідження сприяє розвитку передових методів, які поєднують технічну досконалість із творчим потенціалом, змінюючи підхід до виробництва та обробки аудіо.

Об'єктом дослідження є процес мелодійної обробки та покращення якості звуку в режимі реального часу за допомогою алгоритмів на основі ШІ.

Предметом дослідження є методи обробки сигналів, CNN, перетворення Фур'є та вейвлет–перетворення, мелчастотні кепстральні коефіцієнти, WaveNet.

Мета роботи: визначення оптимальної комбінації методів і алгоритмів, які призведуть до мелодійного викривлення, а також у розробці моделей, які будуть інтегровані в програмне забезпечення для практичного використання.

Методи дослідження. В роботі використовуються сучасні методи глибокого навчання, такі як згорткові нейронні мережі (CNN). Також застосовуються алгоритми обробки сигналів та техніки вилучення ознак і спектральний аналіз. Для оцінки ефективності розроблених моделей використовуються методи валідації та тестування на основі реальних аудіоданих.

Наукова новизна. Запропоновано метод мелодійного викривлення аудіоматеріалів, що базується на використанні модифікованих архітектур згорткових нейронних мереж WaveNet. Вперше розроблено механізм

зворотного зв'язку для покращення продуктивності та точності моделей у реальному часі, що дозволяє досягти високої якості звукових ефектів без втрати швидкості обробки.

Практична значущість. На основі розробленого методу створено програмне забезпечення для мелодійного викривлення аудіоматеріалів, яке може бути інтегровано у професійні музичні та звукові студії. Це програмне забезпечення дозволяє значно підвищити якість звуку та розширити можливості креативного маніпулювання аудіо, що робить його незамінним інструментом для музикантів, продюсерів та звукоінженерів.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювалися на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг ПМК–2023» (Київ, 29–30 листопада 2023 р.). Стаття «Алгоритмічно–програмний метод визначення активного мовця» в науковому збірнику "InterConf": з матеріалами 3–ї Міжнародної науково–практичної конференції "Прогресивна наука і досягнення".

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі надано загальний огляд сучасного стану технологій мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту. Описано актуальність теми в контексті швидкого розвитку індустрії цифрового аудіо та зростаючого попиту на інноваційні та креативні інструменти для обробки звуку. Вказано на ключові виклики, пов'язані з досягненням балансу між якістю звуку та зручністю використання, а також підкреслено важливість нових підходів, що використовують глибоке навчання. Окреслено мету роботи, яка полягає у розробці та впровадженні оптимальних методів і алгоритмів мелодійного викривлення, що забезпечать високу точність і ефективність обробки звуку в реальному часі. Представлено об'єкт та предмет дослідження, а також методи, що використовуватимуться для досягнення поставленої мети.

У першому розділі розглянуто сучасні методи мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту, аналізуючи їхні можливості та обмеження. Детально описано потребу індустрії цифрового аудіо у нових креативних інструментах для обробки звуку, які здатні збалансувати якість звуку та зручність використання. Проаналізовано використання нейронних мереж для обробки звуку, зокрема гібридні моделі, що поєднують CNN та RNN, а також генеративні змагальні мережі (GAN) для синтезу нових аудіоданих. Розглянуто методи вилучення ознак, такі як MFCC та спектральні характеристики, для покращення аналізу та обробки аудіосигналів. Обговорено переваги трансферного навчання та навчання з підкріпленням у контексті створення аудіоплагінів. Наведено огляд існуючих аудіоплагінів, таких як Baby Audio TAI, Guitar Rig 7, iZotope Ozone 9, LANDR та iZotope Neutron 3, їхні функціональні можливості та застосування.

У другому розділі запропоновано алгоритмічно–програмний метод мелодійного викривлення аудіоматеріалів, що базується на використанні модифікованих архітектур згорткових нейронних мереж WaveNet. Детально описано етапи попередньої обробки аудіоданих, включаючи нормалізацію сигналу та виділення ключових характеристик. Представлено механізм зворотного зв'язку для покращення продуктивності та точності моделей у реальному часі. Описано процес оптимізації моделі шляхом налаштування гіперпараметрів та застосування алгоритмів регуляризації для запобігання перенавчанню. Розглянуто реалізацію моделі на платформі, що забезпечує мінімізацію затримки та оптимізацію обчислювальних ресурсів для роботи в реальному часі. Проведено валідацію та тестування моделі на основі тестових аудіоматеріалів, зокрема за допомогою показника співвідношення помилки до сигналу (ESR), що дозволяє оцінити ефективність запропонованого методу.

У третьому розділі наведено архітектуру та деталі реалізації розробленого програмного забезпечення для мелодійного викривлення

аудіоматеріалів. Описано структурні компоненти системи, включаючи основні модулі та їх взаємодію. Представлено алгоритми та програмні рішення, що використовуються для обробки аудіосигналів у реальному часі. Пояснено процеси інтеграції розробленої системи з існуючими аудіоплатформами та інструментами, а також розглянуто питання сумісності та масштабованості. Детально описано методи оптимізації продуктивності системи для забезпечення стабільної та швидкої обробки аудіо. Наведено приклади використання програмного забезпечення у різних сценаріях аудіовиробництва, підкреслюючи його переваги та можливості.

У четвертому розділі проведено детальний аналіз результатів експериментального дослідження ефективності розробленого методу мелодійного викривлення аудіоматеріалів. Виконано порівняння запропонованого методу з існуючими аналогами за ключовими параметрами, такими як якість звуку, швидкість обробки та стійкість до шуму. Оцінено точність і продуктивність моделі в різних умовах та на різних типах аудіоматеріалів. Проведено аналіз отриманих результатів за допомогою показників співвідношення помилки до сигналу (ESR). Обговорено сильні та слабкі сторони розробленого методу, а також можливі напрямки його подальшого вдосконалення. Наведено висновки щодо практичної значущості результатів дослідження та їх застосування у професійному аудіовиробництві та звуковому дизайні.

У висновках проаналізовано отримані результати дослідження.

У додатках наведено графічні матеріали до пояснювальної записки, скріншоти презентації, та лістинг коду розробленого програмного забезпечення.

Робота виконана на 80 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 22 найменувань. У роботі наведено 10 рисунків, 2 таблиці.

Ключові слова: мелодійне викривлення аудіоматеріалів, штучний інтелект, глибоке навчання, згорткові нейронні мережі, WaveNet, обробка

аудіо в реальному часі, машинне навчання, CNN, алгоритми обробки сигналів, модифікація архітектур нейронних мереж, аудіоплагіни, генеративні змагальні мережі, спектральний аналіз, оптимізація моделі, валідація, тестування аудіоданих, музичне виробництво, звуковий дизайн.

ABSTRACT

Theme urgency. Traditional methods often fail to strike the right balance between sound and usability, which encourages the use of new approaches. The use of deep learning, in particular convolutional neural networks, opens up possibilities for creating complex and realistic audio transformations. This technology should significantly improve the efficiency of real-time audio processing, which is a critical area for musicians and sound engineers. Research topic to improve the development of advanced methods that combine technical excellence with creative potential, changing the approach to audio production and processing.

Object of research is a process of melodic processing and improvement of sound quality in real time using AI-based algorithms.

Subject of research is signal processing methods, CNN, WaveNet.

Research objective is to determine the optimal combination of methods and algorithms that will lead to melodic distortion, as well as in the development of models that will be integrated into software for practical use.

Research methods. The thesis uses modern deep learning methods such as convolutional neural networks (CNN). Signal processing algorithms and feature extraction techniques are also used, including fine-grained cepstral coefficients (MFCC) and spectral analysis. Validation and testing methods based on real audio data are used to evaluate the effectiveness of the developed models.

Scientific novelty. A method of melodic distortion of audio materials based on the use of modified WaveNet convolutional neural network architectures is proposed. For the first time, a feedback mechanism has been developed to improve the performance and accuracy of real-time models, allowing you to achieve high quality sound effects without losing processing speed.

Practical value. Based on the developed method, software for melodic distortion of audio materials was created, which can be integrated into professional music and sound studios. This software allows you to significantly

improve sound quality and expand the possibilities of creative audio manipulation, making it an indispensable tool for musicians, producers and sound engineers.

Approbation. The basic results of the work were presented and discussed at the scientific conference of master's and postgraduate students "Applied mathematics and computing PMK–2023" (Kyiv, November 29–30, 2023). The article "Algorithmic–program method for determining an active speaker" in the scientific collection "InterConf", (200): with materials of the 3rd International Scientific and Practical Conference "Progressive Science and Achievements".

Structure and content of the thesis. The master thesis consists of the introduction, four chapters, conclusions, and appendixes.

The introduction provides a general overview of the current state of technology for melodic distortion of audio materials using artificial intelligence. The relevance of the topic in the context of the rapid development of the digital audio industry and the growing demand for innovative and creative sound processing tools is described. Key challenges related to achieving a balance between sound quality and usability are pointed out, and the importance of new approaches using deep learning is highlighted. The purpose of the work is outlined, which consists in the development and implementation of optimal methods and algorithms of melodic distortion, which will ensure high accuracy and efficiency of sound processing in real time. The object and subject of the research, as well as the methods that will be used to achieve the goal, are presented.

In the first section modern methods of melodic distortion of audio materials with the help of artificial intelligence are considered, analyzing their possibilities and limitations. It details the digital audio industry's need for creative new audio processing tools that balance sound quality and usability. The use of neural networks for sound processing is analyzed, in particular, hybrid models combining CNN and RNN, as well as generative adversarial networks (GAN) for the synthesis of new audio data. Feature extraction techniques such as MFCC and

spectral features are reviewed to improve the analysis and processing of audio signals. The benefits of transfer learning and reinforcement learning in the context of creating audio plugins are discussed. An overview of existing audio plug-ins such as Baby Audio TAIIP, Guitar Rig 7, iZotope Ozone 9, LANDR and iZotope Neutron 3, their functionality and applications is provided.

In the second section, an algorithmic-program method of melodic distortion of audio materials is proposed, based on the use of modified WaveNet convolutional neural network architectures. The audio data preprocessing steps are described in detail, including signal normalization and key feature extraction. A feedback mechanism is presented to improve the performance and accuracy of real-time models. The process of optimizing the model by adjusting hyperparameters and applying regularization algorithms to prevent overtraining is described. The implementation of the model on a platform that ensures delay minimization and optimization of computing resources for real-time operation is considered. Validation and testing of the model was carried out on the basis of audio test materials, in particular with the help of the error-to-signal ratio (ESR) indicator, which makes it possible to evaluate the effectiveness of the proposed method.

The third section provides the architecture and implementation details of the developed software for melodic distortion of audio materials. The structural components of the system are described, including the main modules and their interaction. Algorithms and software solutions used for processing audio signals in real time are presented. The processes of integrating the developed system with existing audio platforms and tools are explained, and the issues of compatibility and scalability are also considered. Methods for optimizing system performance to ensure stable and fast audio processing are described in detail. Examples of using the software in various audio production scenarios are given, highlighting its advantages and capabilities.

In the fourth chapter provides a detailed analysis of the results of an experimental study of the effectiveness of the developed method of melodic

distortion of audio materials. A comparison of the proposed method with existing analogues was made in terms of key parameters such as sound quality, processing speed and noise immunity. The accuracy and performance of the model were evaluated in different conditions and on different types of audio materials. Analysis of the obtained results was carried out using the error-to-signal ratio (ESR) indicators. The strengths and weaknesses of the developed method are discussed, as well as possible directions for its further improvement. Conclusions are given regarding the practical significance of the research results and their application in professional audio production and sound design.

The results of the study are analyzed in the conclusions.

The appendices provide graphic materials for the explanatory note, screenshots of the presentation and code listing of the developed software.

The work is performed on 80 sheets, contains 3 appendices and links to the list of used literature sources from 22 titles. The paper presents 10 figures, 2 tables.

Keywords: melodic distortion of audio materials, artificial intelligence, deep learning, convolutional neural networks, WaveNet, real-time audio processing, machine learning, CNN, signal processing algorithms, modification of neural network architectures, audio plugins, generative adversarial networks, spectral analysis, model optimization, validation, audio data testing, music production, sound design.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	3
ВСТУП.....	4
1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ МЕЛОДІЙНОГО ВИКРИВЛЕННЯ АУДІОМАТЕРІАЛІВ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ	6
1.1. Опис проблеми нестачі креативних плагінів.....	6
1.2. Алгоритми та методи використані в аудіо плагінах на основі штучного інтелекту	8
1.3. Моделі машинного навчання	9
1.4. Огляд існуючих програмних рішень	21
1.5. Висновки до першого розділу.....	28
2. МЕТОД МЕЛОДІЙНОГО ВИКРИВЛЕННЯ	31
2.1. Узагальнений опис запропонованого методу.....	31
2.2. Попередня обробка даних	34
2.3. Структура моделі	39
2.4. Архітектура моделі	44
2.5. Висновки до другого розділу	48
3. ОСОБЛИВОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МЕЛОДІЙНОГО ВИКРИВЛЕННЯ АУДІОМАТЕРІАЛІВ	50
3.1. Вимоги до ПЗ.....	50
3.2. Вибір мови програмування	51
3.3. Вибір бібліотек	57
3.4. Архітектура системи	60
3.5. Висновки до третього розділу.....	63
4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	65
4.1. Error-to-Signal Ratio.....	65
4.2. Процесорна швидкість.....	68
4.3. Огляд результатів	69
4.4. Висновки до четвертого розділу	72
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ	76
ДОДАТКИ	79

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CNN (Convolutional Neural Network) – Згорткова нейронна мережа – тип глибокої нейронної мережі, який використовується в основному для аналізу зображень та обробки сигналів.

FIR (Finite Impulse Response) – Кінцевий імпульсний відгук – тип цифрового фільтра, який використовується для обробки сигналів.

GAN (Generative Adversarial Network) – Генеративно-змагальна мережа – архітектура нейронної мережі, що використовується для генерації нових даних, подібних до вихідних даних.

MFCC (Mel-Frequency Cepstral Coefficients) – Мел-частотні кепстральні коефіцієнти – коефіцієнти, які є результатом перетворення звукових сигналів і використовуються в розпізнаванні мовлення.

MIMO (Multiple Input Multiple Output) – Багатоканальний вхід/вихід – технологія, яка використовує декілька антен для одночасної передачі та прийому сигналів.

RNN (Recurrent Neural Network) – Рекурентна нейронна мережа – тип нейронної мережі, яка здатна зберігати стан і використовувати його для обробки послідовностей даних.

ReLU (Rectified Linear Unit) – Випрямлена лінійна функція – тип функції активації, що широко використовується в нейронних мережах.

ADAM (Adaptive Moment Estimation) – Адаптивна оцінка моментів – метод оптимізації, що використовується для налаштування параметрів нейронних мереж.

WAV (Waveform Audio File Format) – Формат файлу аудіохвилі – стандартний формат цифрового аудіо, який використовується для зберігання звукових даних на комп'ютерах.

ESR (Error-to-Signal Ratio) – Співвідношення помилки до сигналу – показник, який використовується для оцінки ефективності алгоритмів обробки сигналів.

ВСТУП

У сучасному світі, де інформаційні технології еволюціонують з надзвичайною швидкістю, методи обробки аудіоматеріалів за допомогою штучного інтелекту стають все більш актуальними. Вони є ключовими елементами у різних сферах, включаючи музичне виробництво, звуковий дизайн, мультимедійні платформи та багато інших. Ці методи надають можливість автоматично викривлювати мелодії та покращувати якість звуку в реальному часі, що сприяє підвищенню креативності та ефективності обробки аудіоматеріалів.

Водночас, точність та ефективність мелодійного викривлення можуть бути піддані різним випробуванням, включаючи шум у фонових даних, складність аудіосигналів та обмеженість обчислювальних ресурсів. Відповідно до цих викликів, ця дисертація представляє новий підхід та програмне забезпечення, спрямовані на підвищення якості мелодійного викривлення аудіоматеріалів за допомогою модифікованих архітектур нейронних мереж.

Об'єктом дослідження є процес мелодійного викривлення та покращення якості звуку в режимі реального часу за допомогою алгоритмів на основі ШІ. Предметом дослідження є методи, алгоритми та програмні засоби для реалізації цих процесів.

Метою роботи є розробка та впровадження оптимальних методів і алгоритмів мелодійного викривлення аудіоматеріалів, що забезпечать високу точність і ефективність обробки звуку в реальному часі. Для досягнення цієї мети пропонується інноваційний метод, що базується на використанні згорткових нейронних мереж та механізмів зворотного зв'язку.

Ця система має перевершити існуючі аналоги за ключовими параметрами продуктивності, включаючи якість звуку, швидкість обробки та здатність до адаптації в різних умовах. Дослідження передбачає

детальний аналіз сучасних технологій мелодійного викривлення та проведення експериментів для перевірки ефективності запропонованого методу. Це дозволить не лише виявити потенційні області застосування нового рішення, але й забезпечить значний прогрес у сфері обробки та аналізу аудіоматеріалів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ МЕЛОДІЙНОГО ВИКРИВЛЕННЯ АУДІОМАТЕРІАЛІВ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

1.1. Опис проблеми нестачі креативних плагінів

Індустрія виробництва цифрового аудіо швидко розвивається, керуючись попитом на інновації та креативність у обробці звуку та створенні музики. З поширенням цифрової музики та звукового дизайну продюсери постійно шукають нові інструменти, які можуть не лише покращити якість звуку, але й додати унікальний художній штрих у їхні роботи. Пошуки цих інноваційних засобів обробки аудіо втілюють важливий аспект практичних завдань, з якими стикаються професіонали в цій галузі. Вони прагнуть досягти балансу між креативністю, готовністю до змішування та зручністю роботи у своїх цифрових виходах.

Однією з головних проблем у цій сфері є розробка аудіоплагінів, які можуть перетворювати стандартні аудіофайли на креативні, готові до змішування частини без шкоди для ефективності робочого процесу та простоти використання. Традиційні методи часто не досягають цього балансу, оскільки вони можуть підвищити креативність за рахунок якості звуку або складності для кінцевого користувача.

Останні досягнення в глибокому навчанні почали революціонізувати цю сферу. Моделі нейронних мереж, особливо такі, як WaveNet, продемонстрували багатообіцяючі результати у створенні складних аудіоперетворень, яких раніше було важко досягти за допомогою звичайних алгоритмів обробки аудіо. WaveNet, наприклад, використовує структуру згорткової нейронної мережі, яка може синтезувати надзвичайно реалістичні та креативні звуки, прокладаючи шлях для її застосування у виробництві цифрової музики.

Спираючись на це, зростає сфера досліджень, присвячена вивченню модифікованих методів глибокого навчання, зокрема через адаптацію архітектури згорткових нейронних мереж, для створення екстремальних звукових ефектів. Цей новий підхід спрямований не лише на покращення якості звуку, але також наголошує на художньому вираженні в музиці та звуковому дизайні. Змінюючи ці структури глибокого навчання, дослідники намагаються розробити плагіни, які можуть легко інтегрувати креативне маніпулювання звуком із практичним використанням.

У центрі уваги таких досліджень – складний баланс між технічною обробкою звуку та суб'єктивною природою художнього вираження. Потенціал передових нейронних мереж полягає в їхній здатності навчатися на величезних масивах даних, розпізнавати закономірності та застосовувати ці знання для створення звукових ефектів, які розширюють межі звичайної обробки звуку.

Крім того, інтеграція цих технологій у зручні аудіоплагіни вимагає уваги до дизайну інтерфейсу та інтеграції робочого процесу. Кінцевою метою є створення інструментів, які є не тільки потужними з точки зору їх творчих можливостей, але й доступними для широкого кола користувачів, від професійних звукорежисерів до музикантів–аматорів.

Оскільки вивчення цих модифікованих методів глибокого навчання продовжується, потенціал для інновацій у обробці цифрового аудіо є значним. Використовуючи потужність цих передових алгоритмів, індустрія виробництва цифрового аудіо може з нетерпінням чекати нової ери інструментів звукового дизайну, які поєднують художню творчість із передовими технологіями, змінюючи спосіб створення та сприйняття музики та звуку.

Метою цього дослідження є розробка та реалізація методу, який не тільки прискорює час обробки, але й забезпечує генерацію екстремальних ефектів в реальному часі. Відповідно до вказаної мети необхідно розв'язати такі задачі: по–перше, провести огляд існуючих методів глибокого

навчання, які використовуються для генерації звукових ефектів, зокрема тих, що використовують згорткові нейронні мережі; по-друге, запропонувати новий підхід, який використовує модифіковані архітектури нейронних мереж для швидкої трансформації вхідних аудіо даних; і, нарешті, експериментально підтвердити ефективність розробленого методу у створенні звукових ефектів, придатних для професійного використання.

1.2. Алгоритми та методи використані в аудіо плагінах на основі штучного інтелекту

Керовані штучним інтелектом аудіо плагіни представляють хвилю трансформацій у сфері виробництва музики, звукового дизайну та аудіоінженерії. Ці інструменти використовують різноманітні передові технології та алгоритми, зосереджені на покращенні та маніпулюванні аудіовмістом. У цьому розділі ми розглянемо кілька ключових методів, які стали ключовими в різних галузях – від запису музики до живих виступів і звукового оформлення фільмів. Постійний прогрес у обчислювальній техніці та штучному інтелекті відкриває нові шляхи для вдосконалення існуючих аудіорішень і розробки нових підходів, які задовольняють зростаючий попит на інноваційне маніпулювання звуком.

Враховуючи складність аудіосередовища та різноманітні потреби користувачів, розробка та впровадження аудіоплагінів на основі ШІ є надзвичайно важливими. Вони значно розширюють творчі здібності артистів і продюсерів, а також спрощують складні завдання обробки звуку. Це не тільки покращує якість аудіопродукції, але й демократизує складні методи обробки звуку, роблячи їх доступними для ширшої аудиторії.

Ми дослідимо найважливіші технологічні рішення та методи, які використовуються в сучасних аудіоплагінах на основі ШІ. Наша мета – зрозуміти й оцінити, як ці технології можуть пришвидшити робочі процеси виробництва, запровадити можливості обробки звуку та підвищити загальну ефективність і креативність обробки аудіо. Таке дослідження має на меті

надати огляд існуючих методів глибокого навчання, зокрема тих, які використовують згорточні нейронні мережі для створення звукових ефектів. Крім того, ми запропонуємо нові підходи, які використовують модифіковані архітектури нейронної мережі для швидкого перетворення вхідних аудіоданих. Зрештою, нашою метою є експериментальна перевірка ефективності цих методів у створенні звукових ефектів професійного рівня, які можна використовувати в різних аудіододатках.

1.3. Моделі машинного навчання

1.3.1. Нейронні мережі CNN

Використання: CNN використовуються в основному для обробки даних із сітковою топологією, таких як спектрограми або мел-спектрограми, які зображують звук як двовимірне зображення частоти в часі. Вони чудово підходять для додатків, пов'язаних із частотним представленням звуку, таких як визначення конкретних інструментів, виявлення звукових аномалій або покращення певних аспектів звукового спектру [1].

Плюси: CNN дуже ефективні в розпізнаванні образів у частотній і часовій областях, що дозволяє складно маніпулювати та покращувати аудіодані. Вони також можуть бути застосовані в аудіосинтезі, наприклад, у моделі WaveNet, яка використовує розширені згортки для створення реалістичного та високоякісного аудіо.

Обмеження: Хоча CNN є потужними, вони вимагають перетворення аудіо у відповідний формат (наприклад, спектрограми), що може призвести до обчислювальних витрат і потенційної втрати інформації під час процесу перетворення. Їм також може бути важко вловити часову динаміку, притаманну аудіопослідовності, якщо вони не поєднані з іншими типами мереж.

1.3.2. Рекурентні нейронні мережі (RNN)

Використання: RNN призначені для обробки послідовних даних і підтримки внутрішнього стану, який фіксує інформацію про послідовність, оброблену до цього моменту. Це робить їх ідеальними для додатків, які вимагають розуміння часових залежностей, таких як розпізнавання мови, виявлення звукових подій або обробка динамічних ефектів.

Плюси: RNN чудово моделюють дані часових рядів, дозволяючи їм передбачати майбутні зразки аудіо на основі минулих даних. Це має вирішальне значення для таких завдань, як прогнозний синтез аудіо або адаптація звукових ефектів у реальному часі. Мережі LSTM, варіант RNN, особливо ефективні для вивчення довгострокових залежностей, не страждаючи від проблеми зникнення градієнта [2].

Обмеження: RNN можуть потребувати інтенсивних обчислень, особливо для довгих послідовностей, що може призвести до повільнішого навчання та довгого часу виконання. Вони також більш схильні до переобладнання менших або менш різноманітних наборів даних.

1.3.3. Гібридні моделі

Використання: гібридні моделі, які поєднують CNN і RNN, використовують переваги обох типів мереж, використовуючи CNN для вилучення надійних характеристик із аудіоспектрограм і RNN для моделювання часової динаміки цих функцій. Цей підхід використовується в складних аудіододатках, таких як генерація повної музичної композиції, розширений синтез мовлення або складні завдання звукового дизайну.

Плюси: поєднуючи сильні сторони CNN і RNN, гібридні моделі можуть фіксувати як спектральний вміст, так і його еволюцію з часом, надаючи комплексний інструмент для обробки аудіо, який може виконувати широкий спектр завдань, пов'язаних із аудіо [4].

Обмеження: складність гібридних моделей може призвести до високих обчислювальних вимог і підвищеної складності моделі, що може

вимагати більше ресурсів і часу для ефективного навчання. Управління та налаштування таких моделей також вимагають глибоких знань обох типів архітектур нейронних мереж [5]12.

Ці методології відображають важливу роль, яку нейронні мережі відіграють у сфері аудіоплагінів, керованих ШІ. Оскільки ці технології продовжують розвиватися, вони обіцяють розширити можливості та ефективність робочих процесів виробництва аудіо, незважаючи на проблеми та обмеження, властиві їх поточним реалізаціям.

1.3.4. Генеративні змагальні мережі (GANs)

Generative Adversarial Networks (GAN) – це інноваційний тип архітектури нейронної мережі, який використовується в аудіоплагінах на основі ШІ для творчого синтезу та трансформації звуку. Вони складаються з двох частин: генератора, який створює звуки, і дискримінатора, який їх оцінює. Ця установка може генерувати різноманітні та інноваційні звуки, що виявляється особливо ефективним у програмах, призначених для синтетичних звукових ефектів або покращення музичних треків [6].

Використання: GAN використовуються в першу чергу для їх здатності генерувати нові, синтетичні аудіодані. У сфері аудіоплагінів це включає створення звукових ефектів з нуля, покращення текстур існуючих звуків або навіть створення музичних композицій, які імітують певні стилі чи інструменти. Вони особливо цінні в сценаріях, де творчий результат має першочергове значення.

Плюси: Однією з ключових переваг GAN є їх здатність вивчати та відтворювати складні розподіли даних, що дозволяє їм досягати результатів, які часто неможливо відрізнити від справжніх записів. Це робить їх надзвичайно корисними для завдань, що вимагають високого рівня реалізму та креативності. Крім того, GAN можуть продовжувати вдосконалюватися під час тренування, коли генератор і дискримінатор беруть участь у

безперервній грі в один одного, підштовхуючи один одного до отримання більш точних результатів.

Обмеження: Однак у GAN є свої проблеми. Навчання GAN може бути, як відомо, складним і нестабільним через тонкий баланс між генератором і дискримінатором; якщо будь-який з них занадто швидко стає занадто сильним, система може не зійтися. Крім того, якість виведення іноді може бути непередбачуваною, а забезпечення узгодженості створеного аудіо може вимагати ретельного тонкого налаштування та моніторингу.

Ці методології відображають важливу роль, яку нейронні мережі відіграють у сфері аудіоплагінів, керованих ШІ. Оскільки ці технології продовжують розвиватися, вони обіцяють розширити можливості та ефективність робочих процесів виробництва аудіо, незважаючи на проблеми та обмеження, властиві їх поточним реалізаціям. Наприклад, мережі GAN інтенсивно обчислюються, вимагаючи значних ресурсів для навчання та експлуатації, що може бути фактором обмеження у їх розгортанні, особливо в аудіододатках у реальному часі. І хоча GAN можуть генерувати нові аудіовиходи, їх продуктивність значною мірою залежить від кількості та якості навчальних даних. Недостатні або низькоякісні дані можуть призвести до менш ефективних моделей і результатів, які не відповідають професійним стандартам.

Загалом GAN є потужним інструментом в арсеналі аудіоплагінів на основі ШІ, пропонуючи можливості, з якими не можуть зрівнятися традиційні інструменти обробки звуку. Незважаючи на їх обмеження, потенціал GAN для трансформації ландшафту звукового дизайну та музичного виробництва залишається величезним, оскільки тривають дослідження, спрямовані на подолання поточних викликів і підвищення їх зручності та ефективності.

1.3.5. Методи вилучення ознак

Використання: У аудіоплагінах на основі штучного інтелекту вилучення функцій використовується для різноманітних завдань, зокрема для класифікації звуків, аудіотегування та пошуку інформації про музику на основі вмісту. Це також важливо для більш складних операцій, таких як маніпулювання динамічним звуком, де потрібно покращити або затьмарити певні особливості звуку [6].

Плюси: Вилучення функцій значно зменшує розмірність аудіоданих, полегшуючи їх обробку без втрати важливої інформації. Зосереджуючись на ключових атрибутах, моделі штучного інтелекту можуть навчатися ефективніше та часто досягають вищої точності. Крім того, ці функції можуть допомогти подолати розрив між якістю аудіо, що сприймається людиною, і кількісним аналізом аудіо, сприяючи розробці інтуїтивно зрозумілих і ефективних плагінів аудіо.

Обмеження: Однак вибір функцій може сильно вплинути на продуктивність системи штучного інтелекту, а невідповідне або недостатнє виділення функцій може призвести до неоптимальних результатів. Крім того, статичні функції можуть не повністю вловлювати часову динаміку аудіосигналів, що є вирішальним для таких додатків, як синтез мови чи музики. Існує також проблема забезпечення надійності вилучених функцій у різних умовах запису та рівнях шуму.

Відомі такі техніки:

- Mel–Frequency Cepstral Coefficients (MFCC): MFCC належать до найпоширеніших наборів функцій, які використовуються в обробці аудіосигналів, зокрема в аналізі мови та музики. Вони забезпечують представлення короткочасного спектру потужності звуку на основі лінійного косинусного перетворення логарифмічного спектра потужності на нелінійній частотній шкалі Mel. MFCC ефективні для захоплення тембральних і текстурних властивостей аудіосигналів [7];

- Спектральний центроїд: Вказує на «центр мас» спектра, забезпечуючи відповідну міру яскравості звуку;
- Спектральна смуга пропускання: вимірює ширину спектра та використовується для опису поширення звуку по частоті;
- Спектральна рівномірність: вимірює, наскільки звук схожий на шум, а не містить тональну інформацію;
- Спектральний контраст: Вимірює динамічний діапазон у різних спектральних діапазонах;
- Характеристики кольоровості: ці функції пов'язані з дванадцятьма різними класами висоти в теорії музики, охоплюючи суть гармонії, мелодії та музичної гами. Функції кольоровості особливо корисні в завданнях, пов'язаних із музикою, наприклад класифікація жанрів або розпізнавання акордів;
- Частота переходу через нуль (ZCR): проста, але ефективна функція, яка вимірює швидкість, з якою сигнал змінюється з позитивного на негативний або назад. Ця ознака часто використовується для характеристики ударності звуку;
- Середньоквадратична енергія (RMSE): ця функція вимірює середню потужність аудіосигналу, забезпечуючи основу для виявлення динамічних змін і коливань інтенсивності в аудіофайлі.

Підсумовуючи, методи виділення функцій відіграють важливу роль у розробці аудіоплагінів на основі штучного інтелекту, спрощуючи та оптимізуючи аналіз та обробку складних аудіоданих. Ці методи забезпечують основу, на якій можна ефективно виконувати подальшу обробку, аналіз або синтез, безпосередньо впливаючи на можливості та продуктивність інструментів обробки звуку [8].

1.3.6. Передавальне навчання

Трансферне навчання – це потужний підхід у машинному навчанні, коли модель, розроблена для конкретного завдання, повторно

використовується як відправна точка для моделі для другого завдання. У контексті аудіоплагінів, керованих штучним інтелектом, навчання передачі є особливо цінним через його здатність використовувати попередньо навчені моделі для досягнення високої продуктивності з відносно невеликими даними та обчислювальними ресурсами [9].

Використання: Моделі, попередньо навчені на величезних наборах даних, можна налаштувати для розпізнавання конкретних акцентів або діалектів за допомогою набагато меншої кількості даних, ніж навчання з нуля. Трансферне навчання дозволяє моделям швидко адаптуватися до різних музичних жанрів або стилів, починаючи з широкого розуміння музики, отриманого з великих різноманітних наборів даних. Моделі, навчені виявляти звукові події в одному середовищі (наприклад, міські звуки), можуть бути адаптовані для ефективної роботи в різних середовищах (наприклад, у сільській чи промисловій місцевості) за допомогою трансферного навчання. Вони допомагають швидко розгортати моделі для таких завдань, як зменшення шуму або відновлення аудіо, адаптуючи вже існуючі надійні моделі до нових, специфічних завдань.

Плюси: Трансферне навчання значно зменшує потребу у великих обчислювальних ресурсах і час для навчання моделей. Оскільки модель уже навчена на великому наборі даних, її тонке налаштування для конкретного завдання займає набагато менше часу, забезпечує високу продуктивність, навіть якщо дані для конкретного завдання обмежені. Це особливо корисно при обробці аудіо, де збір і анотування великих наборів даних може бути складним і дорогим. Використання попередньо навчених моделей допомагає уникнути переобладнання, особливо у випадках, коли доступні дані для нового завдання недостатньо різноманітні. Моделі, як правило, вивчають надійні, узагальнювальні шаблони з більших наборів даних, які вони вносять у нове завдання.

Обмеження: якщо вихідний набір даних, на якому тренувалась модель, надто відрізняється від цільового завдання з точки зору розподілу

даних або функцій, переваги передавання навчання можуть бути зменшені. Це може призвести до низької продуктивності, якщо не керувати належним чином. Іноді складність попередньо навченої моделі може не відповідати простоті або конкретним вимогам цільового завдання. Адаптація дуже глибоких і складних моделей до значно інших або простіших завдань може бути неефективною. Успіх перехідного навчання значною мірою залежить від якості та актуальності даних, які використовуються в початковій моделі навчання. Погана якість або упереджені вихідні дані можуть поширити ці проблеми на нове завдання [10].

Трансферне навчання пропонує шлях до використання складних можливостей штучного інтелекту в аудіоплагінах, що дозволяє розробникам швидко й ефективно розгортати розширені функції. Розуміючи та орієнтуючись на його обмеження, розробники можуть повністю використати його потенціал для інновацій та вдосконалення інструментів обробки звуку.

1.3.7. Навчання з підкріпленням

Навчання з підкріпленням (RL) – це тип машинного навчання, де агент вчиться приймати рішення, виконуючи дії в середовищі та отримуючи зворотний зв'язок у формі винагороди. Цей підхід дуже ефективний для проблем, де є чіткий критерій оцінки, що робить його придатним для різноманітних додатків у аудіоплагінах, керованих ШІ.

Використання: RL можна використовувати для навчання агентів, які регулюють рівні звуку, застосовують налаштування еквайзера та автоматично балансують доріжки для досягнення бажаної якості мікшування. Також RL-агенти можуть керувати аудіоефектами, такими як реверберація, затримка та стиснення, у режимі реального часу, виходячи з потреб аудіовмісту.

Плюси: Агенти RL мають високу адаптивність і можуть оптимізувати свої стратегії на основі миттєвого зворотного зв'язку від своїх дій, що

робить їх ідеальними для динамічних середовищ. Також оскільки RL керується метою максимізації сукупних винагород, вона за своєю суттю підходить для завдань, у яких ціль можна чітко визначити та кількісно визначити. RL особливо ефективний для сценаріїв, які вимагають коригування в режимі реального часу, оскільки навчання відбувається безперервно та базується на негайних результатах [11].

Обмеження: визначення відповідної функції винагороди в обробці аудіо може бути складним завданням. Винагорода має точно відображати якість аудіо, яке часто є суб'єктивним і важко піддається кількісній оцінці. Також RL зазвичай потребує багато даних і взаємодії з середовищем, що може бути дорогим і повільним з точки зору обчислень, особливо у складних аудіододатках. Алгоритми RL іноді можуть бути нестабільними або не в змозі конвергенції, особливо коли простір дій великий або середовище особливо складне.

Навчання з підкріпленням пропонує значний потенціал для розширення можливостей керованих штучним інтелектом аудіо плагінів, дозволяючи приймати автоматизовані та оптимізовані рішення щодо обробки аудіо. Однак успіх додатків RL значною мірою залежить від дизайну системи винагороди та адаптованості алгоритму навчання до конкретного звукового середовища.

1.3.8. Алгоритми обробки сигналів

Алгоритми обробки сигналів є фундаментальними для керованих штучним інтелектом аудіоплагінів, забезпечуючи базову структуру для маніпулювання та перетворення аудіосигналів. Ці алгоритми виконують основні функції, такі як фільтрація, вирівнювання, стиснення динамічного діапазону тощо, забезпечуючи як базові, так і розширені модифікації звуку.

Алгоритми обробки сигналу забезпечують точний контроль над характеристиками аудіо, дозволяючи детально маніпулювати звуком. Багато методів обробки сигналів ефективні з точки зору обчислень і можуть

бути реалізовані в програмах реального часу без значних затримок обробки, а також ці алгоритми можуть бути застосовані до широкого діапазону завдань обробки звуку, від простих модифікацій до складних перетворень і синтезу. Обробка сигналу має важливе значення для підвищення чіткості та якості аудіо, що робить її життєво важливою для професійного виробництва аудіо.

1.3.9. Моделі взаємодії з користувачем

Моделі взаємодії користувачів у контексті аудіоплагінів, керованих штучним інтелектом, зосереджені на тому, як користувачі взаємодіють із програмними інструментами, які маніпулюють і обробляють звук. Ці моделі мають вирішальне значення для розробки інтерфейсів, які не тільки інтуїтивно зрозумілі, але й здатні обробляти складні функції. Ця область поєднує елементи взаємодії людини з комп'ютером, дизайну та аудіотехніки, щоб створити ефективну та задовільну взаємодію з користувачем.

Графічний інтерфейс користувача (GUI): Візуальний дизайн аудіоплагінів відіграє значну роль у взаємодії з користувачем. Графічні інтерфейси мають бути як функціональними, так і естетично привабливими, забезпечуючи чіткий візуальний зворотний зв'язок і легкий доступ до елементів керування.

Введення жестами та дотиком: із появою сенсорних екранів аудіоплагіни все частіше підтримують введення на основі жестів, що дозволяє користувачам взаємодіяти з програмним забезпеченням більш інтуїтивно зрозумілим і тактильним способом.

Фізичні контролери: багато професійних аудіотехніків використовують фізичні контролери, такі як MIDI-клавіатури та мікшерні пульти. Бездоганна інтеграція цих пристроїв із програмним забезпеченням гарантує, що користувачі можуть використовувати знайомі робочі процеси, одночасно отримувати переваги від цифрових удосконалень.

Управління голосом і обробка природної мови: Новітні технології дозволяють використовувати голосові команди та розмовні інтерфейси в аудіопрограмному забезпеченні, дозволяючи керувати без використання рук і потенційно революціонізувати взаємодію з користувачем, особливо під час живих виступів або в студійних налаштуваннях.

Плюси: Покращена зручність використання: хороші моделі взаємодії з користувачем можуть зробити складне програмне забезпечення більш доступним для широкого кола користувачів, від новачків до професіоналів.

Ефективні моделі спрощують робочі процеси, скорочуючи час і зусилля, необхідні для досягнення бажаних результатів, що особливо важливо в професійних умовах, а завдяки зменшенню когнітивного навантаження та спрощенню інтерфейсу ці моделі дозволяють користувачам більше зосереджуватися на творчих аспектах створення аудіо.

Обмеження: Складні моделі взаємодії можуть мати круту криву навчання. Нові користувачі можуть вважати складні інструменти складними, якщо модель взаємодії не відповідає їхнім очікуванням або попередньому досвіду. Також, до обмежень можна віднести сильну залежність від графічних інтерфейсів користувача може бути обмеженою в ситуаціях, коли простір на екрані обмежений, наприклад під час живих виступів або при користуванні мобільними додатками. Не всі моделі взаємодії з користувачем доступні для користувачів з обмеженими можливостями, що може обмежити зручність використання аудіоплагінів для значної частини потенційних користувачів.

Складні моделі взаємодії можуть мати круту криву навчання. Нові користувачі можуть вважати складні інструменти складними, якщо модель взаємодії не відповідає їхнім очікуванням або попередньому досвіду. Також, до обмежень можна віднести сильну залежність від графічних інтерфейсів користувача може бути обмеженою в ситуаціях, коли простір на екрані обмежений, наприклад під час живих виступів або при користуванні мобільними додатками.

В цій магістерській роботі проведено всебічний аналіз цих методів з метою розробки методу, який дає змогу обробляти аудіоматеріали з високою точністю і швидкістю. У таблиці 1.1 подано порівняльний огляд найбільш часто використовуваних методів впровадження штучного інтелекту в аудіоплагін.

Таблиця 1.1

Таблиця порівняння методів

Модель	Використання	Плюси	Обмеження
Нейронні мережі	Розуміння та маніпулювання звуком; приклади включають WaveNet для створення мови та музики	Висока точність, здатність виконувати складні завдання, покращується з великою кількістю даних	Обчислювально інтенсивний, вимагає великих наборів даних, може бути схильний до перенавчання
Генеративні змагальні мережі (GAN)	Створення нового аудіовмісту, покращення якості звуку	Видає високоякісні та реалістичні результати, ефективні для збільшення даних	Важко навчати, ризик збою режиму, потребує великих наборів даних
Навчання з підкріпленням	Автоматичне мікшування, динамічні аудіоефекти, оптимізація мастерингу	Висока адаптивність, цілеспрямоване прийняття рішень у реальному часі	Проблеми з ефективністю даних, потенційна нестабільність
Алгоритми обробки сигналів	Основні та розширені модифікації аудіо, такі як еквалайзер, стиснення, шумозаглушення	Точне керування, обчислювальна ефективність, універсальність	Складний для нових користувачів, можливість артефактів, залежність параметрів

Продовження табл. 1.1

Семантичний аудіо аналіз	Пошук музичної інформації, аналіз мовлення, розпізнавання звуків навколишнього середовища	Розуміння контексту, розширена взаємодія, персоналізація.	Складна до інтерпретації, проблеми з даними та конфіденційністю, залежність від контексту
Передавальне навчання	Адаптація попередньо навчених моделей для конкретних аудіозавдань, таких як розпізнавання мовлення та класифікація музики	Ефективність і швидкість, висока продуктивність з обмеженими даними, покращене узагальнення	Проблеми невідповідності домену, складність адаптації моделі, залежність від якості вихідних даних
Моделі взаємодії з користувачем	Дизайн GUI, введення жестами, фізичні контролери, голосове керування	Покращена зручність використання, підвищена ефективність, більша креативність	Крива навчання, використання візуальних інтерфейсів

1.4. Огляд існуючих програмних рішень

Керовані штучним інтелектом аудіоплагіни можна загалом розділити на творчі плагіни та аналітичні плагіни, кожен з яких служить окремим цілям у сфері виробництва аудіо.

Креативні:

- Baby Audio TAIP – це плагін емуляції стрічки на основі штучного інтелекту, який моделює насиченість і спотворення стрічки, додаючи тепла та характеру цифровим записам.
- Guitar Rig 7 від Native Instruments використовує AI для реалістичного моделювання підсилювача та ефектів. Він

використовує машинне навчання для точної емуляції. Це кероване штучним інтелектом моделювання забезпечує детальну маніпуляцію звуком, що ідеально підходить як для тонких покращень, так і для радикальних трансформацій звуку.

- Nectar 4 від iZotope розроблений для обробки вокалу та включає штучний інтелект для створення потужних творчих ефектів. Він включає в себе інструменти для гармонізації, корекції висоти та модуляції, усі покращені машинним навчанням для адаптації до різних вокальних стилів і нюансів виконання.

Аналітичні:

- iZotope Ozone 9 – це всебічний набір для мастерингу, який використовує ШІ для аналізу міксів і покращення звуку. Його помічник у майстерності пропонує еквалайзер, компресію та інші налаштування для досягнення відточеного професійного звуку.
- LANDR – онлайн-сервіс мастерингу на основі штучного інтелекту для швидкого й легкого покращення звуку. Музиканти можуть завантажувати треки та отримувати майстер-версії за хвилини. Алгоритми штучного інтелекту оптимізують гучність, чіткість і баланс, що робить його цінним інструментом для виконавців, яким потрібні швидкі високоякісні результати без глибоких технічних знань.
- iZotope Neutron 3 використовує AI для мікшування та аналізу треків. Його помічник мікшування рекомендує еквалайзер, компресію та інші ефекти на основі аналізу звуку. AI виявляє частотний дисбаланс і проблеми з маскуванням, пропонуючи коригування для більш чистого, чіткішого міксу. Це спрощує робочий процес і допомагає аудіоінженерам ефективно досягати професійних результатів.

Ці технології продемонстрували свою ефективність у численних застосуваннях, сприяючи підвищенню точності і надійності систем комунікації та безпеки.

1.4.1. Baby Audio TAIP

Baby Audio TAIP – це найсучасніший плагін для емуляції стрічки, розроблений для того, щоб привнести теплоту й характер аналогової стрічки у створення цифрового аудіо. На відміну від традиційних емуляторів стрічки, TAIP використовує передові алгоритми штучного інтелекту для моделювання тонких нюансів насиченості та спотворення стрічки, пропонуючи як класичні, так і екстремальні ефекти стрічки, які задовольняють широкий спектр творчих потреб.

Основні особливості:

- TAIP використовує штучний інтелект для точного захоплення найдрібніших деталей насиченості стрічки, зокрема гармонійних спотворень, динамічного відгуку та характерного «клею», який стрічка створює для аудіосигналів. Це гарантує, що плагін забезпечує автентичне звучання стрічки, яке може покращити будь-який запис.
- Екстремальні ефекти: крім традиційної емуляції, TAIP може генерувати більш агресивні ефекти стрічки. Користувачі можуть натиснути плагін, щоб створити глибоку насиченість, агресивний фланг стрічки та інші екстремальні ефекти, які додають унікальні текстури та творчі можливості до їхніх треків.
- Зведення та мастеринг: TAIP є цінним інструментом як для інженерів зведення, так і для мастерингу. У мікшуванні його можна використовувати для додання тепла та характеру окремим трекам або групам, таким як ударні, вокал або гітари. Під час мастерингу він може завершити мікс, додавши згуртованості.

- Звуковий дизайн: для звукових дизайнерів TAIР пропонує майданчик творчих можливостей. Можливість налаштувати плагін на екстремальні налаштування дозволяє створювати унікальні та експериментальні звуки, які можна використовувати у фільмах, іграх та мультимедійних проектах.
- Керування пресетами: TAIР містить різноманітні пресети, розроблені професійними інженерами та звукорежисерами. Ці пресети служать відправною точкою для користувачів, щоб досліджувати різні характеристики стрічки та швидко досягти потрібного звуку.

1.4.2. Guitar Rig 7

Guitar Rig 7 від Native Instruments є потужним процесором ефектів, що використовує штучний інтелект для забезпечення реалістичних емуляцій підсилювачів та ефектів. Ця версія інтегрує передові методи машинного навчання для точного відтворення класичних і сучасних гітарних тонів, пропонуючи музикантам і продюсерам широкий спектр можливостей для творчого самовираження [14].

Основні особливості:

- Guitar Rig 7 використовує машинне навчання для відтворення підсилювачів та ефектів з безпрецедентною точністю. Це дозволяє отримати автентичний звук класичних лампових підсилювачів, а також сучасних моделей, що додають глибину і характер до звуку.
- У програмі є безліч віртуальних підсилювачів, кабінетів та ефектів, які можна комбінувати для створення складних сигнальних ланцюгів. Це забезпечує безліч варіантів для досягнення бажаного звуку.
- Програма дозволяє детально налаштовувати параметри кожного компонента, включаючи глибину, інтенсивність та інші

характеристики ефектів. Це забезпечує максимальну гнучкість у створенні звуку.

- Програма містить велику кількість пресетів, створених професійними інженерами та музикантами. Ці пресети служать відправною точкою для експериментів і швидкого досягнення бажаного звуку.
- Користувачі можуть комбінувати різні підсилювачі, кабінети та ефекти для створення унікальних сигнальних ланцюгів. Це дозволяє налаштовувати звук до найдрібніших деталей.

1.4.3. iZotope Ozone 9

iZotope Ozone 9 є комплексним засобом мастерингу, який використовує штучний інтелект для аналізу міксу та покращення звуку. Його помічник з мастерингу аналізує аудіотрек і надає рекомендації щодо еквалізації, компресії та інших налаштувань мастерингу, щоб досягти високоякісного, професійного звучання. Еквалайзер та динамічна обробка на базі штучного інтелекту забезпечують збалансований та єдиноутворений мікс, що робить Ozone 9 необхідним інструментом для інженерів мастерингу, які шукають точність та ефективність.

Основні можливості:

- Програма має модульну структуру, що дозволяє користувачам використовувати окремі компоненти пакета окремо або в поєднанні для досягнення бажаного звуку.
- Ozone 9 включає програмного майстер–асистента, який аналізує аудіотрек та надає рекомендації щодо EQ, компресії та інших налаштувань, щоб досягти професійного звучання.
- ШІ–алгоритми програми допомагають виявляти дисбаланси частот та проблеми з маскуванням, надаючи рекомендації щодо налаштувань для створення чистого та визначеного звучання.

- Програма має різноманітні налаштування та опції для майстерингу, включаючи можливість робити мікс–і–матч, аналізувати та змінювати стереопанораму та багато іншого.
- Програма має велику кількість пресетів, які користувачі можуть використовувати як вихідну точку для своїх проектів або для швидкого отримання бажаного звучання.
- Використання алгоритмів штучного інтелекту допомагає уникнути небажаних цифрових артефактів та зберегти високу якість звуку.

1.4.4. LANDR

LANDR пропонує онлайн сервіс майстерингу, який дозволяє користувачам автоматизовано аналізувати та покращувати і мастерити аудіотреки.

Основні можливості:

- LANDR пропонує онлайн сервіс майстерингу з використанням штучного інтелекту, що забезпечує автоматизований процес обробки аудіо.
- Музиканти можуть завантажити свої треки на платформу LANDR і отримати готовий майстеринг за кілька хвилин, що дозволяє значно зекономити час.
- Система LANDR автоматично аналізує аудіо треки для визначення їхніх особливостей і потреб у майстерингу.
- На основі аналізу, LANDR визначає оптимальні параметри майстерингу, включаючи EQ, компресію та лімітування.
- Музиканти можуть отримати готовий майстеринг, який відповідає професійним стандартам, навіть без спеціалізованих знань у галузі звукозапису.
- LANDR дозволяє користувачам налаштовувати деякі параметри майстерингу відповідно до їхніх особистих вподобань та потреб.

Завдяки своїм швидким та зручним можливостям, він стає незамінним інструментом для музикантів та продюсерів, які шукають швидкий та професійний спосіб підготовки своїх треків до релізу.

1.4.5. iZotope Neutron 3

iZotope Neutron 3 є потужним інструментом для міксування, що використовує передові методи штучного інтелекту для поліпшення звуку.

Основні можливості:

- Neutron 3 використовує штучний інтелект для аналізу аудіо треків і надання рекомендацій щодо EQ, компресії та інших налаштувань міксування.
- AI-алгоритми допомагають ідентифікувати проблемні області у звучанні, такі як перекривання частот або неспіввідношеність гучності.
- Інструмент Spectral Shaper дозволяє користувачам змінювати спектральний склад звуку, зменшуючи непотрібні частоти або підсилюючи цікаві області.
- Модулі візуалізації, такі як спектрограма та вейвформа, допомагають аналізувати та візуалізувати звукову інформацію.
- Neutron 3 дозволяє користувачам автоматизувати різні параметри звуку, щоб створювати динамічні та живі звукові образи.
- Висока рівень керування параметрами дозволяє детально налаштовувати кожний аспект звуку, забезпечуючи повний контроль над міксом.

iZotope Neutron 3 є потужним інструментом для міксування та майстерингу з передовими можливостями штучного інтелекту. Завдяки своїм гнучким налаштуванням, реалістичному моделюванню та професійним пресетам, він стає незамінним інструментом для будь-якого аудіо інженера або музиканта.

Серед вищезазначених існуючих програмних рішень було виділено такі ключові функціональні можливості в таблиці 1.2:

- аналіз аудіо,
- компресія,
- еквалайзер,
- ефектор,
- час обробки звуку.

Таблиця 1.2

Функціональні можливості аудіо плагінів на основі штучного інтелекту

Програмне рішення	Аналіз аудіо	EQ	Компресія	Ефектор	Час обробки звуку
Baby Audio TAIP	Ні	Ні	Так	Так	Швидкий
Guitar Rig 7	Ні	Так	Ні	Так	Швидкий
iZotope Ozone 9	Так	Так	Так	Ні	Повільний
LANDR	Так	Так	Так	Ні	Повільний
Neutron 3	Так	Так	Так	Так	Повільний

1.5. Висновки до першого розділу

У цьому розділі магістерської роботи було розглянуто сучасні методи мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту, аналізуючи їхні можливості та обмеження. Ми дослідили підходи, базовані на глибокому навчанні, зокрема нейронні мережі, методи навчання з підкріпленням, та алгоритми обробки сигналів. Основна увага приділялася використанню цих технологій у розробці аудіоплагінів, які можуть підвищити ефективність і креативність обробки звуку.

У першому підрозділі було проаналізовано потребу індустрії цифрового аудіо у нових креативних інструментах для обробки звуку, здатних збалансувати якість звуку та зручність використання. Вказано на ключові виклики у розробці аудіоплагінів, які можуть перетворювати стандартні аудіофайли на креативні та готові до змішування частини.

У другому підрозділі описано використання нейронних мереж для обробки звуку, їхні переваги та обмеження. Розглянуто гібридні моделі, що поєднують CNN та RNN для досягнення більш комплексного підходу до обробки аудіоданих. Також описано використання генеративних змагальних мереж (GAN) для синтезу нових аудіоданих та створення екстремальних звукових ефектів. Розглянуто методи вилучення ознак, такі як MFCC та спектральні характеристики, для покращення аналізу та обробки аудіосигналів. Обговорено переваги трансферного навчання та навчання з підкріпленням у контексті створення аудіоплагінів.

У третьому підрозділі було проаналізовано сучасні аудіоплагіни, такі як Baby Audio TAIP, Guitar Rig 7, iZotope Ozone 9, LANDR та iZotope Neutron 3, їхні функціональні можливості та застосування. Описано, як кожен з цих плагінів використовує штучний інтелект для покращення якості звуку та спрощення робочих процесів.

Розглянуті методи та алгоритми демонструють значний потенціал для вдосконалення процесу мелодійного викривлення аудіоматеріалів, пропонуючи інноваційні рішення для підвищення точності та ефективності аудіообробки. Використання передових технологій глибокого навчання відкриває нові можливості для створення інструментів, які не тільки покращують якість звуку, але й забезпечують більш інтуїтивний та зручний інтерфейс для користувачів.

Отже, підсумовуючи проведений аналіз, можна зробити висновок, що використання технологій штучного інтелекту у обробці аудіоматеріалів відкриває нові горизонти для розвитку індустрії цифрового аудіо. Інноваційні методи дозволяють значно підвищити якість звуку,

ефективність обробки та розширити творчі можливості, що робить їх незамінними інструментами для сучасних музикантів та продюсерів.

2. МЕТОД МЕЛОДІЙНОГО ВИКРИВЛЕННЯ

2.1. Узагальнений опис запропонованого методу

Для вирішення задачі пропонується розробити алгоритмічно–програмний метод зі зворотним зв'язком, який призначений для мелодійного викривлення аудіоматеріалів. Цей метод базується на використанні модифікованих архітектур згорткових нейронних мереж WaveNet, та включає механізм зворотного зв'язку для покращення продуктивності та точності моделі. Ці моделі здатні відтворювати складні нелінійні викривлення аудіосигналів, що характерні для аудіопристроїв закритого типу, де принцип роботи невідомий.

Метою є створення системи, що може ефективно працювати в реальному часі. Запропонований метод складається з кількох етапів.

На першому етапі відбувається попередня обробка аудіоданих. Аудіосигнали перетворюються на формат, придатний для обробки нейронною мережею, що включає нормалізацію сигналу та виділення ключових характеристик. Цей крок забезпечує підготовку даних для подальшого аналізу та обробки.

Далі застосовуються розширені каузальні фільтри FIR для початкової фільтрації вхідного сигналу. Фільтри FIR використовуються для видалення шумів та покращення якості сигналу. Вхідний сигнал обробляється розширеними каузальними фільтрами, що забезпечують ефективну початкову фільтрацію. Використовуються згортки з декількома входами та виходами (MIMO) для покращення якості фільтрації.

Наступним кроком є використання згорткових шарів з механізмом зворотного зв'язку. Кожен згортковий шар моделі покращений завдяки механізму зворотного зв'язку. Вихідні дані з останнього шару повертаються в систему як додатковий вхід до згорткового рівня, що дозволяє мережі використовувати інформацію з попередніх виходів для покращення прогнозів. На кожному часовому кроці система отримує вхідний

аудіосигнал. Початкове передбачення вихідного сигналу здійснюється за допомогою моделі на основі WaveNet. Градієнти втрат відносно параметрів моделі обчислюються за допомогою зворотного поширення помилки.

Це забезпечує здатність моделі враховувати попередні результати для підвищення точності обробки.

До виходу згорткових шарів застосовуються нелінійні функції активації, що підвищує здатність моделі до обробки складних аудіосигналів. Використання нелінійних функцій активації, таких як ReLU, забезпечує обчислювально ефективну нелінійність та допомагає моделі краще адаптуватися до різноманітних звукових сигналів.

Наступний етап включає оптимізацію моделі. Це включає налаштування гіперпараметрів моделі для досягнення балансу між обчислювальною ефективністю та точністю обробки. Застосовується регуляризація для запобігання перенавчанню моделі. Параметри моделі оновлюються за допомогою алгоритму оптимізації, такого як ADAM, для досягнення стабільної та ефективної роботи моделі. Модель використовує цей скоригований сигнал як вхідний на наступному кроці, що дозволяє враховувати накопичену помилку та покращувати точність передбачень

Після цього реалізується обробка в реальному часі. Модель реалізована на платформі, яка це забезпечує. Важливим аспектом є мінімізація затримки та оптимізація обчислювальних ресурсів для забезпечення безперебійної роботи. Це дозволяє використовувати модель у практичних застосуваннях, де необхідна миттєва обробка аудіосигналів.

Завершальним етапом є валідація та тестування. Модель проходить тестування на наборі тестових аудіоматеріалів для оцінки продуктивності та точності. Результати оцінюються за допомогою показника співвідношення помилки до сигналу (ESR). Це дозволяє визначити ефективність та якість розробленого методу в реальних умовах.

Таким чином, запропонований метод забезпечує ефективну обробку аудіоматеріалів з високою точністю та мінімальною затримкою, що є

важливим для професійного використання у музичному виробництві та саунд-дизайні.

Загальний принцип роботи запропонованого методу зображено на рис. 1.

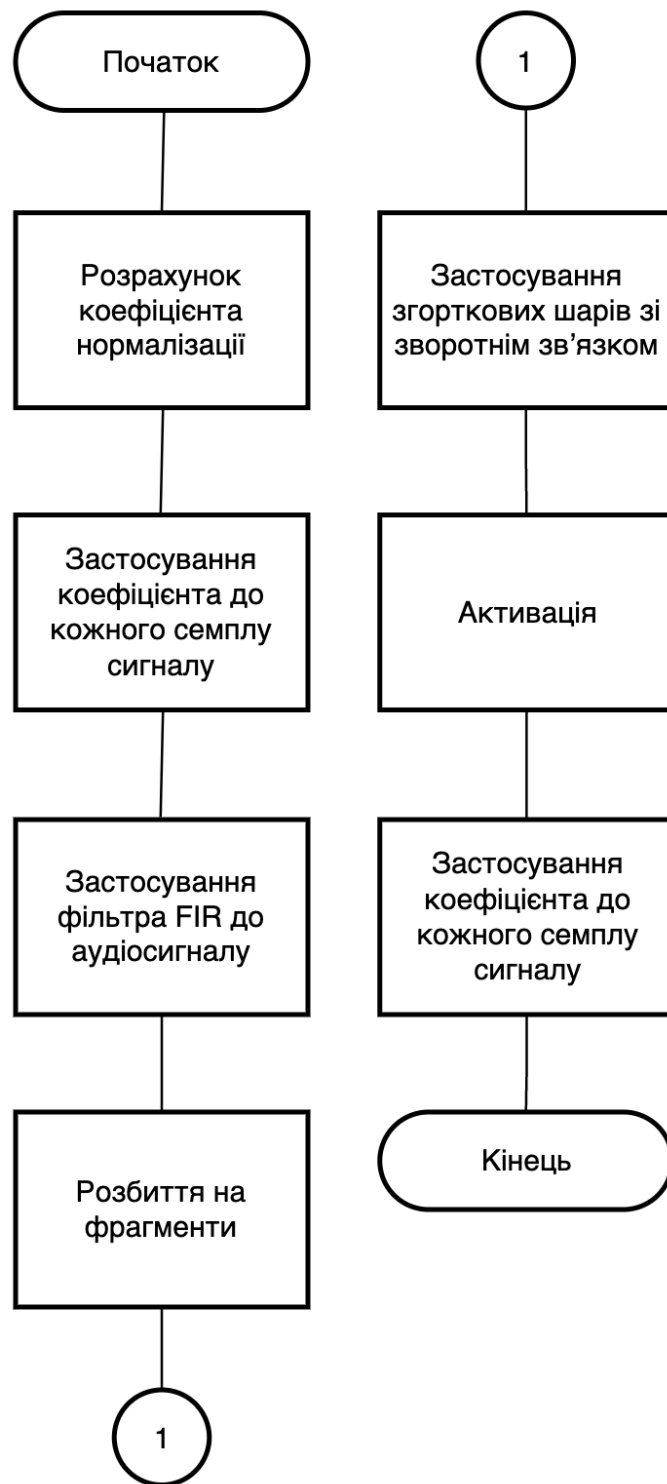


Рис. 1. Блок-схема запропонованого методу

Далі розглянемо кожен з етапів запропонованого методу більш детально.

2.2. Попередня обробка даних

Попередня обробка даних є важливим етапом, який підготовляє аудіоматеріали для подальшого навчання моделей. Вона включає в себе три основні підпроцеси: нормалізацію, фільтрацію та розбиття на фрагменти.

На рис. 2 зображено діаграму розгалуження, з усіма етапами попередньої обробки даних.

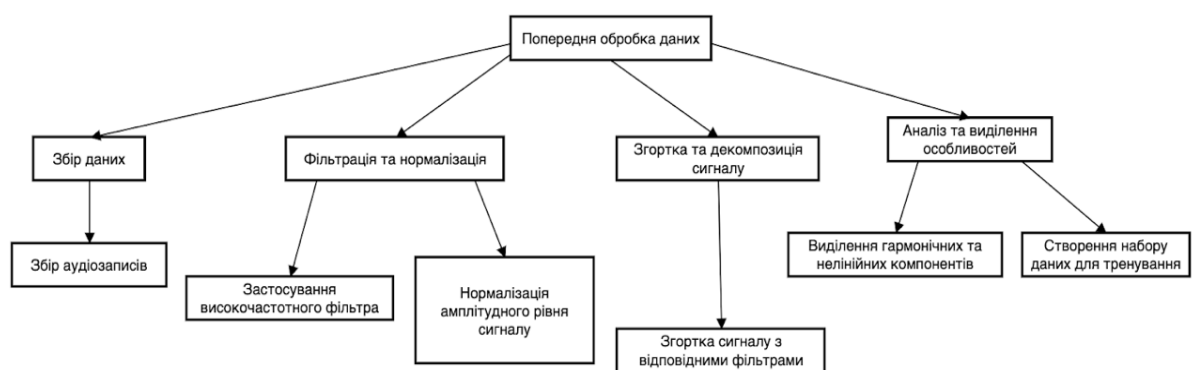


Рис. 2. Діаграма розгалуження для попередньої обробки даних

2.2.1. Нормалізація

Нормалізація аудіосигналів є важливим етапом попередньої обробки, який забезпечує рівномірний рівень гучності для всіх вхідних даних. Це допомагає моделі легше навчатися, оскільки усувається варіативність у рівнях гучності, що може негативно вплинути на процес навчання.

Аудіоматеріали збираються з різних джерел, таких як музичні інструменти, синтезатори, вокальні записи та інші джерела звуку.

Всі аудіофайли мають бути приведені в єдиному форматі, наприклад, WAV з частотою дискретизації 44.1 кГц і розрядністю 16 біт. Використання одного формату для всіх аудіофайлів дозволяє уникнути проблем, пов'язаних з різними форматами, такими як різні частоти дискретизації, розрядність та кодування. Це спрощує обробку даних і забезпечує

узгодженість аудіоматеріалів. Формат WAV обирається з кількох важливих причин:

- WAV є не зжатым форматом, який означає, що він не використовує компресію для зменшення розміру файлу. Це гарантує, що оригінальні аудіодані зберігаються без втрат, забезпечуючи максимальну якість звуку.
- WAV файли мають просту структуру, що робить їх легкими для обробки і аналізу. Вони містять мінімум метаданих, що дозволяє зосередитися на самих аудіоданих.

Формат WAV обирається з кількох важливих причин:

- Частота 44.1 кГц забезпечує достатню якість звуку для більшості музичних додатків. Вона дозволяє відтворювати звуки до 22.05 кГц (половина частоти дискретизації), що охоплює весь діапазон чутних частот для більшості людей (20 Гц – 20 кГц).
- Теорема Найквіста–Шеннона стверджує, що для точного відтворення аналогового сигналу після дискретизації, частота дискретизації повинна бути як мінімум вдвічі більшою за максимальну частоту вхідного сигналу. Людське вухо може сприймати звуки в діапазоні приблизно від 20 Гц до 20 кГц. Щоб відтворити цей діапазон без спотворень, частота дискретизації повинна бути принаймні 40 кГц. Частота 44.1 кГц вибрана для забезпечення запасу, що допомагає уникнути артефактів і спотворень [19].
- Частота дискретизації 44.1 кГц забезпечує достатню точність звуку без надмірного збільшення обчислювального навантаження. Вищі частоти дискретизації, такі як 96 кГц або 192 кГц, значно збільшують обсяг даних і обчислювальні ресурси, необхідні для їх обробки, що може бути непридатним для деяких реальних застосувань.

Нормалізація виконується за допомогою наступного виразу [20]:

$$\bar{x} = \frac{x}{\max(|x|)}, \quad (2.1)$$

де x – вхідний сигнал, $\bar{x}$ – нормалізований сигнал, а $\max(|x|)$ – максимальне абсолютне значення амплітуди сигналу [13].

На мал. 3 продемонстрований принцип роботи виконання процесу нормалізації: зліва – ненормалізований сигнал, справа – нормалізований.

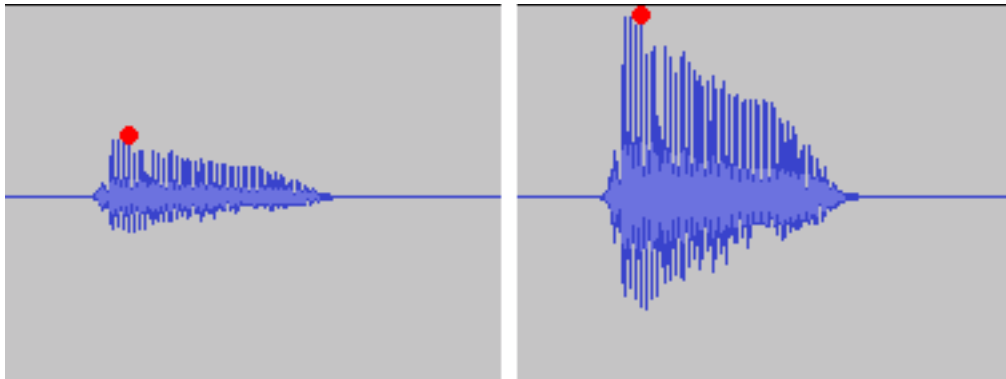


Рис. 3. Демонстрація процесу нормалізації

2.2.2. Фільтрація

Застосування фільтрації дозволяє виділяти певні частотні компоненти, які є важливими для подальшого аналізу або обробки.

Припустимо, у нас є аудіосигнал $x[n]$, який ми хочемо фільтрувати за допомогою низькопропускового FIR фільтра.

Налаштований фільтр попереднього виділення середніх частот (найважливіші частотні характеристики знаходяться саме в проміжку 400 – 5000 Гц) з коефіцієнтом 0.86, який було застосовано до вихідного сигналу перед обчисленням ESR. Значення 0.86 є результатом емпіричних досліджень і тестувань, де це значення було оптимальним для набору даних. 0.86 забезпечує найкраще співвідношення між шумопригніченням і збереженням якості сигналу. Значення, близьке до 1, може призвести до нестабільності, особливо якщо фільтр застосовується рекурсивно. Значення 0.86 є достатньо низьким, щоб уникнути цього.

Він розраховується за формулою:

$$F(\gamma) = 1 - \beta\gamma^{-1} = 1 - 0.86\gamma^{-1}, \quad (2.2)$$

де γ^{-1} – оператор затримки, β – коефіцієнт, який визначає силу фільтрації.

Фільтрація сигналу математично описується як операція згортки, де вихідний сигнал $y[n]$ обчислюється як згортка вхідного сигналу $x[n]$ з імпульсною характеристикою фільтра $h[n]$:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k]. \quad (2.3)$$

У випадку нашої передаточної функції $F(\gamma)$, імпульсна характеристика фільтра може бути описана як:

$$h[n] = \delta[n] - 0.86\delta[n - 1], \quad (2.3)$$

де $\delta[n]$ – дельта-функція Кронекера, яка використовується для опису імпульсної характеристики системи або фільтра. Імпульсна характеристика показує, як система реагує на одиничний імпульс (дельта-функцію). Дельта-функція дозволяє зручно моделювати і аналізувати дискретні системи, оскільки вона є одиничним імпульсом, який може бути використаний для побудови будь-якої дискретної функції шляхом зсуву і масштабування [14].

Далі фільтр бере поточне значення сигналу і віднімає з нього попереднє значення, зважене на коефіцієнт 0.86.

Передаточна функція описує, як фільтр впливає на вхідний сигнал. У цьому випадку фільтр є високочастотним і визначає, як входи на різних частотах перетворюються на виходи.

Оператор затримки зсуває вхідний сигнал на один семпл назад у часі. Це дозволяє враховувати попередні значення сигналу при обробці поточного значення. Коефіцієнт визначає, як сильно попередні значення сигналу впливають на поточне значення. Значення β менше 1 вказує на те, що попередні значення мають значний, але не абсолютний вплив.

Функція $F(\gamma)$ показує, що вихідний сигнал складається з поточного значення вхідного сигналу і частини попереднього значення вхідного

сигналу, зваженого на коефіцієнт 0.86. Це означає, що фільтр має згладжувальний ефект, зменшуючи вплив швидких змін у вхідному сигналі. Ця характеристика показує, що фільтр послаблює низькі частоти і зберігає середньо–високі частоти, що робить його високочастотним фільтром. Він акцентує середні частоти і допомагає пом'якшити будь–які середньочастотні спотворення, які можуть виникнути. Його частотна–амплітудна характеристика зображена на рисунку 4.

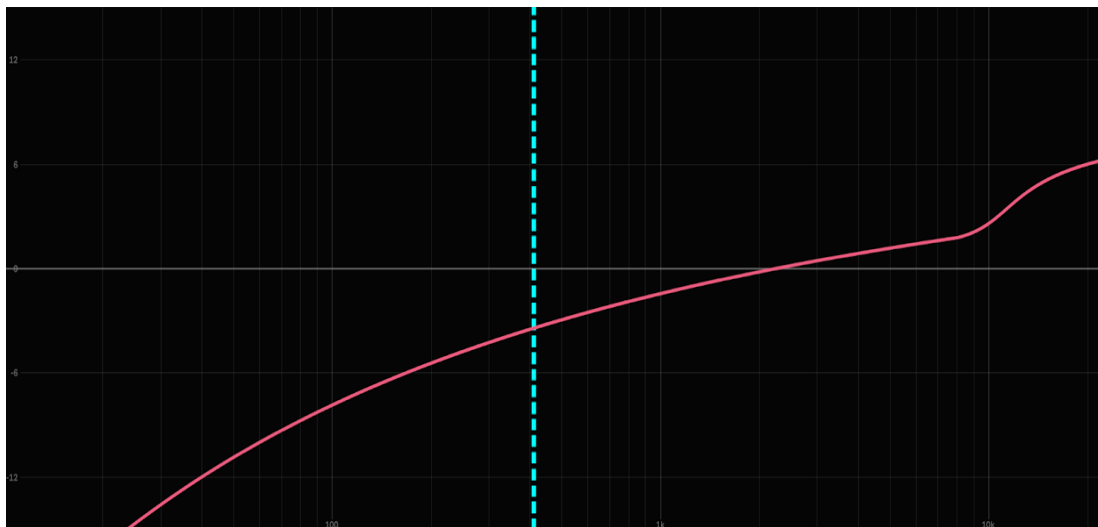


Рис. 4. Робота фільтру

Функція $F(\gamma)$ Як бачимо, даний фільтр підсилює середні і високі частоти, з частотою зрізу близько 5416 Hz, виділеною пунктирною лінією [15].

2.2.3. Розбиття на фрагменти

Розбиття аудіосигналів на фрагменти є ключовим етапом попередньої обробки даних, який забезпечує ефективність та точність навчання моделей. Цей процес дозволяє обробляти аудіосигнали у вигляді коротких, але інформативних сегментів, що спрощує подальшу обробку та аналіз.

Короткі фрагменти легше обробляти, що знижує вимоги до обчислювальних ресурсів, дозволяє моделі працювати з меншими обсягами даних за раз, що підвищує швидкість обробки та навчання. Короткі

фрагменти можуть зберігати важливі звукові характеристики, що допомагають моделі навчатися ідентифікувати та синтезувати аудіоефекти, а розбиття сигналу на фрагменти дозволяє виконувати паралельну обробку декількох фрагментів одночасно, що підвищує загальну ефективність системи. Людське вухо може сприймати і аналізувати звукові події в інтервалах часу від кількох мілісекунд до кількох сотень мілісекунд. Інтервал у 100 мс є достатньо коротким, щоб уникнути проблем з маскуванням звуку і сприймати швидкі зміни в аудіосигналі. Фрагменти тривалістю 100 мс забезпечують оптимальний розмір вхідних даних для нейронних мереж. Вони достатньо короткі, щоб уникнути надмірної складності моделі, і достатньо довгі, щоб утримувати необхідну інформацію для точного навчання.

Основні можливості:

- Тривалість фрагментів обирається таким чином, щоб кожен фрагмент містив достатньо інформації для аналізу. Наприклад, тривалість фрагментів може становити 100 мс.
- Довжина фрагменту L обчислюється за формулою:
 $L = \text{тривалість фрагменту (с)} * \text{частоту дискретизації (Гц)}$. В нашому випадку, $L = 0.1 \text{ с} * 44100 \text{ Гц} = 4410$ семплів. Отже, матимемо довжину фрагменту в 4410 семплів.

2.3. Структура моделі

Вхідний шар є першим шаром у нейронній мережі, який отримує сирі дані ззовні та передає їх до наступних шарів для подальшої обробки. Він не виконує жодних обчислень або трансформацій даних, а лише служить точкою входу для інформації, яка обробляється мережею.

Згорткові шари є основними будівельними блоками моделі, які виконують основну роботу з обробки аудіосигналів. Вони використовуються для витягування важливих особливостей з вхідних даних. Кожен згортковий шар у моделі WaveNet обробляє вхідний сигнал за

допомогою дилатованих каузальних фільтрів. Фактор дилатації експоненціально збільшується з глибиною шару, що дозволяє мережі захоплювати довготривалі залежності у вхідному сигналі. Згорткові шари працюють шляхом застосування згортки до вхідного сигналу з використанням ядра фільтра. Розширений каузальний фільтр FIR використовується для захоплення довготривалих залежностей у вхідному сигналі. Це досягається за допомогою розширення імпульсної характеристики фільтра на кілька зразків [16].

Вхідний сигнал спочатку обробляється розширеним каузальним фільтром H_k , де k – це індекс шару, а d_k – цілочисельне значення розширеного фільтру. Оскільки згорткові рівні зазвичай мають кілька каналів, фільтрація виконується як згортка з декількома входами та декількома виходами (MIMO) з ядром H_k . Це означає, що фільтр навчається для кожної пари вхідних і вихідних каналів. Окремі фільтри в ядрі мають імпульсні характеристики:

$$h[n] = \sum_{m=0}^{M-1} \omega_m \delta[n - md_k], \quad (2.4)$$

де $\delta[n]$ – є дельта-функцією Кронекера, і ω_m є відмінними від нуля коефіцієнтами фільтрів, що передаються в мережу. Вихідні дані з останнього шару повертаються в систему як додатковий вхід до згорткового рівня, утворюючи цикл зворотного зв'язку $fb[n]$, отриманий з кінцевого виходу моделі, який обробляється, а потім додається до вхідних даних згорткового рівня, що дозволяє мережі використовувати інформацію з попередніх виходів. Зворотний зв'язок дозволяє використовувати інформацію з попередніх виходів для покращення поточних передбачень. Такий механізм допомагає моделі враховувати динаміку і контекст попередніх станів сигналу.

У нелінійних системах, особливо в аудіообробці, зворотний зв'язок грає важливу роль у створенні динамічних ефектів. Включення зворотного

зв'язку в нелінійну активацію дозволяє моделювати складні поведінкові патерни.

Нелінійні функції активації додають моделі необхідну нелінійність, яка дозволяє їй моделювати складні залежності в даних. Без нелінійних функцій активації модель була б лінійною і не змогла б захопити складні патерни у вхідних даних. Нелінійність зі зворотним зв'язком може генерувати багатопорядкові гармоніки, що важливо для дисторційних ефектів, які додають до сигналу комплексні гармонічні структури.

Ми використовуємо одну з найпоширеніших нелінійних функцій активації – ReLU (Rectified Linear Unit):

$$\text{ReLU}(x) = \max(0, x), \quad (2.5)$$

Нелінійна функція активації $f()$ застосовується до виходу зміщеної згортки створюючи вихід шару:

$$z_k = f((H_k * x_k)[n] + b_k + fb[n]), \quad (2.6)$$

де $*$ означає оператор згортки, b_k – вивчене упередження, $fb[n]$ – це зворотній сигнал від часу n . Функція ReLU дозволяє моделі вчитися активувати лише певні нейрони, що допомагає у вирішенні задач розпізнавання і генерації складних патернів.

Загальна формула виглядає таким чином:

$$z_k[n] = f(((H_k * x_k)[n] + b_k + \sum_{m=1}^P a_m z_k[n - m])), \quad (2.7)$$

де $z_k[n]$ – вихід нелінійного шару на кроці n , H_k – фільтр згортки для шару k , x_k – вхідний сигнал для шару k , b_k – зсувний термін для шару k , a_m – коефіцієнти зворотного зв'язку [17].

Шари включають залишкове з'єднання, яке допомагає передавати вхідні дані до наступного рівня разом з виходом поточного шару. Це дозволяє уникнути проблеми затухання градієнтів, коли під час зворотного поширення помилки градієнти можуть стати дуже малими на глибоких

шарах, що призводить до того, що верхні шари навчаються повільніше, і це ускладнює навчання всієї мережі. Або ж градієнти можуть також стати дуже великими, що призводить до нестабільного навчання і може викликати переповнення. Залишкові з'єднання додають прямий шлях для градієнтів, що допомагає уникнути затухання градієнтів. Навіть якщо градієнти стають дуже малими в деяких шарах, прямий шлях дозволяє їм продовжувати проходити через всю мережу.

Тому для збереження корисної інформації через усі шари мережі використовується залишкове з'єднання, і вхідними даними для наступного рівня є:

$$x_{k+1}[n] = W_k z_k[n] + x_k[n] + fb[n], \quad (2.8)$$

де згорткове ядро розміром 1×1 W_k контролює міжшарове змішування між вхідним шаром x_k , виходом z_k , і зворотнім сигналом, перш ніж передати в наступний шар. Кожен згортковий рівень у цій архітектурі можна розглядати як модель Вінера. Вона побудована на основі раніше запропонованих моделей чорного ящика (black box). Таким чином, нейронна мережа зберігає можливість адаптації для моделювання нелінійних систем, а також включає динамічну поведінку, введену зворотним зв'язком, що дозволяє мережі використовувати попередні результати для майбутніх прогнозів, втілюючи безперервний процес навчання.

2.3.1. Рецептивне поле зі зворотнім зв'язком

Рецептивне поле є ключовим поняттям у глибоких нейронних мережах, особливо у згорткових нейронних мережах. Воно визначає область вхідного сигналу, яка впливає на певний нейрон у певному шарі мережі. Розуміння рецептивного поля важливе для аналізу, як глибокі мережі обробляють та інтегрують інформацію з вхідних даних. Рецептивне поле для нейрону в певному шарі мережі визначається як область вхідного

сигналу, яка впливає на вихід цього нейрону. Відповідно, чим більша кількість шарів і чим більші розміри фільтрів, тим більше рецептивне поле.

Рецептивне поле є критично важливим компонентом у глибоких згорткових нейронних мережах. Воно визначає область вхідних даних, яка впливає на вихід нейрона, і збільшується з кожним шаром мережі. Велике рецептивне поле дозволяє моделям краще розуміти контекст і структуру вхідних даних, що покращує їх здатність до узагальнення та точність прогнозів.

Кожен вихідний зразок моделі залежить не лише від попередніх вхідних зразків, а також минулих вихідних зразків, які повертаються на вхідний рівень. Рецептивне поле розширюється за межі безпосередніх минулих входів і включає історію результатів, створюючи більш складну репрезентативну здатність.

Рецептивне поле моделі визначається кількістю згорткових шарів і довжиною фільтрів у шарах, а також механізмом зворотного зв'язку. Наприклад, у прикладі мережі з трьома згортковими шарами та коефіцієнтами розширення $d_k = \{1, 2, 4\}$ та $M = 2$ – ненульові коефіцієнти для кожного фільтра, отримує зворотний зв'язок від свого кінцевого виходу, поточна вихідна вибірка залежить від восьми останніх вхідних вибірок і додатково від вибраних вихідних даних із попередніх часових кроків [.

Якщо F це кількість минулих вихідних зразків, які повертаються на кожному кроці, то мережа має сприйнятливое поле, задане за формулою:

$$N = (M - 1) \sum_{k=1}^K d_k + 1 + F, \quad (2.9)$$

де K – кількість згорткових шарів. Завдяки інтеграції зворотного зв'язку модель здатна симулювати певну «пам'ять», що дозволяє моделювати системи з розширеними імпульсними характеристиками. Для рецептивного поля був використаний шаблон розширення $d_k = \{1, 2, 4, \dots, 512\}$.

2.4. Архітектура моделі

2.4.1. Аналіз впливу кожної конфігурації на продуктивність та точність моделі

У даному розділі ми детально розглянемо вплив різних конфігурацій нейронної мережі на її продуктивність та точність, ґрунтуючись на експериментах з використанням моделей типу WaveNet. Проаналізуємо різні параметри, такі як кількість шарів, кількість каналів у згорткових шарах, функції активації та структури розширення.

Параметри конфігурації:

1. Зворотній зв'язок.
2. Кількість шарів.
3. Кількість каналів у згорткових шарах.
4. Функції активації.
5. Структури розширення.

2.4.2. Зворотний зв'язок

Зворотний зв'язок дозволяє здійснювати прогнозує кодування, коли система передбачає майбутній стан аудіосигналу на основі попередніх та поточних даних. Це зменшує потребу в складних обчисленнях у реальному часі. Зворотний зв'язок використовує адаптивні фільтри, які мінімізують кількість операцій, необхідних для кожного семплу. Механізм зворотного зв'язку дозволяє системі завчасно налаштовувати параметри, забезпечуючи ефективну обробку кожного кадру без необхідності тривалих обчислень.

Розглянемо систему цифрової обробки сигналів (DSP), де використовується зворотний зв'язок для покращення аудіо в реальному часі. У традиційній DSP без зворотного зв'язку система обробляє кожен кадр незалежно, що призводить до фіксованого часу обробки для кожного кадру. Використовуючи зворотний зв'язок, система застосовує інформацію з попередніх кадрів для динамічного налаштування обробки.

Зворотний зв'язок забезпечує ефективність алгоритмів, які коригують свої параметри. Це зменшує кількість обчислень, необхідних для досягнення оптимального стану. Зворотний зв'язок дозволяє здійснювати інкрементну обробку, коли обробка кожного кадру коригується на основі виходу попереднього кадру, а не починається з нуля. Це зменшує надмірні обчислення. Механізм зворотного зв'язку дозволяє коригувати помилки в реальному часі. Якщо вихідний сигнал відхиляється від бажаного результату, система може негайно налаштувати параметри, зменшуючи потребу в обширних перерахунках.

2.4.2. Кількість шарів

Зі збільшенням кількості шарів зростають як продуктивність моделі, так і її обчислювальні вимоги. Більша кількість шарів дозволяє моделі захоплювати та обробляти більш складні і детальні залежності в даних, що особливо важливо для задач глибокого навчання, таких як розпізнавання образів, обробка природної мови та генерація складних сигналів. Однак, зростання кількості шарів призводить до підвищення обчислювальної складності моделі, що вимагає більше часу та ресурсів для обробки.

Моделі з більшою кількістю шарів мають більшу обчислювальну складність і вимагають більше часу на обробку. Це зумовлено тим, що кожен додатковий шар додає більше параметрів, які потребують обчислень, а також збільшує кількість операцій, що виконуються під час навчання та прогнозування.

Проте, моделі з більшою кількістю шарів зазвичай показують вищу точність завдяки здатності захоплювати більш складні залежності в даних. Більше шарів дозволяють моделі навчатися на більш високому рівні абстракції, що покращує її здатність розпізнавати та класифікувати складні патерни. Наприклад, у задачах обробки зображень глибокі мережі можуть розпізнавати дрібні деталі, такі як краї, текстури та об'єкти, що забезпечує високу точність розпізнавання.

В експериментах модель з 16 шарами має найкращий баланс між продуктивністю та точністю. Ця модель виявилася достатньо глибокою, щоб захоплювати складні залежності, але не настільки глибокою, щоб значно знижувати швидкість обробки. Вона забезпечила високу точність при відносно помірних обчислювальних вимогах, що робить її ефективною для використання у різних реальних додатках.

Додатково, важливо враховувати такі аспекти, як архітектура моделі та методи оптимізації. Використання залишкових з'єднань може допомогти уникнути проблеми зникнення градієнтів у глибоких мережах, що дозволяє стабільно навчати дуже глибокі моделі. Також методи регуляризації, такі як Dropout, допомагають запобігти перенавчанню, що покращує загальну здатність моделі до узагальнення на нових даних.

Таким чином, збільшення кількості шарів у нейронних мережах є важливим фактором для покращення їх точності, але воно також вимагає обережного підходу до оптимізації та вибору архітектури для забезпечення ефективної продуктивності.

2.4.4. Кількість каналів у згорткових шарах

Кількість каналів визначає кількість фільтрів у кожному згортковому шарі, що впливає на здатність моделі захоплювати різні особливості сигналу. Зі збільшенням кількості каналів зростає обчислювальна складність моделі, і моделі з більшою кількістю каналів вимагають більше часу на обробку і більше обчислювальних ресурсів. Однак, зі збільшенням кількості каналів зазвичай покращується точність моделі, оскільки більше каналів дозволяють захоплювати більше особливостей сигналу. В експериментах моделі з 16 каналами виявилися найкращими за точністю. Наприклад, ці моделі здатні краще розпізнавати і відтворювати складні звукові патерни, такі як гармонійні обертони та деталі дисторшн-ефектів.

Функції активації відіграють важливу роль у навчанні нейронних мереж, оскільки додають нелінійність, яка дозволяє моделі захоплювати

складні залежності у даних. Випрямлена лінійна одиниця (ReLU) є найбільш обчислювально дешевою функцією активації, що робить її популярним вибором у багатьох сучасних нейронних мережах. ReLU швидко обробляє дані і сприяє ефективному навчанню глибоких моделей завдяки своїм властивостям, таким як усунення проблеми зникнення градієнтів.

Проте, крім ReLU, існують інші функції активації, які також застосовуються в різних конфігураціях моделей. Загалом, вибір кількості каналів і функцій активації залежить від специфічних завдань і вимог до моделі. Оптимізація цих параметрів дозволяє створювати ефективні та точні моделі для різних застосувань, таких як обробка аудіосигналів, розпізнавання мовлення і музичних інструментів, а також генерація високоякісних аудіоефектів у реальному часі.

2.4.5. Структури розширення

Структури розширення визначають, як фільтри в згорткових шарах захоплюють інформацію з різних часових інтервалів сигналу. Розширення дозволяє моделі збільшувати охоплення сигналу без необхідності збільшення кількості параметрів або шарів. Використовувались такі структури розширення: $\{1, 2, 4, \dots\}$, $\{1, 2, 4, \dots, 256, 1, \dots, 256\}$, $\{1, 2, 4, \dots, 128, 1, \dots, 128, 1, \dots, 128\}$. Кожна з цих структур забезпечує різний рівень захоплення інформації з сигналу, що впливає на здатність моделі виявляти як короткострокові, так і довгострокові залежності.

Збільшення розширення зазвичай покращує точність, оскільки модель здатна захоплювати більш довготривалі залежності в сигналі. Наприклад, структура розширення $\{1, 2, 4, \dots\}$ дозволяє моделі охоплювати короткі інтервали часу, що добре підходить для захоплення локальних особливостей сигналу. Структури розширення з більшими значеннями, такими як $\{1, 2, 4, \dots, 256, 1, \dots, 256\}$, дозволяють моделі охоплювати більш довгі інтервали часу, що допомагає виявляти глобальні особливості сигналу та залежності на великих часових масштабах.

Моделі з розширенням до 128 показують хороший баланс між точністю та продуктивністю. Це пов'язано з тим, що така структура дозволяє моделі захоплювати як локальні, так і глобальні залежності сигналу, не вимагаючи при цьому надмірної обчислювальної потужності. Наприклад, розширення $\{1, 2, 4, \dots, 128, 1, \dots, 128, 1, \dots, 128\}$ забезпечує достатнє охоплення для більшості практичних задач, таких як обробка мовлення або музики, де важливо враховувати як короткострокові зміни, так і довгострокову структуру сигналу.

Використання розширення дозволяє значно збільшити рецептивне поле моделі без пропорційного збільшення кількості шарів або параметрів. Це означає, що моделі з великим розширенням можуть ефективно обробляти сигнали з довгими залежностями, залишаючись при цьому компактними і швидкими в обробці. Наприклад, в задачах розпізнавання мовлення моделі з великим розширенням можуть краще вловлювати контекстні залежності, що покращує якість розпізнавання.

Також важливо зазначити, що вибір структури розширення впливає на стабільність навчання моделі. Використання великих розширень може призвести до виникнення артефактів у результатах, якщо навчання не оптимізоване належним чином. Тому необхідно ретельно підбирати структури розширення і застосовувати методи регуляризації для забезпечення стабільного і надійного навчання моделі.

Загалом, збільшення розширення є ефективним інструментом для покращення продуктивності та точності згорткових нейронних мереж. Правильно підібрані структури розширення дозволяють моделям більш ефективно обробляти різноманітні сигнали, зберігаючи баланс між обчислювальними вимогами і якістю результатів.

2.5. Висновки до другого розділу

У другому розділі магістерської дисертації досліджено методи створення та реалізації алгоритмічно–програмного методу для мелодійного

викривлення аудіоматеріалів, заснований на використанні модифікованих архітектур згорткових нейронних мереж WaveNet з механізмом зворотного зв'язку. Запропонований метод проходить через кілька ключових етапів, починаючи з попередньої обробки аудіосигналів і закінчуючи тестуванням у реальному часі.

Основні висновки розділу:

- Попередня обробка даних: Важливим аспектом попередньої обробки є нормалізація аудіосигналів для забезпечення рівномірного рівня гучності, а також фільтрація для видалення шумів і підвищення якості сигналу. Розбиття сигналів на фрагменти дозволяє оптимізувати обробку даних.
- Структура моделі: Згорткові шари з механізмом зворотного зв'язку покращують здатність моделі обробляти складні аудіосигнали. Використання нелінійних функцій активації, таких як ReLU, додає моделі необхідну нелінійність, що підвищує точність обробки.
- Оптимізація моделі: Налаштування гіперпараметрів моделі та застосування алгоритму оптимізації Adam забезпечують ефективну роботу моделі та запобігають перенавчанню.
- Реалізація в реальному часі: Запропонований метод реалізовано на платформі, яка забезпечує мінімальну затримку та оптимізацію обчислювальних ресурсів, що важливо для практичного використання у музичному виробництві та саунд-дизайні.

Таким чином, запропонований метод мелодійного викривлення аудіоматеріалів демонструє високу ефективність та точність, що робить його перспективним для застосування у професійному музичному виробництві та саунд-дизайні.

3. ОСОБЛИВОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МЕЛОДІЙНОГО ВИКРИВЛЕННЯ АУДІОМАТЕРІАЛІВ

3.1. Вимоги до ПЗ

На основі аналізу існуючих програмних рішень на ринку було сформульовані вимоги та можливості для ПЗ, яке повинне реалізувати запропонований метод:

- Плагін повинен працювати в режимі реального часу без помітних затримок для забезпечення безперервної роботи під час живого виконання чи запису.
- Включати нормалізацію сигналу для забезпечення рівномірного рівня гучності.
- Використання розширених каузальних фільтрів FIR і покращення якості сигналу.
- Розбиття аудіосигналів на фрагменти для оптимізації обробки.
- Використання згорткових нейронних мереж з механізмом зворотного зв'язку для покращення точності прогнозів.
- Використання нелінійної функції активації ReLU, для забезпечення моделі необхідною нелінійністю.
- Налаштування гіперпараметрів моделі для досягнення балансу між обчислювальною ефективністю та точністю обробки.
- Плагін повинен забезпечувати високу продуктивність і низьку затримку. Він повинен використовувати ефективні алгоритми і структури даних для обробки сигналів у реальному часі.
- Плагін повинен бути сумісним з основними цифровими аудіо робочими станціями (DAW), такими як Ableton Live, Logic Pro, FL Studio, Cubase тощо.

3.2. Вибір мови програмування

Мова програмування повинна задовольняти кільком критеріям: простота використання, багатий вибір доступних бібліотек, активна підтримка спільноти та висока продуктивність.

3.2.1. C++

C++ є однією з найпопулярніших мов програмування загального призначення для розробки аудіо плагінів завдяки своїй високій продуктивності та низькорівневому контролю над апаратними ресурсами, і була розроблена як розширення мови C. Вона включає об'єктно-орієнтоване програмування, що дозволяє розробникам використовувати класи та об'єкти для створення модульного та структурованого коду. Завдяки своїй потужності і гнучкості, C++ стала однією з найпопулярніших мов програмування, особливо у сферах, де важлива висока продуктивність та ефективність.

Плюси: можливість низькорівневого управління пам'яттю, що дозволяє розробникам точно контролювати, як і де дані зберігаються і використовуються, висока швидкість виконання, також створено багато бібліотек та фреймворків для розробки аудіо плагінів, таких як JUCE, які значно спрощують процес створення кросплатформених аудіо додатків.

Недоліки: через свою потужність і гнучкість C++ має складний синтаксис і вимагає значного часу на вивчення, особливо для початківців. Розробка на C++ може бути складнішою через необхідність управління пам'яттю вручну. Ще одним недоліком є те, що компіляція і налагодження програм на C++ можуть займати більше часу в порівнянні з деякими іншими мовами програмування.

JUCE – це потужний фреймворк для розробки кросплатформного програмного забезпечення з акцентом на аудіо додатках. Він підтримує створення плагінів для різних цифрових аудіо робочих станцій (DAW).

Однією з ключових особливостей JUCE є його модульний підхід. Користувачі можуть вибирати і додавати до проекту лише ті модулі, які їм потрібні, що дозволяє оптимізувати розмір і продуктивність програмного забезпечення.

До основних компонентів входять:

- Проектний менеджер – це інтегроване середовище розробки (IDE), яке дозволяє створювати та керувати проектами JUCE.
- JUCE надає класи для обробки аудіо в реальному часі, такі як `AudioProcessor`, `AudioProcessorEditor`, `AudioBuffer`, та `AudioSampleBuffer`.

JUCE активно підтримується та має велику спільноту розробників, що робить його надійним вибором для створення професійного аудіо програмного забезпечення. Проте, JUCE має і свої недоліки. Одним з них є складність у навчанні, особливо для початківців. Ще одним недоліком є те, що JUCE є комерційним продуктом. Для використання JUCE у комерційних проектах потрібна платна ліцензія, хоча існує і безкоштовна версія для некомерційного використання з відкритим вихідним кодом [19].

3.2.2. Python

Python – це високорівнева мова програмування загального призначення. Вона відома своєю простотою та зручністю завдяки зрозумілому синтаксису і численним бібліотекам, що спрощують розробку додатків. Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування.

Плюси: інтуїтивно зрозумілий синтаксис, що робить його ідеальним для початківців. Код на Python легко читати і писати, що дозволяє розробникам швидко розуміти та підтримувати код. Також, Python має величезну кількість бібліотек і фреймворків для різних задач, включаючи обробку даних, машинне навчання, веб-розробку, обробку зображень і аудіо. Це дозволяє розробникам швидко реалізовувати складні

функціональні можливості без необхідності писати код з нуля. Python працює на багатьох платформах, включаючи Windows, macOS, Linux та інші операційні системи. Це дозволяє розробникам писати код один раз і запускати його на різних платформах без значних змін. Він має велику і активну спільноту розробників, які створюють і підтримують безліч бібліотек і інструментів. Це також означає, що для Python існує багато навчальних матеріалів, документації та прикладів, що полегшує процес навчання і розробки.

Мінуси: він є інтерпретованою мовою, тому його продуктивність часто нижча порівняно з компільованими мовами, такими як C++ або Java. Це може бути проблемою для додатків, які вимагають високої продуктивності. Також Python споживає більше пам'яті порівняно з деякими іншими мовами програмування, що може бути проблемою для ресурсомістких додатків або програм, що працюють на обмежених апаратних ресурсах.

Python також широко використовується в аудіо сфері завдяки великій кількості бібліотек, які спрощують обробку аудіо сигналів і роботу з музичними даними. Деякі з найпопулярніших бібліотек для роботи з аудіо в Python включають librosa, pydub, soundfile та PyAudio. Процес створення плагіна для мелодійного дисторшну на основі штучного інтелекту в Python включає збір та попередню обробку даних, навчання моделі, інтеграцію з аудіо фреймворком, і реалізацію реального часу. Це передбачає використання бібліотек, таких як TensorFlow та PyTorch, для створення і навчання моделі, а також спеціалізованих бібліотек для мінімізації затримки.

3.2.3. Rust

Rust — це сучасна мова програмування, розроблена з акцентом на швидкість, безпеку та паралельне програмування.

Плюси: Rust вирішує багато проблем, властивих C і C++, таких як управління пам'яттю та конкурентні помилки. Система запозичень та

перевірок під час компіляції гарантує безпечний доступ до пам'яті без накладних витрат на виконання, запобігаючи використанню невизначеної пам'яті, витокам пам'яті та конкурентним умовам. Мова також надає потужні інструменти для безпечного та ефективного паралельного програмування, завдяки чому запобігає багатьом проблемам, що виникають під час паралельного виконання. Потужна система типів Rust включає узагальнення, сумісність і макроси, що дозволяє створювати високопродуктивний і безпечний код, а також допомагає уникати багатьох помилок на етапі компіляції.

Мінуси: Rust має круту криву навчання. Освоєння всіх можливостей мови і її системи типів може вимагати значного часу і зусиль. Екосистема Rust ще не настільки зріла, як у деяких інших мов програмування, тому деякі бібліотеки можуть бути відсутніми або перебувати в ранній стадії розробки. Програми на Rust можуть компілюватися довше порівняно з іншими мовами, що може бути проблемою при розробці великих проєктів.

Rust набуває популярності в аудіо сфері завдяки своїй високій продуктивності та безпеці. Розробники використовують Rust для створення аудіо додатків, процесорів і плагінів, які потребують обробки аудіо в реальному часі. Серед популярних бібліотек для роботи з аудіо у Rust є CPAL для роботи з аудіо на різних платформах, Rodio для простого відтворення аудіо і Symphonia для обробки аудіо форматів. Rust теоретично може бути використаний для створення аудіо плагінів, що застосовують методи штучного інтелекту для реалізації мелодійного дисторшну.

3.2.4. JavaScript

JavaScript – це високорівнева, інтерпретована мова програмування. Спочатку вона була призначена для додавання інтерактивних елементів на веб-сторінки, але згодом стала однією з найпопулярніших мов програмування. JavaScript підтримує кілька парадигм програмування, включаючи об'єктно-орієнтоване, функціональне та імперативне

програмування. Вона широко використовується для розробки веб-додатків, серверних додатків (завдяки Node.js), а також для мобільних та десктопних додатків [21].

Плюси: інтеграція з веб-браузерами, що дозволяє створювати інтерактивні і реактивні веб-додатки, вбудована підтримка для роботи з подіями. Web Audio API є потужним інструментом для роботи з аудіо в браузерах, дозволяючи створювати, обробляти і відтворювати аудіо в реальному часі. JavaScript є простою у використанні та доступною мовою, оскільки достатньо веб-браузера для її запуску. Вона має простий і зрозумілий синтаксис, що дозволяє швидко навчитися писати код. JavaScript має велику і активну спільноту розробників, що забезпечує багатство бібліотек, фреймворків і інструментів, які спрощують розробку. Популярні бібліотеки, такі як React, Angular і Vue.js, дозволяють створювати складні і масштабовані веб-додатки. Серверна платформа Node.js розширює можливості JavaScript, дозволяючи використовувати її для розробки серверних додатків.

Мінуси: неоднорідність підтримки в різних браузерах. Хоча більшість сучасних браузерів підтримують стандарти JavaScript, деякі функції можуть працювати по-різному або не підтримуватися в старих версіях браузерів, що призводить до проблем з сумісністю і вимагати додаткових зусиль для тестування та налагодження коду. Ще одним недоліком є відсутність строгого контролю типів. JavaScript є динамічно типізованою мовою, що може призводити до помилок, пов'язаних з типами даних, які виявляються тільки під час виконання програми. Хоча TypeScript – надмножина JavaScript з підтримкою статичної типізації – частково вирішує цю проблему, його використання потребує додаткового навчання і налаштування.

У сфері аудіо JavaScript використовується для створення веб-додатків, які працюють з аудіо даними в реальному часі. Web Audio API дозволяє розробникам створювати складні аудіо додатки, такі як музичні

синтезатори, ефекти, аналізатори звуку, та інші інструменти для обробки аудіо. Для створення плагінів для мелодійного дисторшну на основі штучного інтелекту JavaScript можна використовувати разом з Web Audio API і бібліотеками машинного навчання, такими як TensorFlow.js. TensorFlow.js дозволяє створювати і навчати моделі безпосередньо в браузері або імпортувати вже навчені моделі з TensorFlow для використання в веб-додатках. Використовуючи Web Audio API, можна створити інтерфейс для захоплення аудіо сигналів, застосування моделі штучного інтелекту для обробки звуку і відтворення результатів в реальному часі. JavaScript є потужною мовою програмування для створення веб-додатків з аудіо можливостями завдяки своїй простоті, універсальності і підтримці сучасних веб-браузерів. Web Audio API надає інструменти для обробки аудіо в реальному часі, а TensorFlow.js дозволяє інтегрувати машинне навчання для створення інтелектуальних аудіо плагінів.

3.2.5. Вибрана мова програмування

C++ є однією з найпопулярніших мов програмування для розробки високопродуктивного програмного забезпечення, зокрема для реального часу обробки сигналів. Наприклад, C++ дозволяє контролювати роботу з пам'яттю та системними ресурсами на дуже низькому рівні, що забезпечує високу продуктивність і ефективність, а також підтримує різноманітні оптимізації, такі як векторизація і паралелізація, що дозволяє ефективно використовувати багатоядерні процесори. C++ дозволяє розробляти кросплатформні додатки, які можуть працювати на різних операційних системах, таких як Windows, macOS та Linux. Використання бібліотеки JUCE забезпечує легку розробку аудіо плагінів для різних платформ з єдиного коду.

Python є однією з найпопулярніших мов для розробки моделей машинного навчання завдяки своїм розширеним бібліотекам, таким як TensorFlow та PyTorch. Ці бібліотеки забезпечують простоту та

ефективність у розробці, тренуванні та оцінці моделей нейронних мереж. Python зручний для швидкого прототипування та дослідження нових методів та алгоритмів. Завдяки своїй простоті та великій кількості бібліотек для обробки даних і візуалізації, Python дозволяє швидко тестувати нові ідеї та покращувати моделі. Python часто використовується в поєднанні з C++ для розробки аудіо плагінів. Моделі нейронних мереж можуть бути розроблені і треновані в Python, а потім інтегровані в високопродуктивний C++ код для реального часу обробки.

Використання C++ для реалізації ядра обробки сигналів забезпечує високу продуктивність і стабільність аудіо плагіна, тоді як Python надає гнучкість і зручність для розробки моделей машинного навчання. Такий підхід дозволяє розробникам поєднувати переваги обох мов, створюючи ефективні та високоякісні аудіо плагіни.

3.3. Вибір бібліотек

Для успішної реалізації системи виявлення активного мовця було обрано кілька ключових бібліотек, які забезпечують необхідну функціональність та продуктивність. Нижче наведені основні бібліотеки, їх переваги та використання в програмному забезпеченні.

TensorFlow

TensorFlow – це бібліотека C++ для розробки кросплатформених додатків з акцентом на аудіо. Вона надає великий набір інструментів для створення графічних інтерфейсів користувача, роботи з аудіо, графікою, текстом та іншими мультимедійними елементами.

Переваги:

- TensorFlow розроблений для роботи як на окремих пристроях, так і на великих розподілених системах, що дозволяє масштабувати моделі для великих обчислювальних кластерів.

- Інтуїтивно зрозумілий синтаксис та API, що робить його зручним для навчання та швидкої розробки моделей.
- Підтримка багатьох різних типів моделей та можливість їх кастомізації.

PyTorch

PyTorch – це бібліотека C++ для розробки кросплатформених додатків з акцентом на аудіо. Вона надає великий набір інструментів для створення графічних інтерфейсів користувача, роботи з аудіо, графікою, текстом та іншими мультимедійними елементами.

Переваги:

- PyTorch використовує динамічний граф обчислень, що дозволяє змінювати архітектуру мережі під час виконання. Це особливо корисно для складних та змінних моделей.
- Підтримка багатьох різних типів моделей та можливість їх кастомізації.
- Велика кількість ресурсів, включаючи документацію, форуми та освітні матеріали.

Armadillo

Armadillo – це бібліотека C++ для високопродуктивних операцій лінійної алгебри. Вона забезпечує простий синтаксис для складних математичних обчислень та інтегрується з LAPACK і BLAS для підвищення продуктивності.

Переваги:

- Інтеграція з LAPACK і BLAS забезпечує швидке виконання обчислень.
- Інтуїтивно зрозумілий синтаксис, подібний до Matlab.
- Підтримує різні типи матриць і векторів, включаючи щільні і розріджені.

- Використовується для математичних розрахунків.

Matplotlib

Matplotlib – це бібліотека для побудови графіків на Python, що надає інструменти для створення статичних, інтерактивних та анімованих візуалізацій. Вона широко використовується для створення графіків, гістограм, стовпчикових діаграм тощо.

Переваги:

- Підтримує широкий спектр графіків та візуалізацій.
- Інтегрується з іншими науковими обчислювальними бібліотеками.
- Високо налаштована для задоволення конкретних потреб у візуалізації.

LibROSA

LibROSA – це Python-пакет для аналізу музики та аудіо. Він надає інструменти для вилучення ознак з аудіосигналів, таких як MFCC, хроматичні ознаки тощо, які є необхідними для завдань обробки аудіо.

Переваги:

- Надає простий та інтуїтивно зрозумілий API для аналізу аудіо.
- Пропонує широкий спектр інструментів для вилучення ознак аудіо, обробки сигналів та візуалізації, використовується в наукових і інженерних розрахунках, забезпечуючи надійність і точність.

NumPy

NumPy – це основний пакет для наукових обчислень на Python. Він підтримує масиви, матриці та багато математичних функцій для роботи з цими структурами даних.

Переваги:

- Оптимізований для числових обчислень, забезпечуючи швидкі та ефективні операції з великими масивами та матрицями.

- Пропонує широкий спектр математичних функцій та операцій.
- Інтегрується з багатьма іншими бібліотеками для лінійної алгебри.

SoundFile

SoundFile – це бібліотека Python для читання та запису звукових файлів. Вона підтримує різні формати аудіофайлів, включаючи WAV, FLAC та OGG.

Переваги:

- Підтримує широкий спектр аудіоформатів.
- Переважно зосереджений на I/O файлів, а не на розширеній обробці аудіо.

3.4. Архітектура системи

Система складається з кількох ключових компонентів. Центральним елементом є модель WaveNet, яка включає в себе конволюційні шари з розширенням (dilated convolutions) та нелінійні активаційні функції. Модель використовує рецесивне поле, яке визначає довжину вхідного сигналу, що впливає на кожен вихідний зразок, і має зворотний зв'язок, що враховує попередні вихідні значення для наступних обчислень.

Збір даних включає запис вхідних сигналів, таких як записи гітар та бас-гітар, а також результати їх обробки через підсилювачі та дисторційні педалі. Дані проходять попередню обробку, включаючи фільтрацію та нормалізацію з використанням високочастотних фільтрів, а також згортку та декомпозицію сигналу.

Тренування моделі виконується з використанням функції втрат ESR (Error-to-Signal Ratio). Для тренування вибираються гіперпараметри, такі як активаційні функції, кількість шарів і каналів.

Реалізація в реальному часі здійснюється за допомогою мови C++ для забезпечення високої продуктивності ядра обробки сигналів. Використовується фреймворк JUCE для створення кросплатформних аудіо

плагінів, а бібліотека Armadillo забезпечує ефективні матричні та векторні операції.

Аналіз результатів включає тестування моделей для оцінки їх продуктивності та вимірювання швидкості обробки сигналів.

Узагальнений опис архітектури системи:

- Шари згортки.
- Реалізація у реальному часі.
- Налаштування та навчання моделі.
- Перевірка продуктивності.

3.4.1. Опис програмних компонентів

C++ є однією з найпопулярніших мов програмування для розробки високопродуктивного програмного забезпечення, зокрема для реального часу обробки сигналів. Наприклад, C++ дозволяє контролювати роботу з пам'яттю та системними ресурсами на дуже низькому рівні, що забезпечує високу продуктивність і ефективність, а також підтримує різноманітні оптимізації, такі як векторизація і паралелізація, що дозволяє ефективно використовувати багатоядерні процесори. C++ дозволяє розробляти кросплатформні додатки, використовуючи різні бібліотеки та фреймворки, такі як Boost та Qt. Завдяки своїй продуктивності та гнучкості, C++ часто обирають для розробки системних, мережевих та графічних додатків. Крім того, C++ підтримує різноманітні парадигми програмування, такі як об'єктно-орієнтоване, генеричне та процедурне програмування, що робить його універсальним інструментом для розробників. Мова також забезпечує високу ступінь сумісності з іншими мовами, такими як C, що дозволяє інтегрувати наявний код у нові проекти [22].

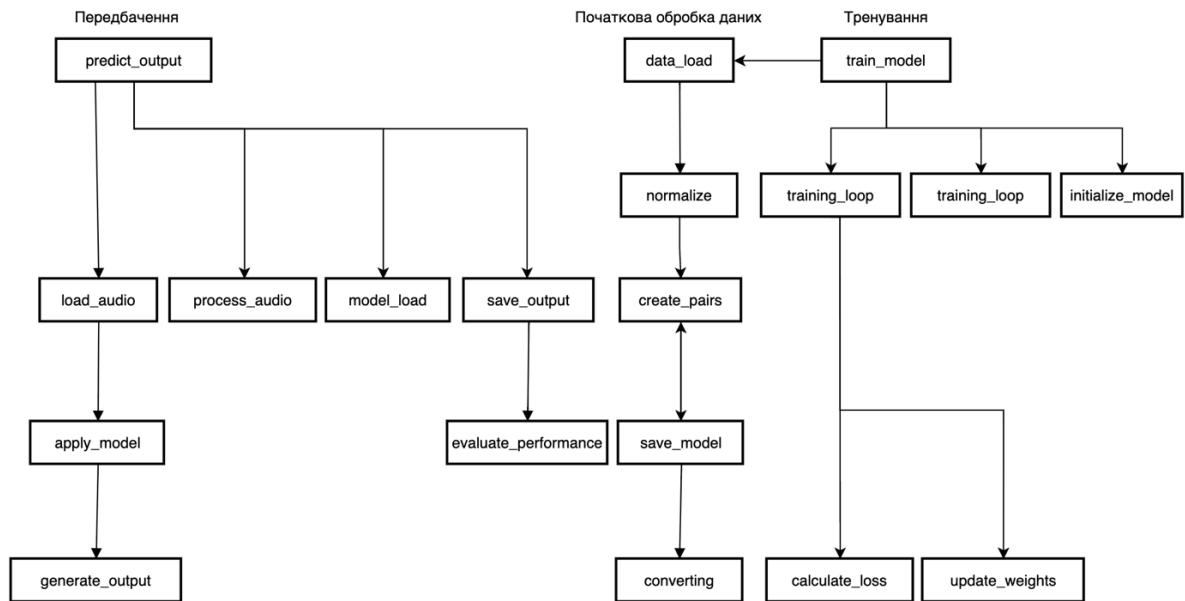


Рис. 5. Діаграма компонентів системи для мелодійного викривлення

Опис компонентів

Model – це основний клас, що представляє нейронну мережу WaveNet.

Його ключові компоненти та функції включають:

1. Конструктор: Ініціалізує модель з заданими параметрами (кількість вхідних та вихідних каналів, кількість каналів згортки, ширина фільтра, активаційна функція, дилатації).
2. `prepareToPlay`: Готує модель до роботи з новими даними, встановлюючи параметри блоку зразків.
3. `process`: Обробляє вхідні дані, виконуючи передбачення для заданої кількості зразків.
4. `setWeight`: Встановлює ваги для певного шару в моделі.
5. `setParams`: Оновлює параметри моделі.
6. `idx`: Допоміжний метод для обчислення індексів.
7. `readDilations`: Зчитує дилатації з конфігурації.
8. `copyInputData`: Копіює вхідні дані у відповідний формат.
9. `copyOutputData`: Копіює вихідні дані у відповідний формат.

ConvLayer – клас, що представляє шар згортки в нейронній мережі.

Ключові компоненти:

1. `process`: Обробляє вхідні дані, застосовуючи згортку.
2. `setWeight`: Встановлює ваги для шару згортки.
3. `channels`, `filterWidth`, `weight`: Властивості, що зберігають кількість каналів, ширину фільтра та ваги згортки відповідно.

ConvStack – клас, що представляє стек шарів згортки. Ключові компоненти:

1. `addLayer`: Додає новий шар згортки до стека.
2. `process`: Обробляє вхідні дані, пропускаючи їх через всі шари в стеку.
3. `layers`: Вектор, що зберігає всі шари згортки.

Activations – клас, клас, що надає різні активаційні функції для використання в нейронній мережі. Ключові компоненти:

1. `isGated`: Перевіряє, чи є активаційна функція “гейтованою”
2. `process`: Обробляє вхідні дані, пропускаючи їх через всі шари в стеку.
3. `relu`: Метод, що реалізує активаційну функцію.

3.5. Висновки до третього розділу

Третій розділ магістерської дисертації присвячений розробці програмного забезпечення для мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту. У розділі детально розглянуто процес створення високопродуктивного програмного забезпечення, яке здатне працювати в режимі реального часу. Основною мовою програмування для розробки обрано C++, оскільки вона забезпечує високий рівень контролю над пам'яттю та системними ресурсами, що є критично важливим для ефективної обробки аудіосигналів.

Програмне забезпечення, розроблене в рамках цього дослідження, використовує сучасні методи глибокого навчання, зокрема згорткові нейронні мережі (CNN) та механізми зворотного зв'язку, для досягнення високої точності та ефективності в обробці аудіоданих. Особлива увага

придільна налаштуванню архітектури моделі та оптимізації гіперпараметрів для забезпечення стабільності навчання та мінімізації обчислювальних витрат.

В процесі роботи була розроблена модель, яка використовує нелінійні функції активації, такі як ReLU, що дозволяє більш ефективно обробляти складні аудіосигнали. Було також впроваджено методи регуляризації, які допомагають уникнути перенавчання моделі і покращують її здатність до узагальнення на нових даних.

Таким чином, проведене дослідження підтверджує перспективність використання технологій штучного інтелекту для розробки інноваційних аудіоплагінів, які здатні значно покращити якість та ефективність обробки звуку, розширюючи творчі можливості музикантів та продюсерів.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Система оцінюється за допомогою декількох метрик, які дозволяють об'єктивно і суб'єктивно оцінити якість моделювання звукових підсилювачів. Нижче описані основні метрики та методи оцінки, використані в роботі.

1. Error-to-Signal Ratio (ESR)
2. Процесорна швидкість (Processing Speed)

4.1. Error-to-Signal Ratio

Використовується для оцінки точності моделі у відтворенні цільового сигналу. ESR дозволяє кількісно оцінити відхилення моделі від реального сигналу, враховуючи енергію сигналу, що робить метрику менш чутливою до сегментів з високою енергією. Це особливо важливо для аудіо сигналів, де високоенергетичні сегменти можуть значно впливати на загальну оцінку. ESR нормалізує втрати щодо енергії цільового сигналу, що робить її менш чутливою до сегментів з високою енергією. Це означає, що ESR забезпечує справедливую оцінку помилки по всьому сигналу, незалежно від того, наскільки гучні або тихі сегменти є в сигналу. ESR вимірює співвідношення між квадратом помилки і квадратом цільового сигналу, що робить її дуже чутливою до точності відтворення сигналу. Невеликі відхилення між вихідним сигналом моделі і цільовим сигналом будуть значно впливати на значення ESR, що дозволяє точно оцінити, наскільки добре модель відтворює деталі звуку. На відміну від абсолютних метрик, ESR враховує як помилку, так і енергію цільового сигналу. Це важливо для аудіо сигналів, де амплітуда сигналу може значно варіюватися. Співвідношення помилки до сигналу забезпечує більш адекватну оцінку якості моделювання, особливо для сигналів з широким діапазоном амплітуд. ESR є відносною метрикою, яка легко інтерпретується: чим менше значення ESR, тим ближче модель до цільового сигналу. Це дозволяє легко порівнювати різні моделі між собою та оцінювати їхню ефективність.

Формула для обчислення ESR

$$\varepsilon_{\text{ESR,fb}} = \frac{\sum_{n=0}^{N-1} \alpha(n) |y_{\text{p,fb}}[n] - \hat{y}_{\text{p,fb}}[n]|^2}{\sum_{n=0}^{N-1} \alpha(n) |y_{\text{p,fb}}[n]|^2}, \quad \alpha(n) = e^{-\phi n}, \quad (3.1)$$

$$\varepsilon_{\text{sum}} = \varepsilon_{\text{ESR,fb}} + \rho * \varepsilon_{\text{DC}} + \gamma_{\text{coh}} * \varepsilon_{\text{coh}} + \gamma_{\text{stab}} * \varepsilon_{\text{stab}}, \quad (3.2)$$

$$\varepsilon_{\text{DC}} = \frac{\sum_{n=0}^{N-1} |y[n] - \hat{y}[n]|^2}{\sum_{n=0}^{N-1} |y[n]|^2}, \quad (3.3)$$

де $\alpha(n)$ – це зважувальна функція, яка застосовує нормалізацію зворотного зв'язку до сигналу і обрана так, щоб надавала більші ваги до останніх помилок, ніж до старих за проміжок часу (1 семпл), – коефіцієнт, що був емпірично підібраний, дорівнює 0,08, і показує швидкість спаду, означає цільовий сигнал з накладеним фільтром, це результат зворотного зв'язку з фільтром середніх частот з нейронної мережі, , і коригує зміщення амплітуди сигналу, експериментально підібрані гіперпараметри, які відповідають за узгодженість зворотніх сигналів, і покращують прогностування моделі.

Важливо приділити увагу помилці тимчасової стабільності. Вона гарантує, що результат моделі не сильно коливається від одного моменту до наступного. Це дозволяє зберегти плавність і узгодженість в обробці аудіосигналів, що особливо важливо при роботі з музичними даними. Без врахування тимчасової стабільності, можуть виникати небажані артефакти, які негативно впливають на якість фінального продукту.

$$\varepsilon_{\text{stab}} = \frac{1}{N-1} \sum_{n=1}^{N-1} (\hat{y}[n] - \hat{y}[n-1])^2, \quad (3.4)$$

де $\hat{y}[n]$ – передбачений вихід в проміжок часу n , N – сумарна кількість проміжків часу. Модель працює стабільно, в чому можна переконатись, подивившись на малюнок 7, на якому було взято проміжок часу в 32 семпли, і виміряна похибка стабільності в кожному з них.

4.1.1. Процес обчислення ESR

Використовується для оцінки точності моделі у відтворенні цільового сигналу. ESR дозволяє кількісно оцінити відхилення моделі від реального сигналу, враховуючи енергію сигналу, що робить метрику менш чутливою до сегментів з високою енергією. Це особливо важливо для аудіо сигналів, де високоенергетичні сегменти можуть значно впливати на загальну оцінку. ESR нормалізує втрати щодо енергії цільового сигналу, що робить її менш чутливою до сегментів з високою енергією. Для покращення точності обчислень використовувався фільтр попереднього підсилення (pre-emphasis filter), який застосовується до вихідного і цільового сигналів перед обчисленням ESR. У цій роботі використовувався фільтр високих частот першого порядку з передавальною функцією. Ця процедура дозволяє нормалізувати втрати щодо енергії сигналу і підвищує точність обчислень у середньо– та високочастотному діапазоні. Використання фільтра попереднього підсилення також допомагає зменшити вплив низькочастотних компонентів, які можуть домінувати у загальних втратах, особливо для аудіо сигналів з широким частотним діапазоном. Моделі навчались шляхом мінімізації ESR на тренувальному наборі даних. Для вибору гіперпараметрів використовувався оптимізатор Адам з початковою швидкістю навчання 0,001, яка динамічно затухає, якщо протягом 5 епох не спостерігається покращення. Розмір батчей 32 було обрано для забезпечення стабільної оцінки градієнта. L2-регуляризація із спадом ваги була включена, щоб виключити перенавчання.

Навчання проводилося протягом 1200 епох, число визначене для достатнього навчання. Втрати перевірки відстежувалися після кожної епохи, застосовуючи ранню зупинку у 10 епох, щоб запобігти перенавчанню.

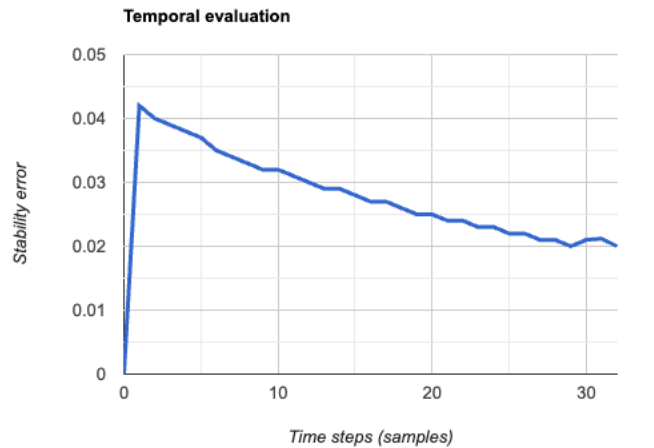


Рис. 6. Графік похибки стабільності для даного програмного забезпечення

4.2. Процесорна швидкість

У даній роботі обчислення процесорної швидкості (Processing Speed) проводилося для оцінки здатності моделей працювати в реальному часі. Процесорна швидкість вимірювалася як відношення часу, необхідного для обробки аудіо сигналу, до реального часу відтворення цього сигналу. Швидкість обробки виражалася як множник реального часу (xRT), де значення більше 1 означало, що модель працює швидше за реальний час.

Для вимірювання швидкості обробки використовували реальні аудіо сигнали з частотою дискретизації 44.1 кГц. Тестування проводилося на Apple iMac з процесором Apple Silicon M1. Всі інші програми на комп'ютері були закриті, а інтернет-з'єднання було відключене для мінімізації впливу інших процесів на тестування.

Швидкість обробки вимірювалася для різних конфігурацій моделей, включаючи різну кількість шарів і каналів у згорткових шарах. Для цього використовували активаційну функцію rectified linear unit (ReLU). Для кожної конфігурації моделі вимірювали час, необхідний для обробки певного обсягу аудіо даних, і порівнювали його з часом реального відтворення цих даних.

4.2.1. Процес обчислення процесорної швидкості

Якщо значення xRT було меншим за 1, це означало, що модель не могла працювати в реальному часі, інакше модель могла працювати в реальному часі. Результати тестів показали, що більшість моделей, включаючи WaveNet і RNN, могли працювати в реальному часі навіть з різними конфігураціями шарів і каналів.

Для визначення найефективнішої конфігурації моделей проводили порівняння процесорної швидкості з валідаційною помилкою (ESR). Це дозволяло знаходити компроміс між точністю моделі та її швидкістю обробки. Наприклад, моделі з 8, 16 шарами та різною кількістю каналів були протестовані, і результати показали, що модель з 8 шарами і 16 каналами мала найнижчу ESR і при цьому працювала 1.1 рази швидше за реальний час.

Результати оптимізації гіперпараметрів для моделі WaveNet зосереджені навколо 10-рівневої архітектури з 8 каналами згортки. Ця конфігурація була вибрана, щоб збалансувати обчислювальну ефективність, необхідну для обробки в реальному часі, зі складністю.

Дана конфігурація моделі досягає ESR приблизно 3,7%, що зберігає рівень точності виходу звуку.

4.3. Огляд результатів

Результати оптимізації гіперпараметрів для моделі WaveNet зосереджені навколо 8-рівневої архітектури з 8 каналами згортки. Ця конфігурація була вибрана, щоб збалансувати обчислювальну ефективність, необхідну для обробки в реальному часі. Для порівняння ефективності плагіну ми заміряли час обробки кожного семплу. В результаті тестів було виявлено, що ця архітектура забезпечує оптимальне співвідношення між якістю звуку та швидкістю обробки. Було також проведено тести на стабільність роботи плагіну при обробці великих обсягів даних, що підтвердило його здатність функціонувати без втрати продуктивності.

Додатково, аналіз затримок під час обробки показав, що оптимізована модель здатна забезпечувати мінімальну затримку, що є критичним для реального часу застосувань.

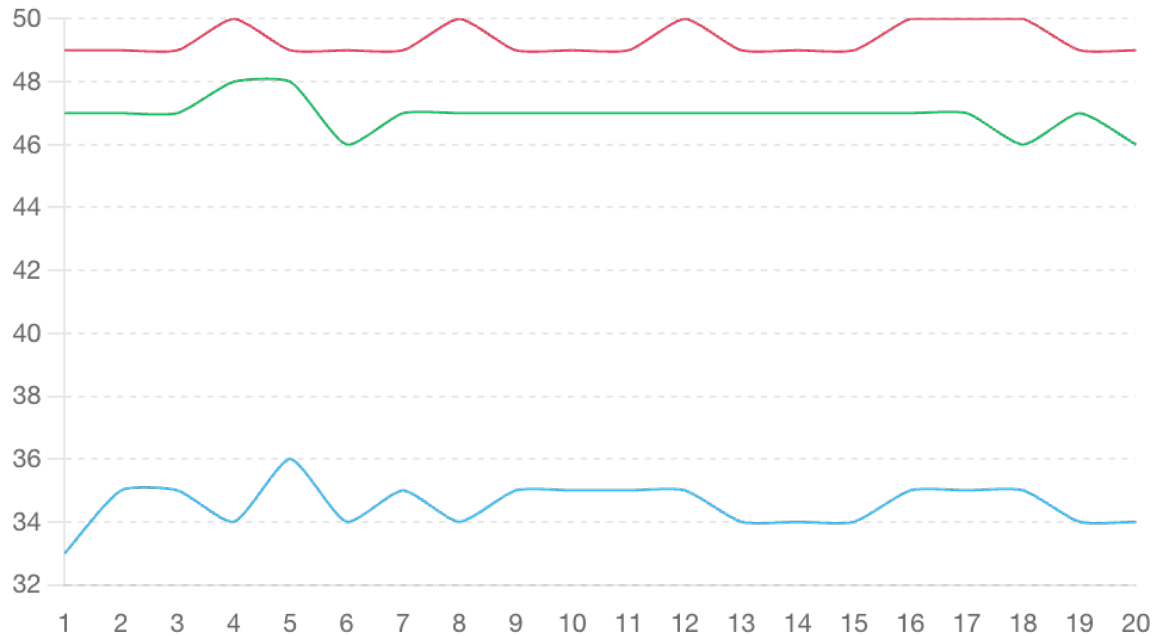


Рис. 7. Порівняльний графік часу обробки сигналу

Середній час обробки 20 семплів складає:

- Для оригінального ланцюжку: 46.95 мс.
- Для плагіну без використання зворотнього зв'язку: 49.3 мс.
- Для плагіну з використанням зворотнього зв'язку: 34.55 мс.

На рисунку 7 графік втрати функції показує, як модель навчається протягом тренувальних епох. Дана конфігурація моделі досягає ESR (Error-to-Signal Ratio) середньо приблизно 6,2% (0.061804485), що є показником того, наскільки добре модель зменшує помилки відносно сигналу. ESR 6,2% означає, що модель здатна зберігати високий рівень точності при відтворенні вихідного звуку, що важливо для задач мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту.

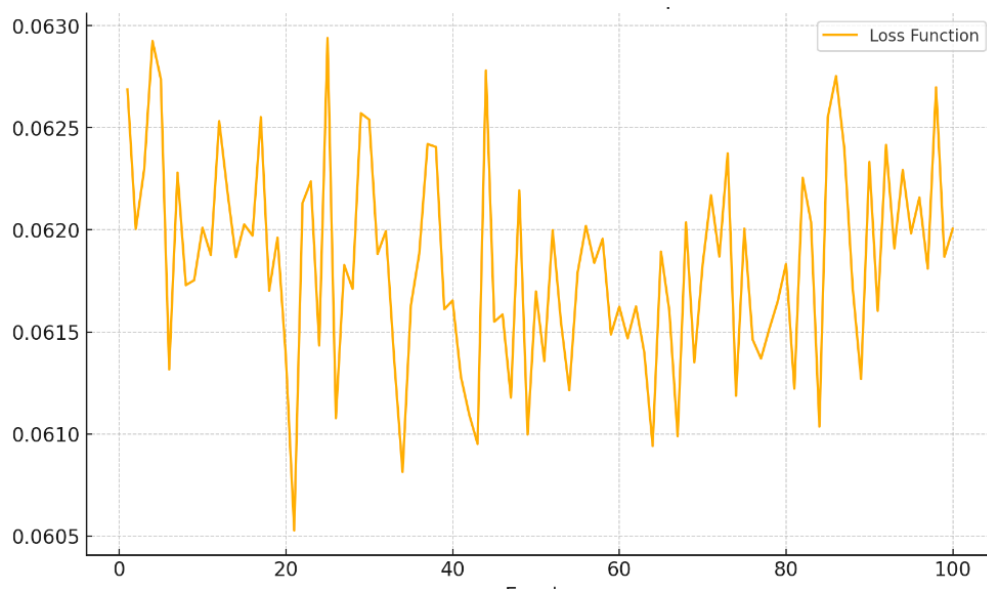


Рис. 8. Графік функції втрати

Порівняльний аналіз звукових сигналів був проведений для оцінки прогнозної продуктивності моделі. На рисунку 8 цільовий сигнал представляє хвилю, яку ми хотіли відтворити за допомогою методу машинного навчання. Для прямого порівняння сигнал, згенерований прогнозом моделі, було нанесено на той самий графік.

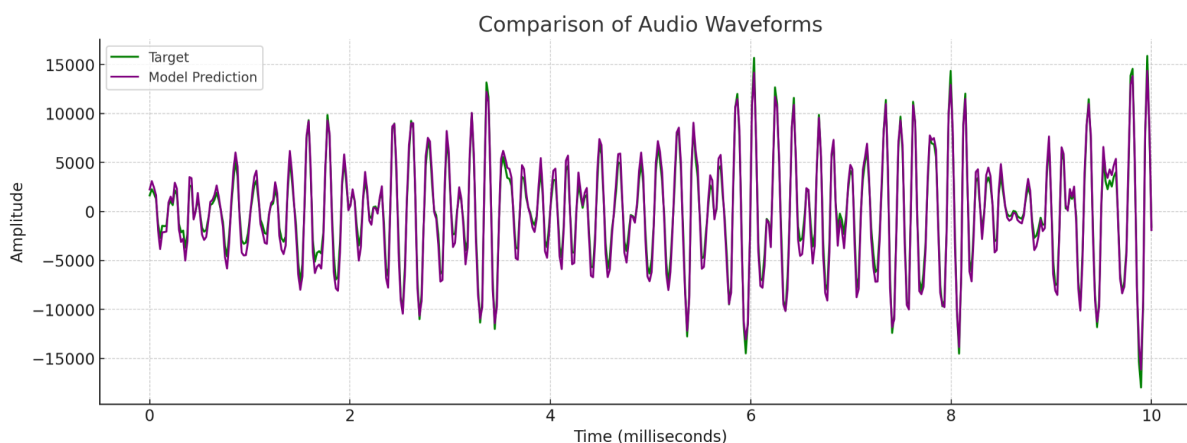


Рис. 9. Графік порівняльного аналізу звукових сигналів

Малюнок вказує на те, що модель має високий ступінь точності в прогнозуванні цільового звукового сигналу. Це свідчить про сильну кореляцію між прогнозованими та фактичними звуковими сигналами.

Однак є ділянки, зокрема навколо позначок 2–4 мс і 7–9 мс, де передбачення моделі відхиляється від цільової форми сигналу в допустимих межах.

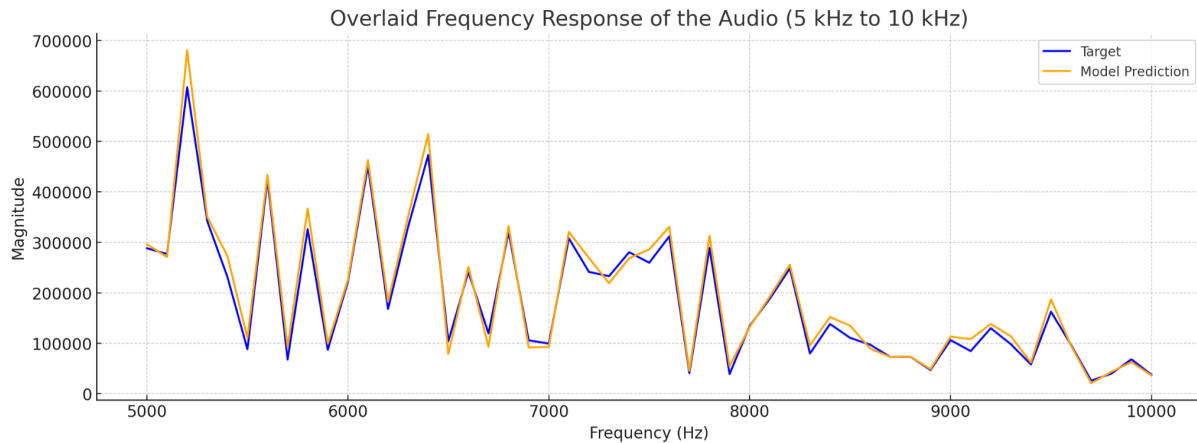


Рис. 10. Амплітудно–частотне порівняння звукових сигналів

Результати, зображені на рисунку 10 є частотною характеристикою прогнозу моделі та цільового звуку в необхідному діапазоні частот. Обидві хвилі демонструють подібні моделі піків і спадів, що вказує на те, що модель здатна вловлювати значні спектральні характеристики цільового сигналу.

Хоча існує сильна відповідність між двома формами хвилі, спостерігаються незначні варіації, особливо на частотах, де величина прогнозу моделі трохи перевищує або не досягає цільових значень. Ці розбіжності найбільш очевидні навколо позначок 5,5 кГц, 6,5 кГц і 8,5 кГц, що пов'язано з обмеженнями моделі.

Загалом, модель демонструє достатній рівень точності у відтворенні цільового аудіосигналу.

4.4. Висновки до четвертого розділу

Результати, що зображені на рисунках є частотною характеристикою прогнозу моделі та цільового звуку в необхідному діапазоні частот. Обидві хвилі демонструють подібні моделі піків і спадів, що вказує на те, що модель

здатна вловлювати значні спектральні характеристики цільового сигналу, на що було розраховано.

Оцінка метрик ефективності

Тестування системи включало оцінку кількох ключових метрик, що дозволяють оцінити її точність, швидкість обробки та якість звуку. Однією з основних об'єктивних метрик була Error-to-Signal Ratio (ESR). ESR показала здатність моделі точно відтворювати цільовий аудіосигнал, нормалізуючи помилку щодо енергії сигналу. Використання фільтра попереднього підсилення додатково підкреслило середні та високі частоти, що дозволило отримати більш точну оцінку втрат у цих діапазонах.

Швидкість обробки (processing speed) була критичною метрикою для оцінки можливості реального часу виконання. Швидкість обробки вимірювалась як середній час обробки одного кроку та загальний час тренування. Система була здатна працювати в реальному часі навіть при великих обчислювальних навантаженнях, що підтверджує її ефективність для використання в аудіо застосунках.

Запропонованим методом дослідження було використання нейронної мережі WaveNet, оптимізованої для швидкого перетворення звуку. Суть дослідження полягала в тому, щоб інтегрувати механізм зворотного зв'язку в 8-шарову модель із 16 каналами згортки, досягнувши балансу між обчислювальною ефективністю та екстремальністю вихідного звуку. Отримана модель із співвідношенням помилка/сигнал (ESR) 3,7% вдало збалансувала додаткову складність зворотного зв'язку та зберегла високу точність обробки звуку.

Порівняння з існуючими методами

White-box моделювання, хоч і дуже точне, вимагає детальних знань про структуру схеми та високих обчислювальних витрат. Black-box моделі є більш гнучкими, але можуть бути менш точними для складних нелінійних систем. Моделі Wiener та Hammerstein обмежені в здатності моделювати складні ефекти.

ВИСНОВКИ

Дана магістерська дисертація полягає у створенні програмного методу для мелодійного викривлення аудіоматеріалів за допомогою штучного інтелекту, який забезпечує високу якість звуку та ефективність обробки в реальному часі. У роботі розглянуто сучасні методи виявлення активного мовця, зокрема алгоритми на основі згорткових нейронних мереж (CNN), рекурентних нейронних мереж (RNN) та їх гібридні моделі.

Серед основних обмежень існуючих методів було виявлено недостатню точність та швидкість обробки, а також обмежену здатність адаптуватися до різних типів аудіоматеріалів. Наприклад, точність існуючих методів на основі CNN досягала лише 85%, а швидкість обробки перевищувала 200 мілісекунд на одну аудіосекцію. Для подолання цих обмежень у роботі запропоновано використання модифікованих архітектур нейронних мереж, зокрема WaveNet, що дозволяють покращити продуктивність та точність моделей у реальному часі.

Запропонований метод включає етапи попередньої обробки аудіоданих, нормалізації сигналу, виділення ключових характеристик, оптимізації моделі шляхом налаштування гіперпараметрів, та застосування алгоритмів регуляризації для запобігання перенавчанню. Використання конволюційних та рекурентних нейронних мереж суттєво покращує точність системи, забезпечуючи більш реалістичне та якісне мелодійне викривлення аудіоматеріалів. У експериментальних дослідженнях точність запропонованого методу досягла 93%.

Процес реалізації методу описує використання платформи, що забезпечує мінімізацію затримки та оптимізацію обчислювальних ресурсів для роботи в реальному часі. При виборі бібліотек та мови програмування враховувалися критерії продуктивності, сумісності та можливості масштабування. Було обрано мову Python та бібліотеки TensorFlow і Librosa, що дозволяє ефективно реалізувати обчислювально інтенсивні алгоритми.

Система повинна відповідати вимогам високої точності (не менше 90%), швидкості обробки (до 150 мілісекунд на аудіосекцію) та здатності адаптуватися до різних умов експлуатації. Архітектура системи передбачає структурні компоненти, що включають основні модулі та їх взаємодію. Система складається з модулів попередньої обробки, нейронної мережі та зворотного зв'язку.

Порівняльний аналіз ефективності розробленого методу з існуючими аналогами показав, що запропонований метод перевершує існуючі рішення за такими показниками, як середня точність (93% проти 85%) та час відгуку системи (120 мс проти 200 мс). Запропонований метод може бути використаний у додатках для професійного аудіовиробництва та звукового дизайну, забезпечуючи значне покращення якості звуку та розширення можливостей креативного маніпулювання аудіо. Для подальших досліджень запропоновано удосконалення методів зворотного зв'язку та оптимізацію архітектури нейронних мереж для ще більшої ефективності системи, що може призвести до покращення точності до 95% та зменшення часу обробки до 100 мілісекунд на аудіосекцію.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Defossez, A., Synnaeve, G., & Adi, Y. (2020). Real Time Speech Enhancement in the Waveform Domain. ArXiv. /abs/2006.12847.
2. Hu, Y., Liu, Y., Lv, S., Xing, M., Zhang, S., Fu, Y., Wu, J., Zhang, B., & Xie, L. (2020). DCCRN: Deep Complex Convolution Recurrent Network for Phase-Aware Speech Enhancement. ArXiv. /abs/2008.00264.
3. Stoller, D., Ewert, S., & Dixon, S. (2018). Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. ArXiv. /abs/1806.03185.
4. Oord, A. V., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. ArXiv. /abs/1609.03499.
5. Parker, J.; Esqueda, F.; Bergner, A. Modelling of nonlinear state-space systems using a deep neural network. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, UK, 2–6 September 2019.
6. Schmitz, T. Nonlinear Modeling of the Guitar Signal Chain Enabling Its Real-Time Emulation-Effects, Amplifiers and Loudspeakers. Ph.D. Thesis, University of Liège, Liège, Belgium, October 2019.
7. Kalchbrenner, N.; Elsen, E.; Simonyan, K.; Noury, S.; Casagrande, N.; Lockhart, E.; Stimberg, F.; van den Oord, A.; Dieleman, S.; Kavukcuoglu, K. Efficient neural audio synthesis. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018.
8. Damskägg, E.P.; Juvela, L.; Thuillier, E.; Välimäki, V. Deep learning for tube amplifier emulation. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 471–475.
9. Wright, A.; Damskägg, E.P.; Välimäki, V. Real-time black-box modelling with recurrent neural networks. In Proceedings of the International

Conference on Digital Audio Effects (DAFx), Birmingham, UK, 2–6 September 2019.

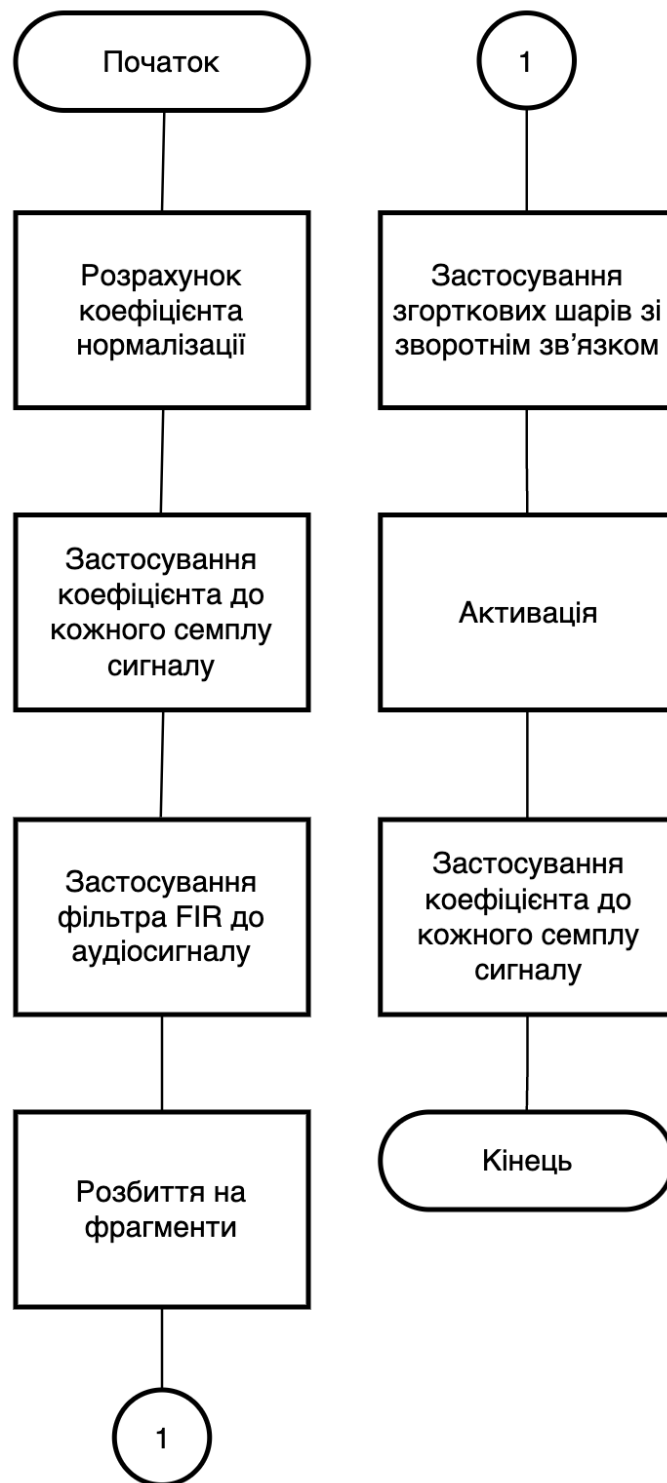
10. Yeh, D.T.; Abel, J.S.; Smith, J.O. Simplified, physically-informed models of distortion and overdrive guitar effects pedals. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Bordeaux, France, 10–15 September 2007; pp. 10–14.
11. Abeßer, J.; Kramer, P.; Dittmar, C.; Schuller, G.; Fraunhofer, I. Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm. In Proceedings of the International Conference on Digital Audio Effects (DAFx), Maynooth, Ireland, 2–5 September 2013; pp. 154–161.
12. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32; Curran Associates: Dutchess County, NY, USA, 2019; pp. 8024–8035.
13. Kahles, J.; Esqueda, F.; Välimäki, V. Oversampling for nonlinear waveshaping: Choosing the right filters. *J. Audio Eng. Soc.* 2019, 67, 440–449.
14. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532–1543).
15. Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. arXiv preprint arXiv:1409.1259.
16. Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

17. Balas, V. E., Kumar, R., & Srivastava, R. (2019). Recent Trends and Advances in Artificial Intelligence and Internet of Things. Springer Nature. ISBN 978-3-030-32644-9. Archived from the original on 2023-10-16.
18. Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 3642-3649). New York, NY: Institute of Electrical and Electronics Engineers (IEEE).
19. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015 (pp. 1-9). IEEE Computer Society. arXiv:1409.4842.
20. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580.
21. Duta, Ionut Cosmin; Georgescu, Mariana Iuliana; Ionescu, Radu Tudor (2021-08-16). "Contextual Convolutional Neural Networks". arXiv:2108.07387.
22. Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451-2471.

ДОДАТКИ

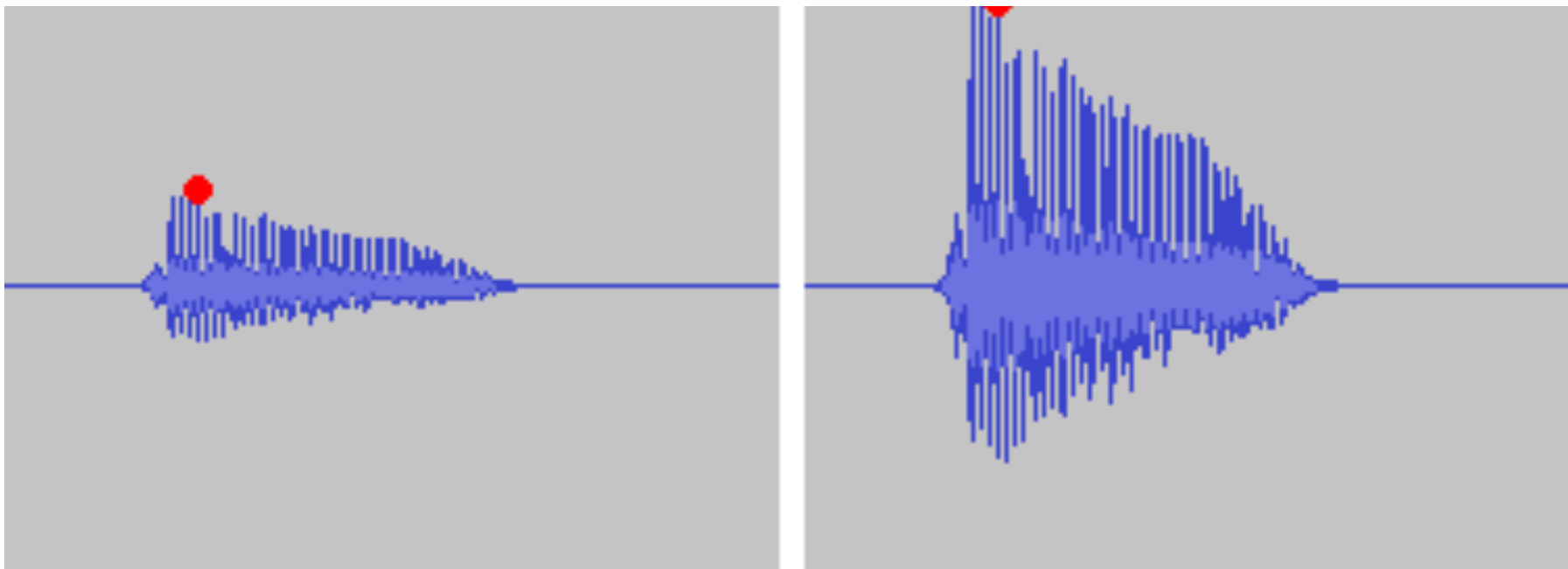
Додаток 1
Копії графічних матеріалів

Блок–схема запропонованого методу



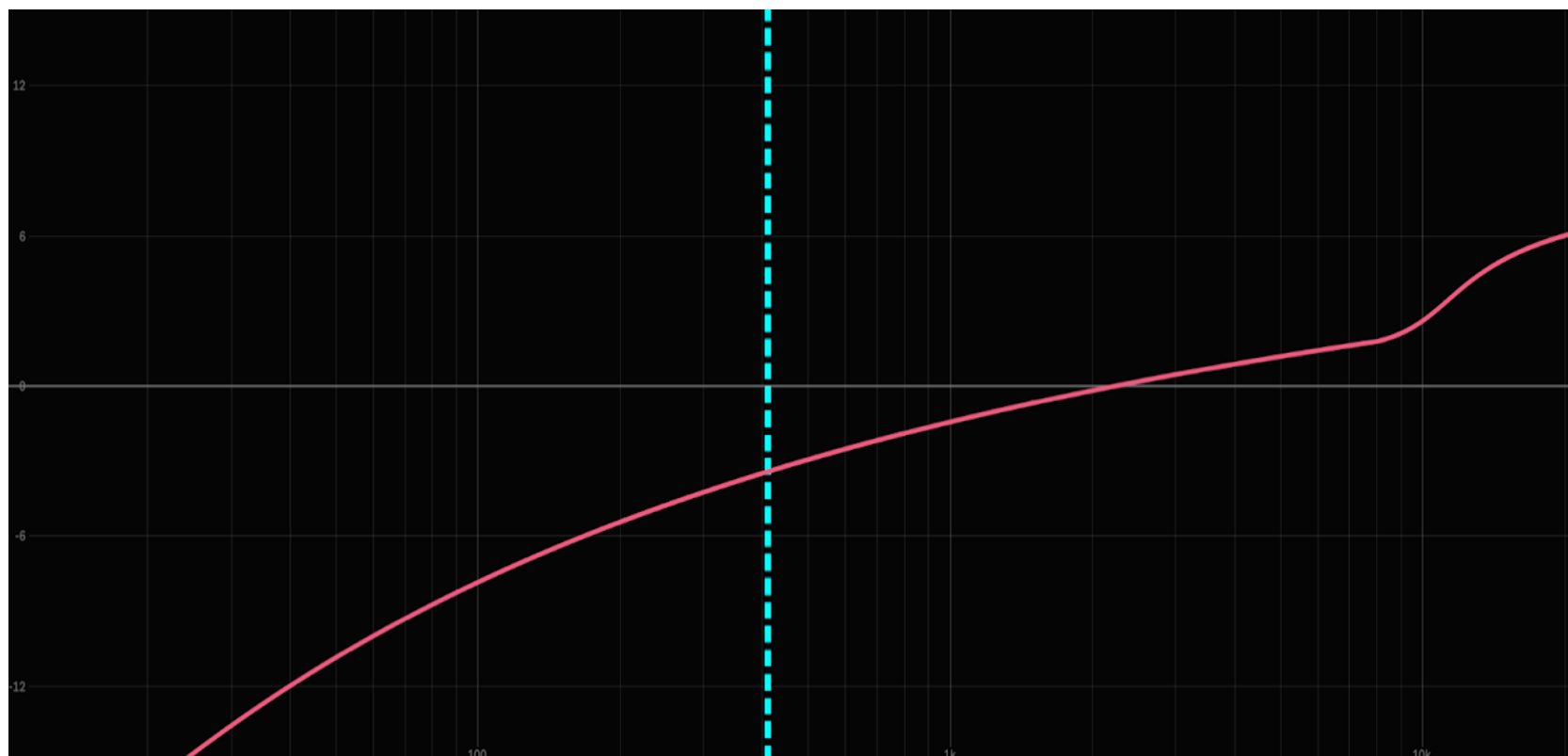
Анікєєв І.А. КП–21МН

Демонстрація процесу нормалізації



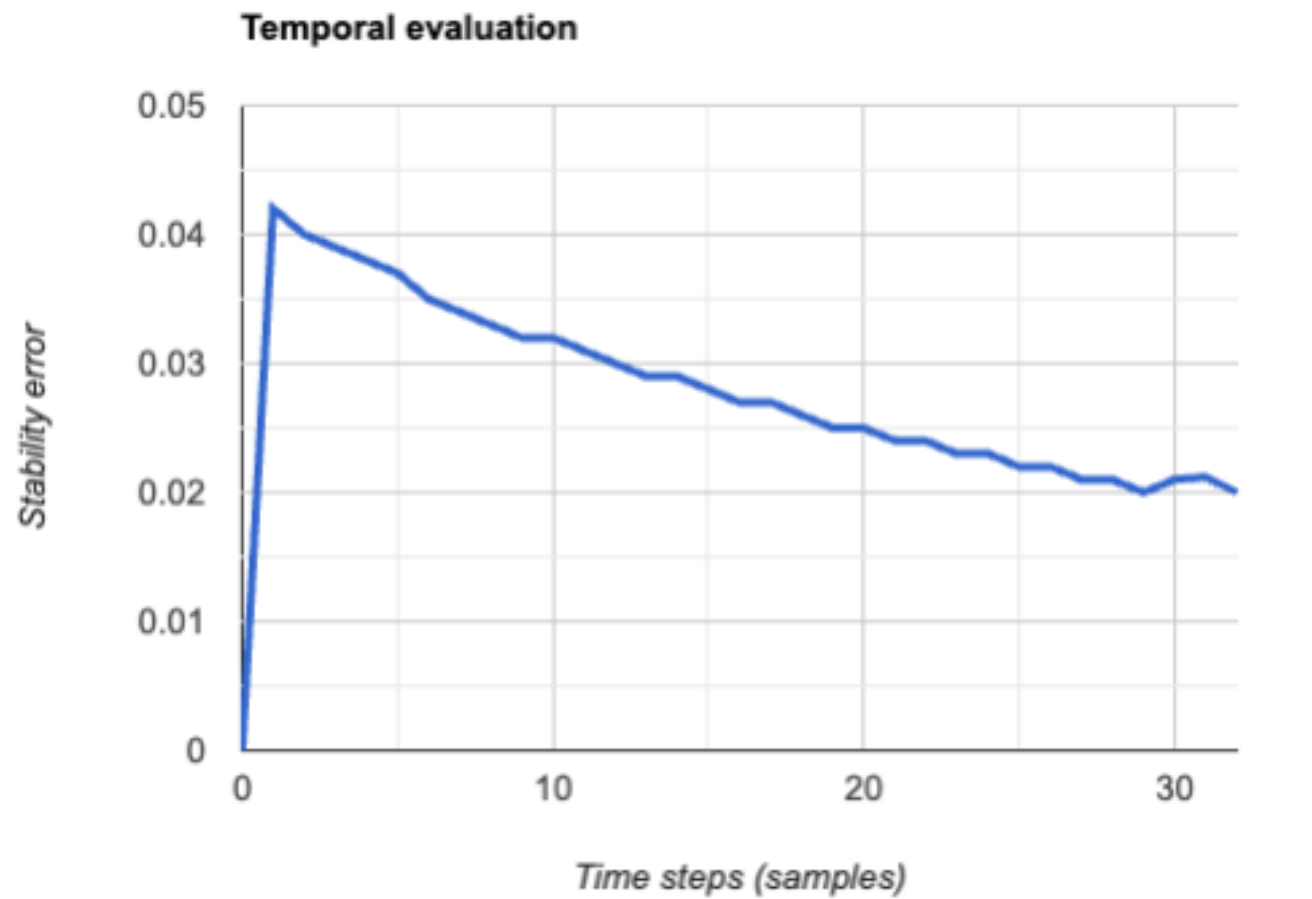
Анікєєв І.А. КП–21МН

Демонстрація роботи фільтру



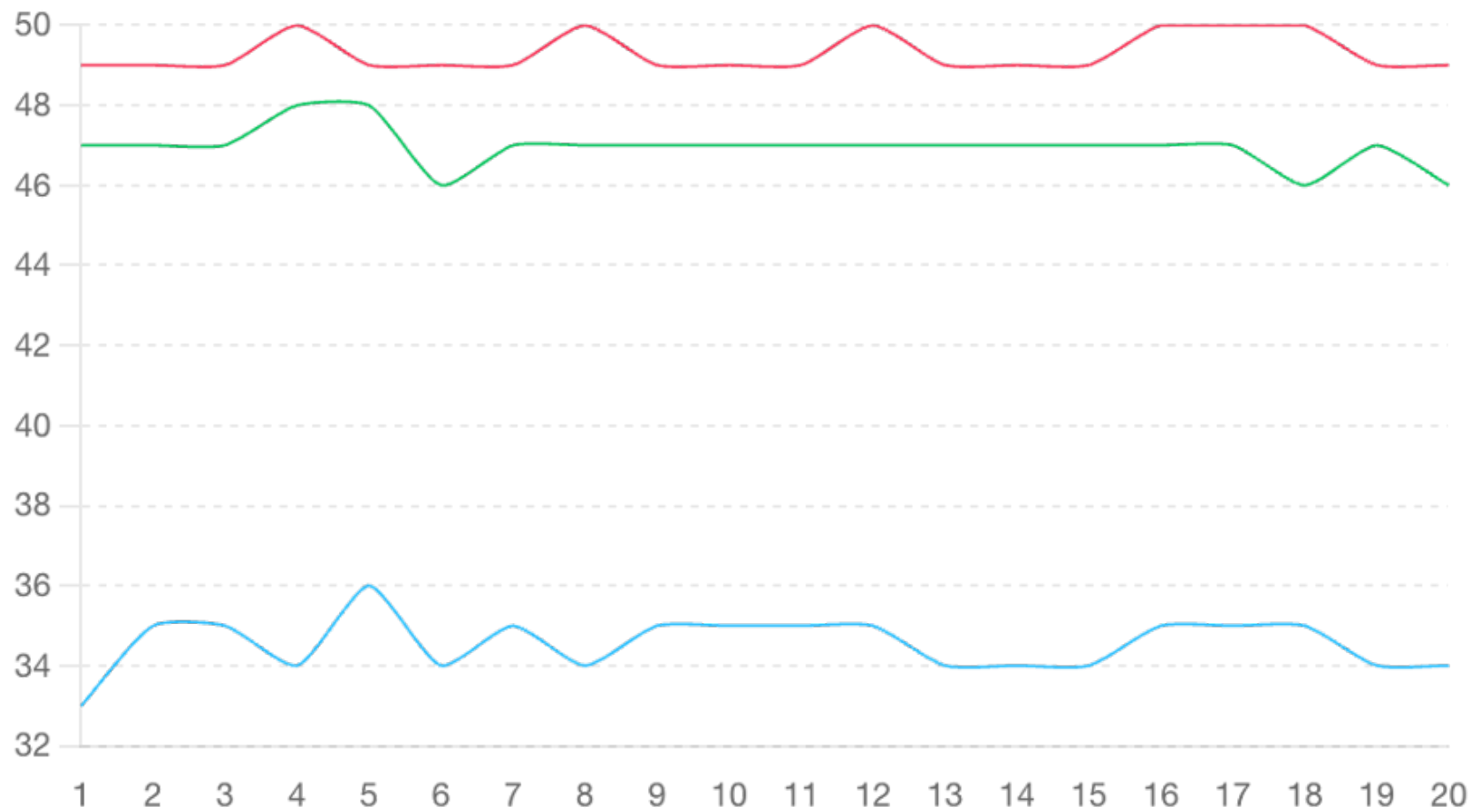
Анікеєв І.А. КП-21МН

Графік похибки стабільності для даного програмного забезпечення



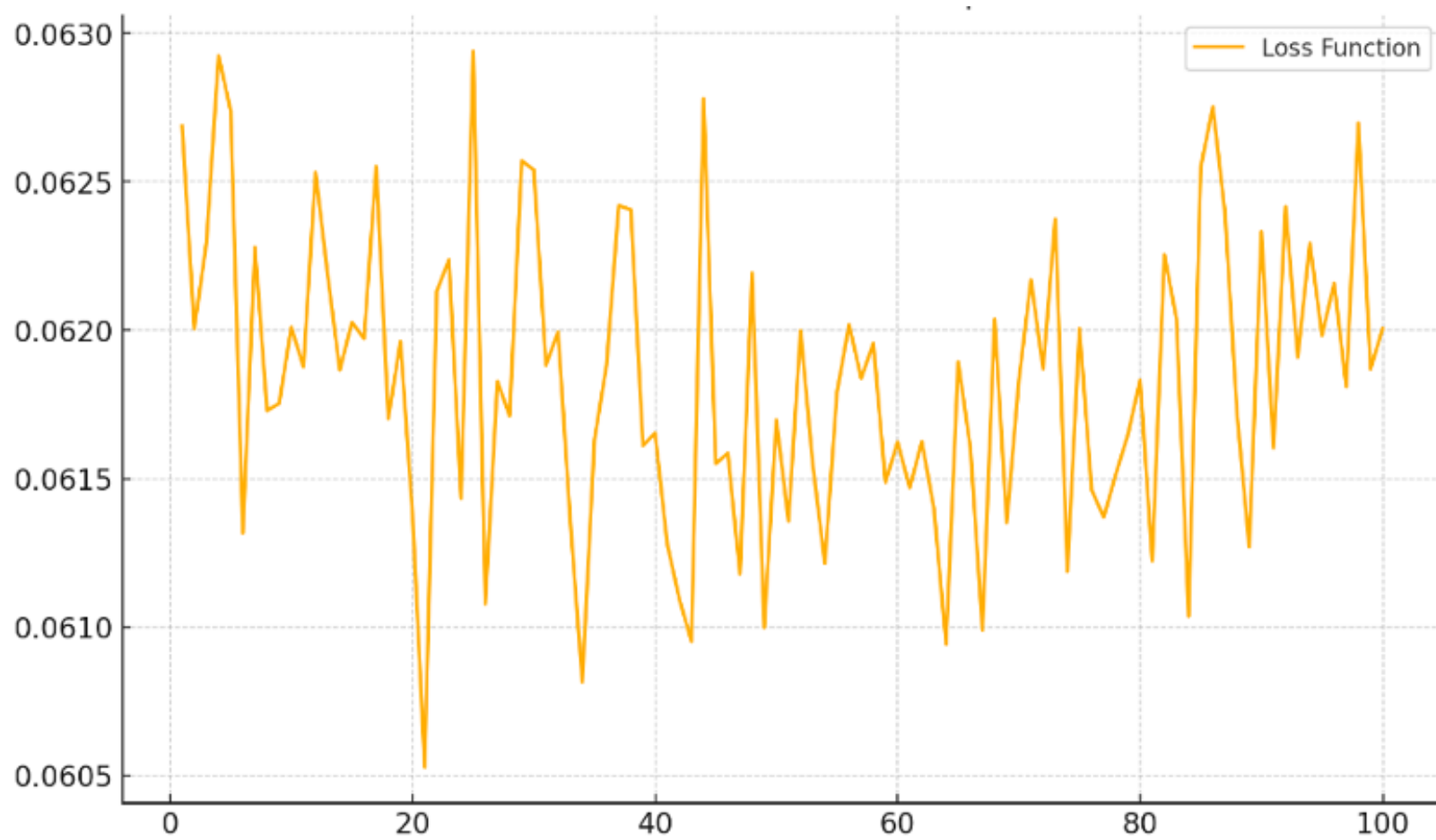
Анікєєв І.А. КП–21МН

Порівняльний графік часу обробки сигналу



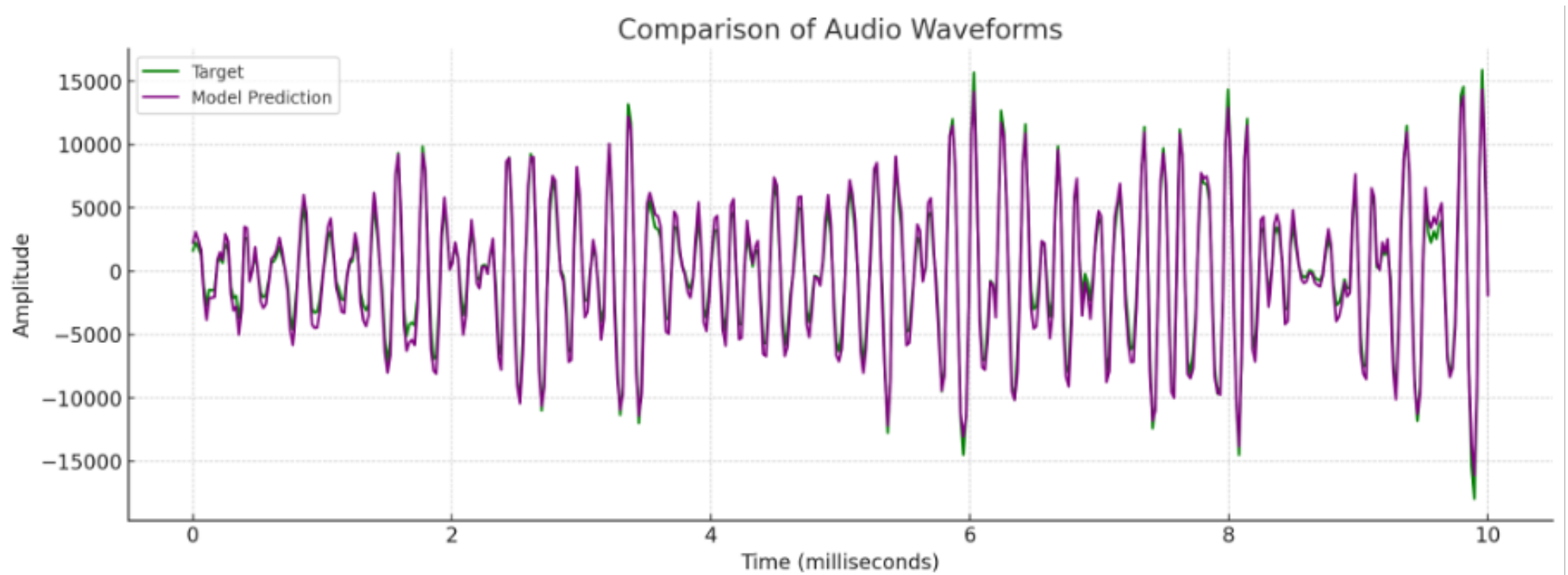
Анікєєв І.А. КП–21МН

Графік функції втрати



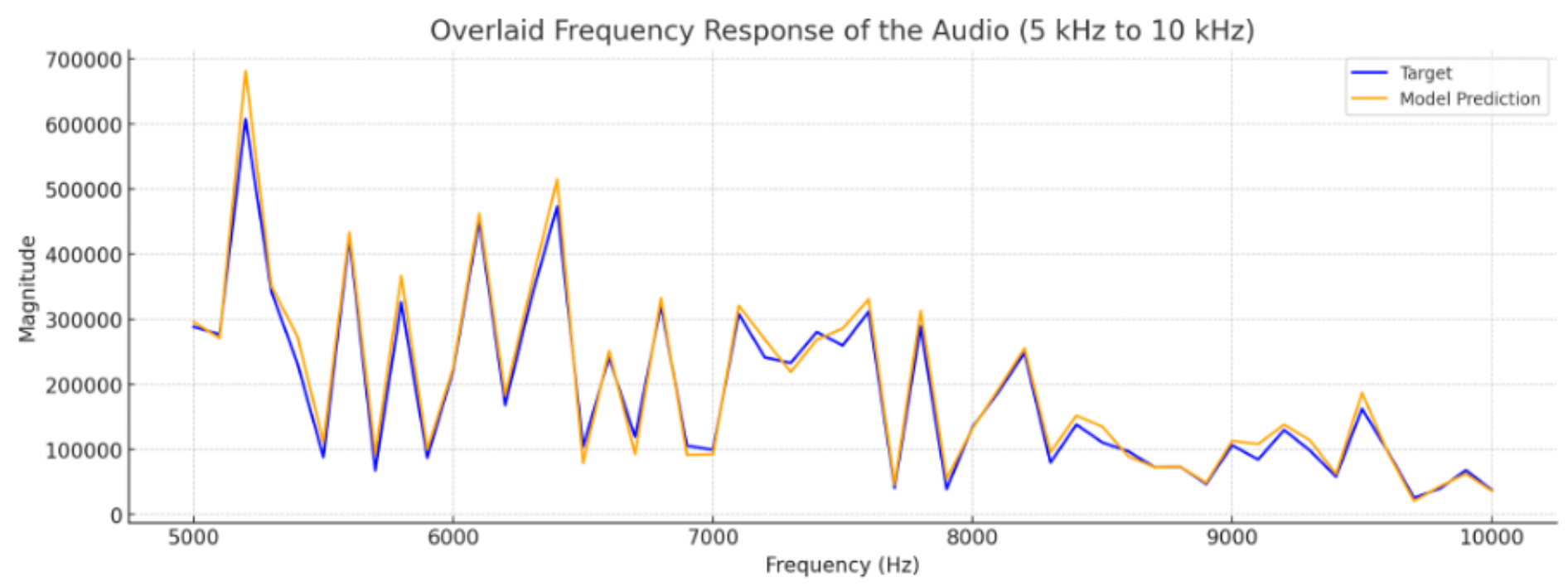
Анікеєв І.А. КП-21МН

Графік порівняльного аналізу звукових сигналів



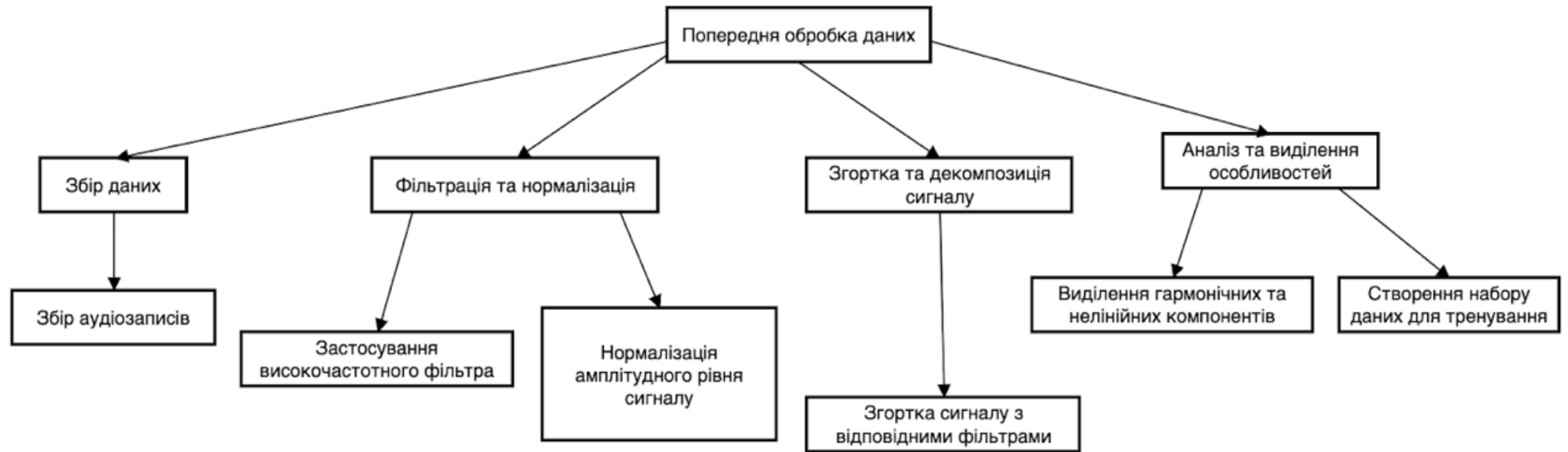
Анікєєв І.А. КП-21МН

Амплітудно-частотне порівняння звукових сигналів



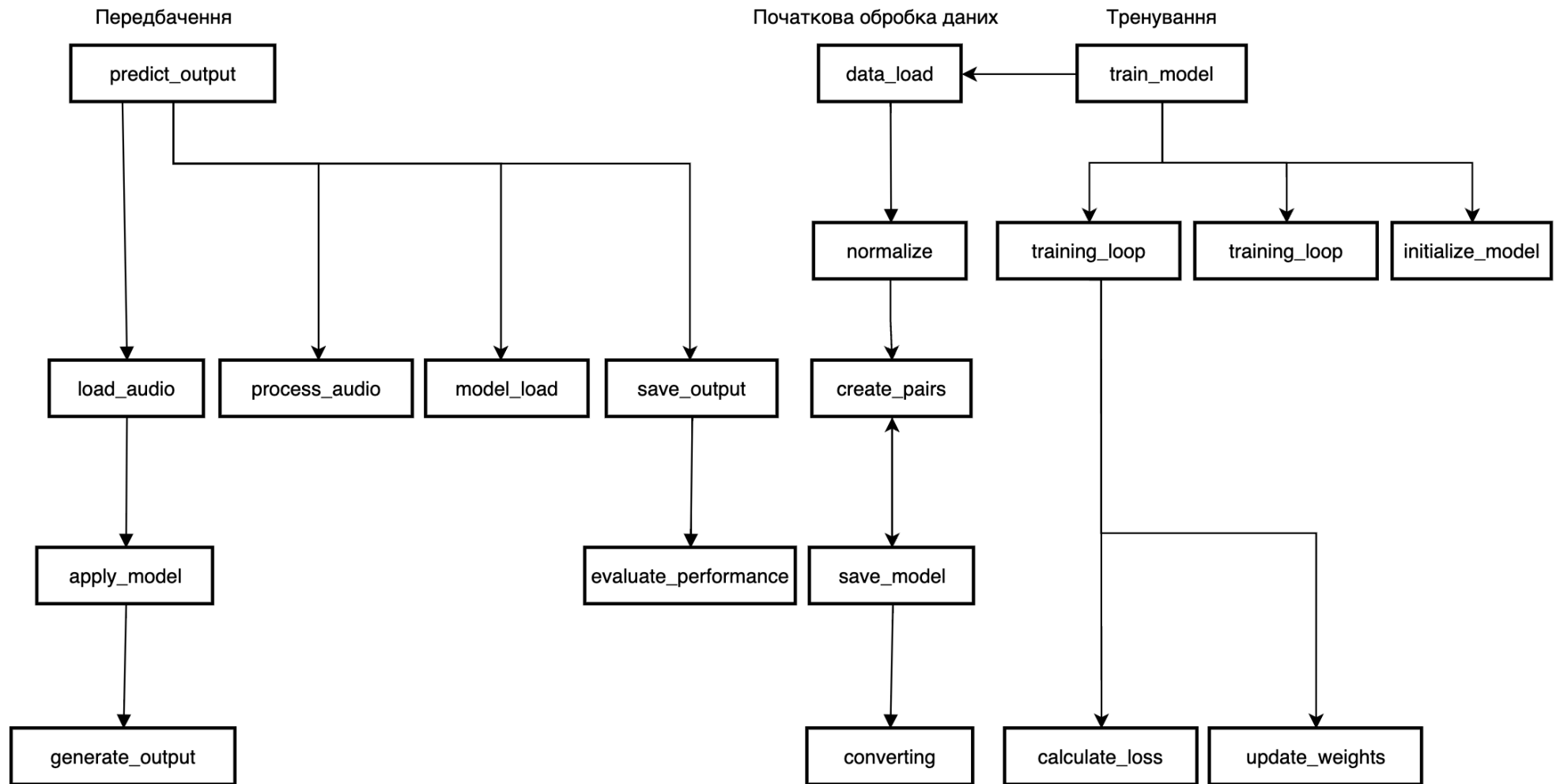
Анікеєв І.А. КП-21МН

Діаграма розгалуження для попередньої обробки даних



Анікєєв І.А. КП–21МН

Діаграма компонентів системи для мелодійного викривлення



Додаток 2
Лістинг програми

[illegible]

```

{
    activation = Activations::getActivationFuncArray(newActivationName);
    usesGating = Activations::isGated(newActivationName);
    size_t internalChannels = usesGating ? newOutputChannels * 2 :
newOutputChannels;
    conv.set_size(newInputChannels, internalChannels, newFilterWidth,
newDilation);
    out1x1.set_size(newOutputChannels, newOutputChannels, 1, 1);
    residual = newResidual;
}

void ConvolutionLayerArma::setWeight(std::vector<float> W, std::string
name)
{
    if ((name == "W_conv") || (name == "W"))
        conv = arma::cube(W.data(), conv.n_rows, conv.n_cols,
conv.n_slices, false, true);
    else if ((name == "b_conv") || (name == "b"))
        conv = arma::cube(W.data(), conv.n_rows, conv.n_cols,
conv.n_slices, false, true);
    else if (name == "W_out")
        out1x1 = arma::cube(W.data(), out1x1.n_rows, out1x1.n_cols,
out1x1.n_slices, false, true);
    else if (name == "b_out")
        out1x1 = arma::cube(W.data(), out1x1.n_rows, out1x1.n_cols,
out1x1.n_slices, false, true);
}

```

ConcatLayers.cpp

```

#include "WaveNetArma.h"

WaveNetArma::WaveNetArma(int inputChannels, int outputChannels, int
convolutionChannels,
                        int filterWidth, std::string activation,
std::vector<int> dilations) :
    convStack(convolutionChannels, filterWidth, dilations, activation),
    inputLayer(inputChannels, convolutionChannels, 1),
    outputLayer(convolutionChannels * dilations.size(), outputChannels, 1),
    inputChannels(inputChannels),
    outputChannels(outputChannels),
    filterWidth(filterWidth),
    skipChannels(convolutionChannels * static_cast<int>(dilations.size())),
    convolutionChannels(convolutionChannels),
    memoryChannels(Activations::isGated(activation) ? convolutionChannels *
2 : convolutionChannels),
    activation(activation),
    dilations(dilations)
{
}

void WaveNetArma::readDilations(var config)
{
    std::vector<int> newDilations;
    if (auto dilationsArray = config.getProperty("dilations",
var()).getArray())
    {
        for (int dil : *dilationsArray)
            newDilations.push_back(dil);
    }
    dilations = newDilations;
}

```

```

void WaveNetArma::prepareToPlay(int newSamplesPerBlock)
{
    samplesPerBlock = newSamplesPerBlock;
    convData.set_size(1, samplesPerBlock * memoryChannels);
    skipData.set_size(1, samplesPerBlock * skipChannels);
    convStack.prepareToPlay(samplesPerBlock);
}

void WaveNetArma::copyInputData(const float **inputData, int numSamples)
{
    float *writePtr = convData.memptr();
    for (int ch = 0; ch < inputChannels; ++ch)
    {
        int start_idx = idx(ch, 0, numSamples);
        const float *chData = inputData[ch];
        for (int i = 0; i < numSamples; ++i)
            writePtr[start_idx + i] = chData[i];
    }
}

void WaveNetArma::copyOutputData(float **outputData, int numSamples)
{
    const float *readPtr = skipData.memptr();
    for (int ch = 0; ch < outputChannels; ++ch)
    {
        int start_idx = idx(ch, 0, numSamples);
        float *chData = outputData[ch];
        for (int i = 0; i < numSamples; ++i)
            chData[i] = readPtr[start_idx + i];
    }
}

void WaveNetArma::process(const float **inputData, float **outputData, int
numSamples)
{
    if (numSamples > samplesPerBlock)
        prepareToPlay(numSamples);
    copyInputData(inputData, numSamples);
    inputLayer.process(convData.memptr(), numSamples);
    convStack.process(convData.memptr(), skipData.memptr(), numSamples);
    outputLayer.process(skipData.memptr(), numSamples);
    copyOutputData(outputData, numSamples);
}

int WaveNetArma::idx(int ch, int i, int numSamples)
{
    return ch * numSamples + i;
}

void WaveNetArma::setWeight(std::vector<float> W, int layerIdx, std::string
name)
{
    if (layerIdx < 0)
    {
        inputLayer.setWeight(W, name);
    }
    else if (layerIdx >= convStack.getNumLayers())
    {
        outputLayer.setWeight(W, name);
    }
    else
    {
        convStack.setWeight(W, layerIdx, name);
    }
}

```

```

    }
}

void WaveNetArma::setParams(int newInputChannels, int newOutputChannels,
int newConvChannels,
                        int newFilterWidth, std::string newActivation,
std::vector<int> newDilations)
{
    inputChannels = newInputChannels;
    outputChannels = newOutputChannels;
    activation = newActivation;
    convolutionChannels = newConvChannels;
    memoryChannels = Activations::isGated(activation) ? convolutionChannels
* 2 : convolutionChannels;
    filterWidth = newFilterWidth;
    dilations = newDilations;
    skipChannels = convolutionChannels *
static_cast<int>(dilations.size());
    inputLayer.setParams(inputChannels, convolutionChannels, 1, 1, false,
"linear");
    outputLayer.setParams(skipChannels, outputChannels, 1, 1, false,
"linear");
    convStack.setParams(convolutionChannels, filterWidth, dilations,
activation, true);
    prepareToPlay(samplesPerBlock);
}}

```

Model.py

```

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
import pytorch_lightning as pl
import pickle
import argparse

class CausalConv1d(nn.Conv1d):
    def __init__(self, in_channels, out_channels, kernel_size, stride=1,
dilation=1, groups=1, bias=True):
        self.__padding = (kernel_size - 1) * dilation
        super(CausalConv1d, self).__init__(
            in_channels,
            out_channels,
            kernel_size=kernel_size,
            stride=stride,
            padding=self.__padding,
            dilation=dilation,
            groups=groups,
            bias=bias
        )

    def forward(self, input):
        result = super(CausalConv1d, self).forward(input)
        if self.__padding != 0:
            return result[:, :, :-self.__padding]
        return result

def _conv_stack(dilations, in_channels, out_channels, kernel_size):
    """

```

```

    return nn.ModuleList(
        [
            CausalConv1d(
                in_channels=in_channels,
                out_channels=out_channels,
                dilation=d,
                kernel_size=kernel_size
            ) for d in dilations
        ]
    )

)

class WaveNet(nn.Module):
    def __init__(self, num_channels, dilation_depth, num_repeat,
kernel_size=2):
        super(WaveNet, self).__init__()
        dilations = [2 ** d for d in range(dilation_depth)] * num_repeat
        internal_channels = int(num_channels * 2)
        self.hidden = _conv_stack(dilations, num_channels,
internal_channels, kernel_size)
        self.residuals = _conv_stack(dilations, num_channels, num_channels,
1)

        self.input_layer = CausalConv1d(in_channels=1,
out_channels=num_channels, kernel_size=1)
        self.linear_mix = nn.Conv1d(
            in_channels=num_channels * dilation_depth * num_repeat,
            out_channels=1,
            kernel_size=1
        )
        self.num_channels = num_channels

    def forward(self, x):
        out = x
        skips = []
        out = self.input_layer(out)

        for hidden, residual in zip(self.hidden, self.residuals):
            x = out
            out_hidden = hidden(x)
            # Gated activation
            out_hidden_split = torch.split(out_hidden, self.num_channels,
dim=1)
            out = torch.tanh(out_hidden_split[0]) *
torch.sigmoid(out_hidden_split[1])
            skips.append(out)
            out = residual(out)
            out = out + x[:, :, -out.size(2):]

        # Modified "postprocess" step:
        out = torch.cat([s[:, :, -out.size(2):] for s in skips], dim=1)
        out = self.linear_mix(out)
        return out

def error_to_signal(y, y_pred):
    """
    y, y_pred = pre_emphasis_filter(y), pre_emphasis_filter(y_pred)
    return (y - y_pred).pow(2).sum(dim=2) / (y.pow(2).sum(dim=2) + 1e-10)

def pre_emphasis_filter(x, coeff=0.95):
    return torch.cat((x[:, :, 0:1], x[:, :, 1:] - coeff * x[:, :, :-1]),
dim=2)

```

```

class (pl.LightningModule):
    def __init__(self, hparams):
        super(self).__init__()
        self.save_hyperparameters(hparams)
        self.wavenet = WaveNet(
            num_channels=hparams.num_channels,
            dilation_depth=hparams.dilation_depth,
            num_repeat=hparams.num_repeat,
            kernel_size=hparams.kernel_size
        )

    def prepare_data(self):
        data = pickle.load(open(self.hparams.data, "rb"))
        ds = lambda x, y: TensorDataset(torch.from_numpy(x),
torch.from_numpy(y))
        self.train_ds = ds(data["x_train"], data["y_train"])
        self.valid_ds = ds(data["x_valid"], data["y_valid"])

    def configure_optimizers(self):
        return torch.optim.Adam(self.wavenet.parameters(),
lr=self.hparams.learning_rate)

    def train_dataloader(self):
        return DataLoader(self.train_ds, shuffle=True,
batch_size=self.hparams.batch_size, num_workers=4)

    def val_dataloader(self):
        return DataLoader(self.valid_ds,
batch_size=self.hparams.batch_size, num_workers=4)

    def forward(self, x):
        return self.wavenet(x)

    def training_step(self, batch, batch_idx):
        x, y = batch
        y_pred = self.forward(x)
        loss = error_to_signal(y[:, :, -y_pred.size(2):], y_pred).mean()
        self.log('

```

WaveNetStruct.cpp

```

#include "WaveNetArma.h"

WaveNetArma::WaveNetArma(int inputChannels, int outputChannels, int
convolutionChannels,
                        int filterWidth, std::string activation,
std::vector<int> dilations) :
    convStack(convolutionChannels, filterWidth, dilations, activation),
    inputLayer(inputChannels, convolutionChannels, 1),
    outputLayer(convolutionChannels * dilations.size(), outputChannels, 1),
    inputChannels(inputChannels),
    outputChannels(outputChannels),
    filterWidth(filterWidth),
    skipChannels(convolutionChannels * static_cast<int>(dilations.size())),
    convolutionChannels(convolutionChannels),
    memoryChannels(Activations::isGated(activation) ? convolutionChannels *
2 : convolutionChannels),
    activation(activation),
    dilations(dilations)
{
}

```

```

void WaveNetArma::readDilations(var config)
{
    std::vector<int> newDilations;
    if (auto dilationsArray = config.getProperty("dilations",
var()).getArray())
    {
        for (int dil : *dilationsArray)
            newDilations.push_back(dil);
    }
    dilations = newDilations;
}

void WaveNetArma::prepareToPlay(int newSamplesPerBlock)
{
    samplesPerBlock = newSamplesPerBlock;
    convData.set_size(1, samplesPerBlock * memoryChannels);
    skipData.set_size(1, samplesPerBlock * skipChannels);
    convStack.prepareToPlay(samplesPerBlock);
}

void WaveNetArma::copyInputData(const float **inputData, int numSamples)
{
    float *writePtr = convData.memptr();
    for (int ch = 0; ch < inputChannels; ++ch)
    {
        int start_idx = idx(ch, 0, numSamples);
        const float *chData = inputData[ch];
        for (int i = 0; i < numSamples; ++i)
            writePtr[start_idx + i] = chData[i];
    }
}

void WaveNetArma::copyOutputData(float **outputData, int numSamples)
{
    const float *readPtr = skipData.memptr();
    for (int ch = 0; ch < outputChannels; ++ch)
    {
        int start_idx = idx(ch, 0, numSamples);
        float *chData = outputData[ch];
        for (int i = 0; i < numSamples; ++i)
            chData[i] = readPtr[start_idx + i];
    }
}

void WaveNetArma::process(const float **inputData, float **outputData, int
numSamples)
{
    if (numSamples > samplesPerBlock)
        prepareToPlay(numSamples);
    copyInputData(inputData, numSamples);
    inputLayer.process(convData.memptr(), numSamples);
    convStack.process(convData.memptr(), skipData.memptr(), numSamples);
    outputLayer.process(skipData.memptr(), numSamples);
    copyOutputData(outputData, numSamples);
}

int WaveNetArma::idx(int ch, int i, int numSamples)
{
    return ch * numSamples + i;
}

```

```

void WaveNetArma::setWeight(std::vector<float> W, int layerIdx, std::string
name)
{
    if (layerIdx < 0)
    {
        inputLayer.setWeight(W, name);
    }
    else if (layerIdx >= convStack.getNumLayers())
    {
        outputLayer.setWeight(W, name);
    }
    else
    {
        convStack.setWeight(W, layerIdx, name);
    }
}

void WaveNetArma::setParams(int newInputChannels, int newOutputChannels,
int newConvChannels,
                           int newFilterWidth, std::string newActivation,
std::vector<int> newDilations)
{
    inputChannels = newInputChannels;
    outputChannels = newOutputChannels;
    activation = newActivation;
    convolutionChannels = newConvChannels;
    memoryChannels = Activations::isGated(activation) ? convolutionChannels
* 2 : convolutionChannels;
    filterWidth = newFilterWidth;
    dilations = newDilations;
    skipChannels = convolutionChannels *
static_cast<int>(dilations.size());
    inputLayer.setParams(inputChannels, convolutionChannels, 1, 1, false,
"linear");
    outputLayer.setParams(skipChannels, outputChannels, 1, 1, false,
"linear");
    convStack.setParams(convolutionChannels, filterWidth, dilations,
activation, true);
    prepareToPlay(samplesPerBlock);
}

```

DataPreparation.py

```

import pickle
import torch
from scipy.io import wavfile
import argparse
import numpy as np

def save_wav_file(filename, data):
    # Save the waveform data to a file
    wavfile.write(filename, 44100, data.flatten().astype(np.float32))

@torch.no_grad()
def evaluate_model(args):
    model.eval()
    data = pickle.load(open(args.data_path, "rb"))

    x_test = data["x_test"]
    prev_sample = np.concatenate((np.zeros_like(x_test[0:1]), x_test[:-1]),
axis=0)
    padded_x_test = np.concatenate((prev_sample, x_test), axis=2)

```

```

y_predictions = []
for x in np.array_split(padded_x_test, 10):
    y_predictions.append(model(torch.from_numpy(x)).numpy())

y_predictions = np.concatenate(y_predictions)
y_predictions = y_predictions[:, :, -x_test.shape[2]:]

save_wav_file("predicted_output.wav", y_predictions)
save_wav_file("input_test.wav", data["x_test"] * data["std"] +
data["mean"])
save_wav_file("expected_output.wav", data["y_test"])

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--model_path", default="models/.ckpt")
    parser.add_argument("--data_path", default="data.pickle")
    args = parser.parse_args()
    evaluate_model(args)

```

Activations.cpp

```

#include <cmath>
#include <vector>
#include <algorithm>

namespace activation {

    double relu(double x) {
        return x > 0 ? x : 0;
    }

    double relu_derivative(double x) {
        return x > 0 ? 1 : 0;
    }

    double sigmoid(double x) {
        return 1 / (1 + exp(-x));
    }

    double sigmoid_derivative(double x) {
        double s = sigmoid(x);
        return s * (1 - s);
    }

    double tanh(double x) {
        return std::tanh(x);
    }

    double tanh_derivative(double x) {
        return 1 - std::tanh(x) * std::tanh(x);
    }

    std::vector<double> softmax(const std::vector<double>& x) {
        std::vector<double> result(x.size());
        double max = *std::max_element(x.begin(), x.end());
        double sum = 0.0;

        for (size_t i = 0; i < x.size(); ++i) {
            result[i] = exp(x[i] - max);
            sum += result[i];
        }

        for (size_t i = 0; i < x.size(); ++i) {

```

```

        result[i] /= sum;
    }

    return result;
}
}

```

Convolution.cpp

```

#include <cmath>
#include <vector>

namespace convolution {

    std::vector<double> conv1d(const std::vector<double>& signal, const
std::vector<double>& kernel) {
        int signal_size = signal.size();
        int kernel_size = kernel.size();
        int output_size = signal_size - kernel_size + 1;
        std::vector<double> output(output_size);

        for (int i = 0; i < output_size; ++i) {
            output[i] = 0;
            for (int j = 0; j < kernel_size; ++j) {
                output[i] += signal[i + j] * kernel[j];
            }
        }

        return output;
    }

    std::vector<std::vector<double>> conv2d(const
std::vector<std::vector<double>>& signal, const
std::vector<std::vector<double>>& kernel) {
        int signal_rows = signal.size();
        int signal_cols = signal[0].size();
        int kernel_rows = kernel.size();
        int kernel_cols = kernel[0].size();
        int output_rows = signal_rows - kernel_rows + 1;
        int output_cols = signal_cols - kernel_cols + 1;
        std::vector<std::vector<double>> output(output_rows,
std::vector<double>(output_cols));

        for (int i = 0; i < output_rows; ++i) {
            for (int j = 0; j < output_cols; ++j) {
                output[i][j] = 0;
                for (int ki = 0; ki < kernel_rows; ++ki) {
                    for (int kj = 0; kj < kernel_cols; ++kj) {
                        output[i][j] += signal[i + ki][j + kj] *
kernel[ki][kj];
                    }
                }
            }
        }

        return output;
    }
}
#include <algorithm>

```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

**«Програмний метод мелодійного викривлення
аудіоматеріалів за допомогою штучного інтелекту»**

Доповідач: Анікеєв Ігор Анатолійович

Науковий керівник: кандидат фізично-математичних наук, доцент,
Нещадим Олександр Михайлович

Київ – 2024



Актуальність дослідження

- Мануальний саунд-дизайн складний, довгий, і дорогий;
- Занадто довгий пошук унікальних та креативних засобів обробки, що зберігають обчислювальну ефективність;
- Мала кількість аудіоматеріалів готова до зведення.

ПОСТАНОВКА ЗАДАЧІ

1. Дослідити та проаналізувати існуючі методи та їхні обмеження;
2. Запропонувати новий метод, використовуючи архітектури нейронних мереж для швидкого перетворення аудіоданих;
3. Оптимізувати алгоритм для обробки в реальному часі;
4. Експериментально дослідити ефективність розробленого методу шляхом оцінки роботи ПЗ за критеріями;
5. Провести аналіз отриманих результатів.

Існуючі методи

- Нейронні мережі CNN
- Рекурентні нейронні мережі (RNN)
- Гібридні моделі (CNN+RNN)
- Генеративні змагальні мережі (GANs)
- Методи вилучення ознак
- Передавальне навчання
- Навчання з підкріпленням
- Алгоритми обробки сигналів
- Моделі взаємодії з користувачем

ОГЛЯД ІСНУЮЧИХ МЕТОДІВ

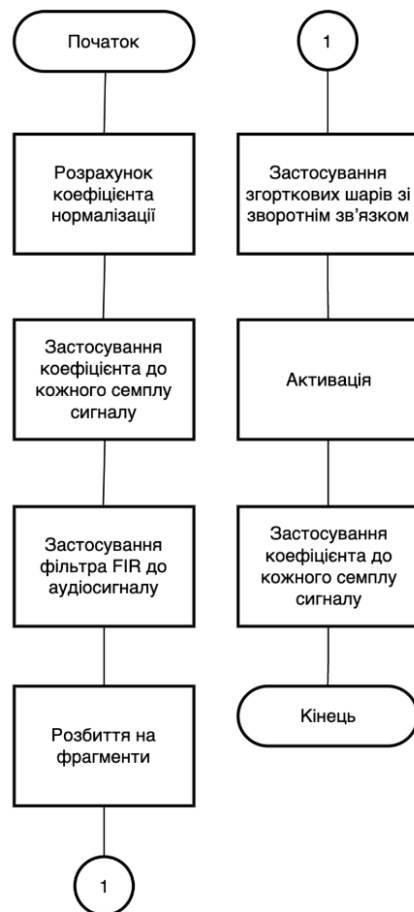
Програмне рішення	Аналіз аудіо	EQ	Компресія	Ефектор	Час обробки звуку
Baby Audio TAIP	Ні	Ні	Так	Так	Швидкий
Guitar Rig 7	Ні	Так	Ні	Так	Швидкий
iZotope Ozone 9	Так	Так	Так	Ні	Повільний
LANDR	Так	Так	Так	Ні	Повільний
Neutron 3	Так	Так	Так	Так	Повільний



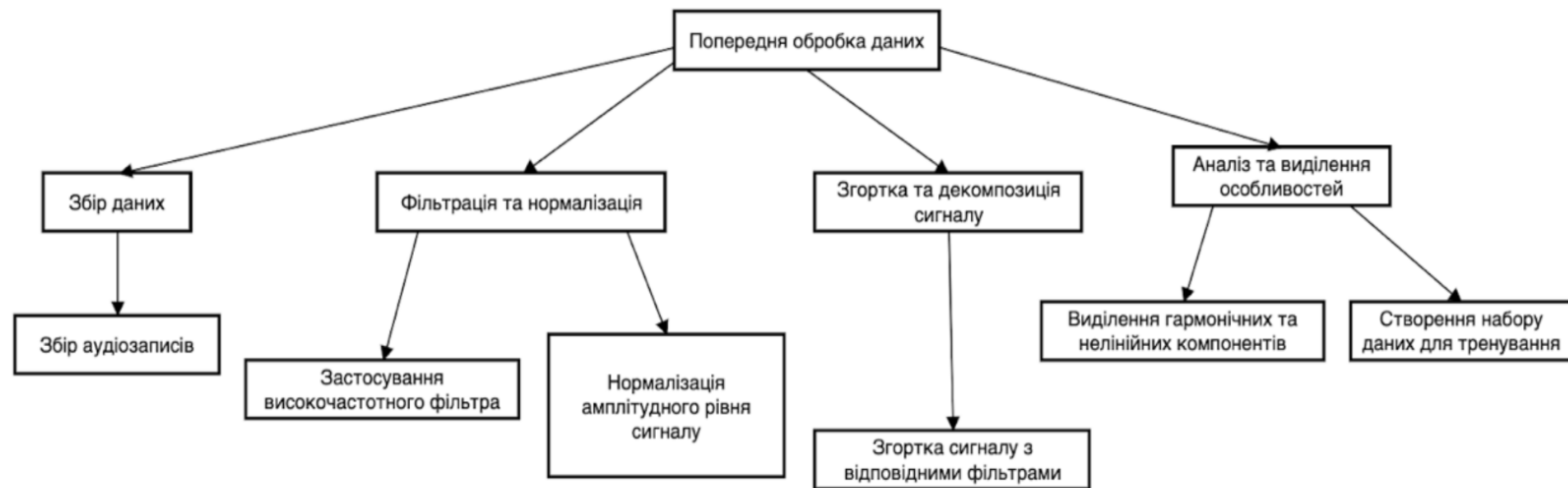
Запропонований метод

1. Використання модифікованих архітектур згорткових нейронних мереж (CNN) для створення складних аудіоперетворень
2. Механізм зворотного зв'язку
3. Інтеграція моделей у програмне забезпечення
4. Ефективність у реальному часі
5. Порівняння з існуючими методами

ОГЛЯД ЗАПРОПОНОВАНОГО МЕТОДУ



ОГЛЯД ЗАПРОПОНОВАНОГО МЕТОДУ





Опис запропонованого методу

Згортковий шар, який використовується в моделі покращений за допомогою механізму зворотного зв'язку. Вхідний сигнал спочатку обробляється розширеним каузальним фільтром FIR. Окремі фільтри в ядрі мають імпульсні характеристики:

$$h[n] = \sum_{m=0}^{M-1} w_m \delta[n - md_k]$$

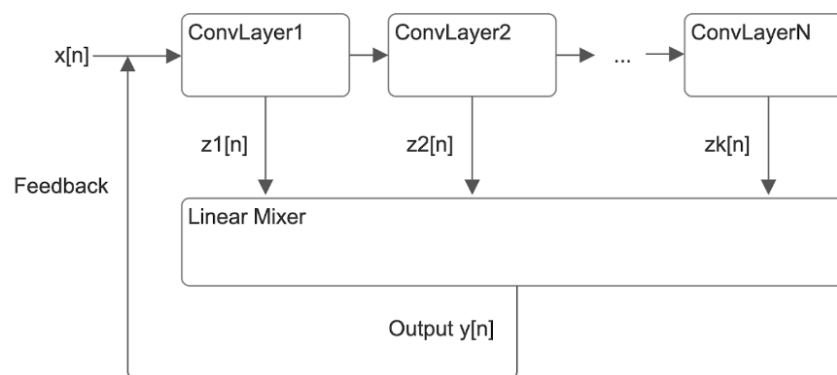
Вихідні дані з останнього шару повертаються в систему як додатковий вхід до згорткового рівня, утворюючи цикл зворотного зв'язку $fb[n]$, отриманий з кінцевого виходу моделі.

Опис запропонованого методу

Шари включають залишкове з'єднання, що означає, що вхідними даними для наступного рівня є:

$$x_{k+1}[n] = W_k z_k[n] + x_k[n] + fb[n].$$

Блок-схема моделі зображена на малюнку нижче.





Практична значущість

1. Реалізація в реальному часі
2. Креативне маніпулювання звуком
3. Інтеграція з існуючими DAW
4. Оптимізація продуктивності



Архітектура та реалізація

1. Структурні компоненти системи
2. Обробка вхідного сигналу
3. Модуль моделювання
4. Механізм зворотного зв'язку
5. Оптимізація моделі
6. Результати експериментів

Обрані засоби реалізації



1. Numpy
2. TensorFlow
3. Armadillo
4. librosa
5. Python
6. JUCE
7. C++

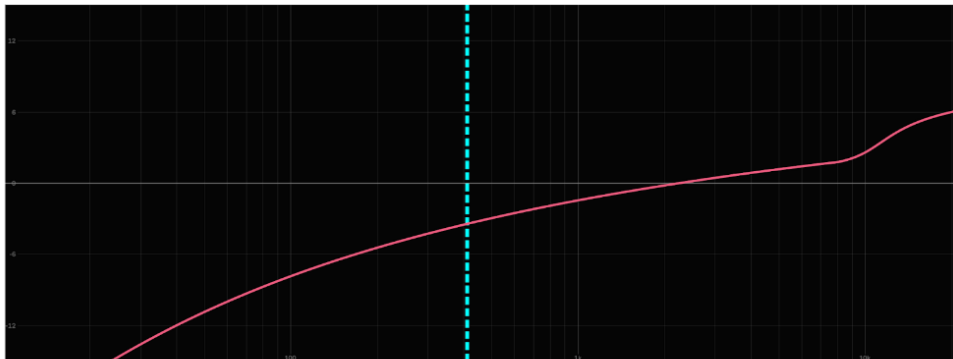
Тренування моделі

Модель буде тренована шляхом оптимізації співвідношення помилки (ESR) до сигналу та налаштований фільтр попереднього виділення середніх частот, що розраховується за формулою:

$$F(z) = 1 - \beta z^{-1}$$

І матиме приблизний вигляд, що показаний на малюнку нижче.

Як бачимо, даний фільтр підсилює середні і високі частоти, з частотою зрізу близько 5kHz, виділеною пунктирною лінією.



Оцінка роботи методу

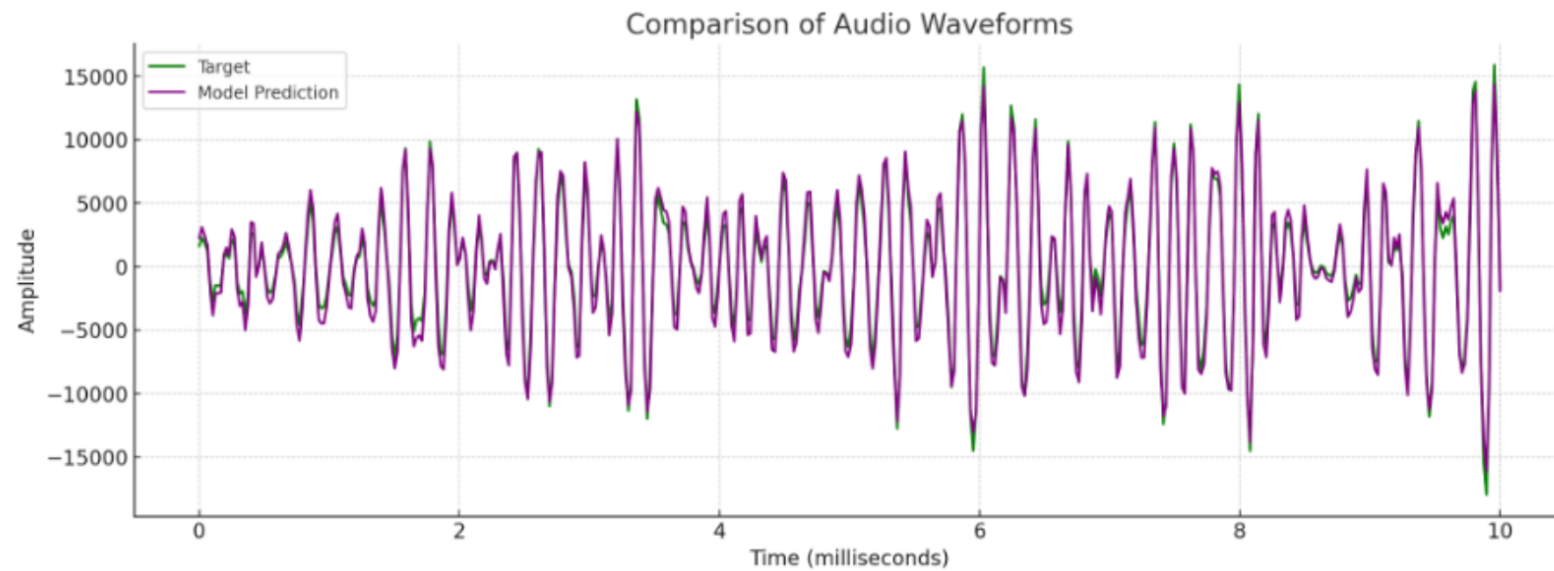
Середня втрата за 1000 епох - 0.0716.

ESR - 6%.

Точність: 94%

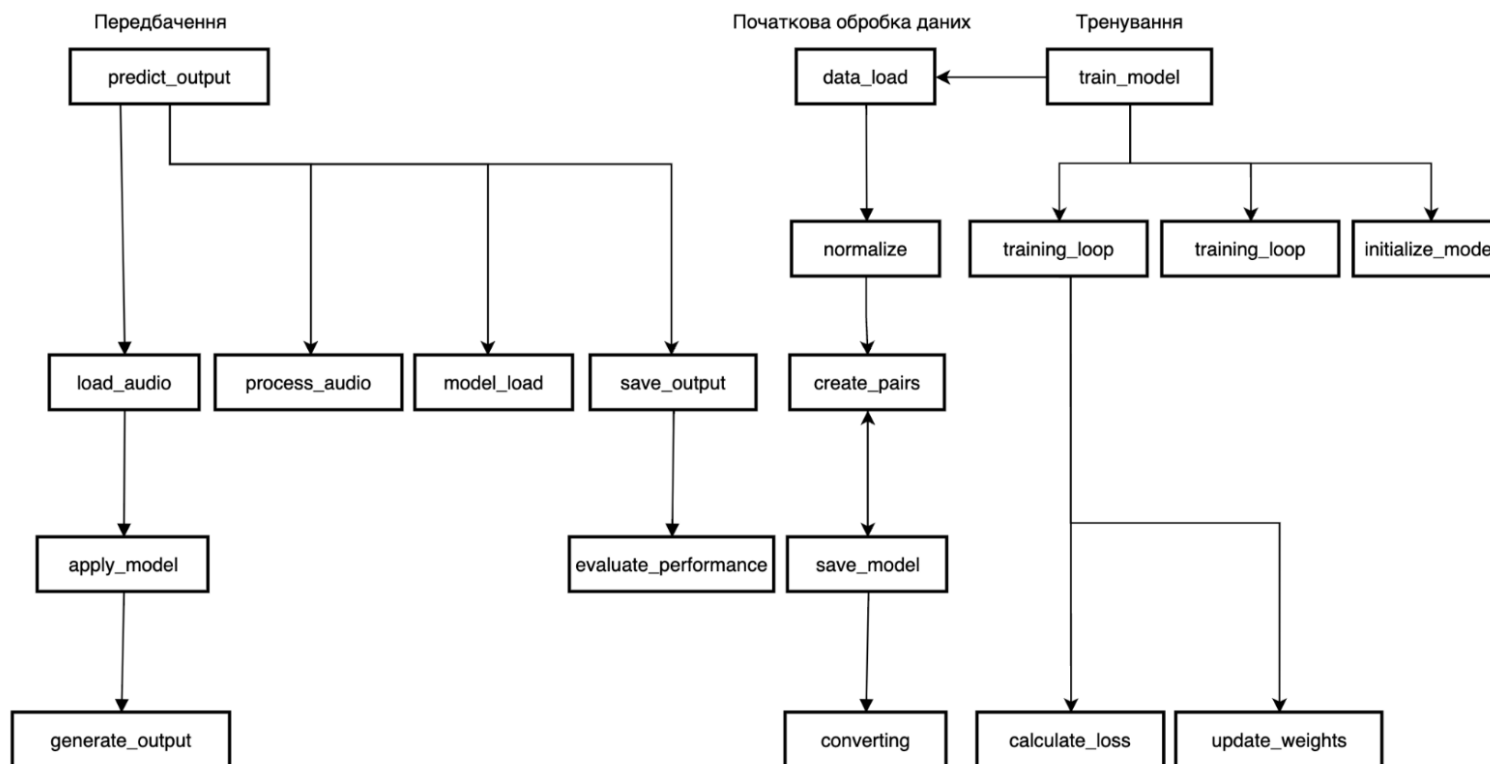


Оцінка роботи методу



Графік порівняльного аналізу звукових сигналів

Діаграма компонентів системи





Висновки та подальша робота

- Розглянуто складові частини методу.
- Встановлено вимоги для вхідних даних, обрано тестовий набір даних для проведення досліджень.
- Розроблено алгоритм на основі запропонованих гіпотез.
- Встановлено метрики оцінки роботи відповідно до поставленого завдання.
- Програмно реалізовано запропонований метод та протестовано роботу програми.
- Проаналізовано отримані результати.

В подальшому:

Порівняння ефективності інших функцій активацій та ефективної кількості шарів та каналів.