Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

VST3 ШІ-еквалайзер для автоматизованого налаштування звуку в цифровій аудіо робочій станції
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-2
Олексій Ларіонов
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ст. викл. В'ячеслав Гребенюк
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ларіонову Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи VST3 ШІ-еквалайзер для автоматизованого налаштування звуку в цифровій аудіо робочій станції

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 червня 20 25 р.

3. Вихідні дані до роботи науково-технічні публікації, дані Інтернет-джерел, документація JUCE, набір даних для тренування та тестування системи, аудіоматеріали для тренування агента

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Методи вирішення задачі

3) Програмна реалізація плагіну

4) Використання плагіна в цифрових аудіо робочих станціях

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05-21.05	виконано
3	Огляд науково-дослідницьких джерел	21.05-23.05	виконано
	за темою дослідження		
4	Постановка задачі	24.05-25.05	виконано
5	Огляд методів вирішення задачі	26.05-27.05	виконано
6	Реалізація програмної системи	28.05-30.05	виконано
7	Написання пояснювальної записки	31.05-04.06	виконано
8	Перевірка на академічний плагіат	05.06-10.06	виконано
9	Нормоконтроль	10.06-12.06	виконано
10	Підготовка презентації та доповіді	13.06-14.06	виконано
11	Попередній захист	14.06-15.06	виконано
12	Рецензування	16.06-17.06	виконано
13	Захист перед ЕК	18.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. В’ячеслав Гребенюк
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 108 с., 25 рис., 1 табл., 1 дод., 67 джерел.

АВТОМАТИЗАЦІЯ, АГЕНТ, АУДІООБРОБКА, ЕКВАЛАЙЗЕР, ШТУЧНИЙ ІНТЕЛЕКТ, C++, DAW, DSP, JUCE, ONNX, PYTHON, VST3.

Об’єкт дослідження – процес автоматизованої обробки звукового сигналу в цифровій аудіо робочій станції.

Предмет дослідження – використання нейронних мереж прогнозування параметрів еквалайзеру на основі аудіоознак.

Мета роботи – створення VST3-плагіна еквалайзера з підтримкою автоматичного налаштування параметрів за допомогою створеної моделі штучного інтелекту.

Методи дослідження – теоретичний (збір та аналіз наукової літератури, та технологій DSP та машинного навчання), експериментальний (створення набору даних, програмна реалізація нейронної мережі та її навчання підготовленим набором даних, інтеграція моделі ONNX з JUCE). Методи розробки базуються на технологіях Python з фреймворками torch, librosa, а також C++ і фреймворком JUCE.

У результаті роботи проведено теоретичний аналіз технологій штучного інтелекту та аудіообробки. Практичним результатом є створення VST3-плагіна еквалайзера з чотирма смугами обробки, кожна з яких має параметри частоти, підсилення і добротності. Створено власний набір даних на основі порівняння сирого й референсного аудіофайлів з використанням агента для автоматичного підбору параметрів. Навчено нейронну мережу для передбачення значень параметрів еквалайзера за ознаками, які було видобуто з аудіофайлу. Модель експортовано в ONNX-форматі та інтегровано в плагін для використання в аудіо робочій станції.

ABSTRACT

Bachelor’s thesis contains: 108 pp., 25 fig., 1 tabl., 1 ann., 67 references.

AGENT, ARTIFICIAL INTELLIGENCE, AUDIO PROCESSING, AUTOMATION, C++, DAW, DSP, EQUALIZER, JUCE, ONNX, PYTHON, VST3.

The object of research is the process of automated audio signal processing in a digital audio workstation.

The subject of research is the use of neural networks for predicting equalizer parameters based on audio features.

Purpose to create a VST3 equalizer plug-in with support for automatic parameter adjustment using the created artificial intelligence model.

Research methods: theoretical (collection and analysis of scientific literature, DSP and machine learning technologies), experimental (creation of a data set, software implementation of a neural network and its training with the prepared data set, integration of the ONNX model with JUCE). The development methods are based on Python technologies with the torch and librosa frameworks, as well as C++ and the JUCE framework.

The work resulted in a theoretical analysis of artificial intelligence and audio processing technologies. The practical result is the creation of a VST3 equalizer plug-in with four processing bands, each of which has frequency, gain, and quality factor parameters. A custom data set was created based on the comparison of raw and reference audio files using an agent for automatic parameter selection. A neural network was trained to predict the values of the EQ parameters based on the features extracted from the audio file. The model was exported in ONNX format and integrated into a plug-in for use in an audio workstation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DAW – Digital Audio Workstation – цифрова аудіо робоча станція;

DQN – Deep Q-Learning Network – мережа глибокого Q-навчання;

DSP – Digital Signal Processing – цифрова обробка сигналів;

GAW – Generative Audio Workstation – генеративна аудіо робоча станція;

MIDI – Musical Instrument Digital Interface – цифровий інтерфейс музичних інструментів;

MFCC – Mel-Frequency Cepstral Coefficients – коефіцієнти мел-частотного кепстру;

ONNX – Open Neural Network Exchange – відкритий обмін нейронними мережами;

VST – Virtual Studio Technology – віртуальна студійна технологія.

ВСТУП

У сучасній музичній індустрії налаштування звуку є важливою і невід'ємною частиною, як студійної роботи, так і живих виступів. Існує багато інструментів для маніпулювання звуком і його налаштування для надання йому якості та виділення деталей, які підкреслять його художній сенс та ідентифікують композицію між іншими. Одним з ключових інструментів, який використовується для обробки звуку повсякчасно є еквалайзер. Це пристрій або програмне забезпечення, яке дозволяє вирівнювати амплітудно-частотну характеристику звукового сигналу різних частотних діапазонів [1]. Проте ефективне налаштування еквалайзера вимагає як технічних знань, так і досвіду, що часто створює складнощі для музикантів-початківців і звукорежисерів без відповідної підготовки.

Актуальність даної роботи полягає в зростаючій потребі в інтелектуальних інструментах, які могли б автоматизувати рутинні етапи обробки аудіо або допомогти новачкам у налаштуванні їхнього звуку. В умовах зростаючої уваги до різноманітних систем штучного інтелекту в житті людей і зокрема у творчих професіях, створення еквалайзеру з вбудованою моделлю машинного навчання, яка зможе запропонувати параметри для еквалізації та показати напрям, в якому можна налаштовувати звук, є досить актуальним засобом, який відповідає сучасним тенденціям розвитку ШІ і запитам саме музичної індустрії.

Ціллю даної роботи є створення VST3-плагіну еквалайзеру з підтримкою, як і ручного, що робить його самостійним, так і автоматизованого налаштування параметрів. Модель МН, інтегрована в плагін, повинна аналізувати певні ознаки аудіо та прогнозувати значення параметрів для покращення звучання. Для цього було використано відповідні сучасні технології, фреймворки та бібліотеки.

Сферою застосування такого плагіну є використання їх у цифрових звукових робочих станціях. Плагін можна використовувати автономно для ручного налаштування, для прискорення процесу еквалізації за допомогою автоматизації, а також для користувачів, які хочуть зрозуміти логіку еквалізації та недосвідчені у використанні подібних інструментів для обробки звуку.

Ідеї автоматизації налаштування звуку не є новими, але є актуальними. І зараз поява різноманітних відкритих стандартів, як ONNX та фреймворків для МН, роблять можливості для автоматизації ще більш доступними для розробки та втілення подібних проєктів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Розвиток цифрових технологій суттєво змінив підходи до створення, обробки та відтворення аудіоінформації. Сучасні цифрові аудіоробочі станції (DAW) у поєднанні з плагінами, надають звукорежисерам широкий спектр інструментів для творчої та технічної роботи із звуком. Одним із таких інструментів є еквалайзер – засіб корекції спектральних характеристик аудіосигналу, який відіграє ключову роль у налаштуванні звуку, мікшингу та майстерингу.

Водночас з розвитком машинного навчання та штучного інтелекту з’явилися нові можливості щодо автоматизації рутинних задач звукової обробки, зокрема налаштування еквалайзера. І подібні плагіни з інтегрованим ШІ вже не є не тільки рідкістю, а й дивиною. Це відкриває багато перспективних та цікавих розробок для створення інтелектуальних плагінів, які можуть адаптуватися до вхідного аудіосигналу і надавати корисні рекомендації або виконувати частину роботи замість користувача.

Цей розділ присвячено огляду предметної галузі, аналізу сучасного стану інструментів обробки звуку, а також формулюванню мети і задач даного дослідження.

1.1 Цифрова обробка звуку як складова створення музики

Цифрова обробка звуку (DSP) є ключовим компонентом сучасного музичного мистецтва та виробництва, що охоплює різні методи та технології для маніпуляції з аудіосигналом. DSP є процесом перетворення сигналів, які представлено в цифровій формі з використанням обчислювальної техніки, зокрема використовується і в музиці.

Сам процес перетворення складається з декількох основних кроків. На першому кроці необхідно перетворити аналоговий сигнал на цифровий або отримати цифровий сигнал. Якщо на вхід поступає аналоговий сигнал з

мікрофона чи електрогітари, наприклад на живих виступах чи при запису, то це відбувається за допомогою аналого-цифрового перетворювача, де сигнал перетворюється в інформацію у вигляді дискретних бітів (нуль та одиниця). Вхідний сигнал може бути і цифровим, тоді для нього перетворення не потрібне. Після нормалізації сигналу вони можуть бути оброблені комп'ютером: вони аналізуються і їх можливо обробляти за допомогою різних алгоритмів. До них можна застосовувати фільтри, стискати, еквалізувати в цифровому вигляді. У результаті, на останньому кроці отримують кілька масивів даних, які відповідають заданому обробленому сигналу або сигнал перетворюють на аналоговий знову [2], [3]. Приклад вигляду цифрового та аналогового сигналів наведено на рисунку 1.1.

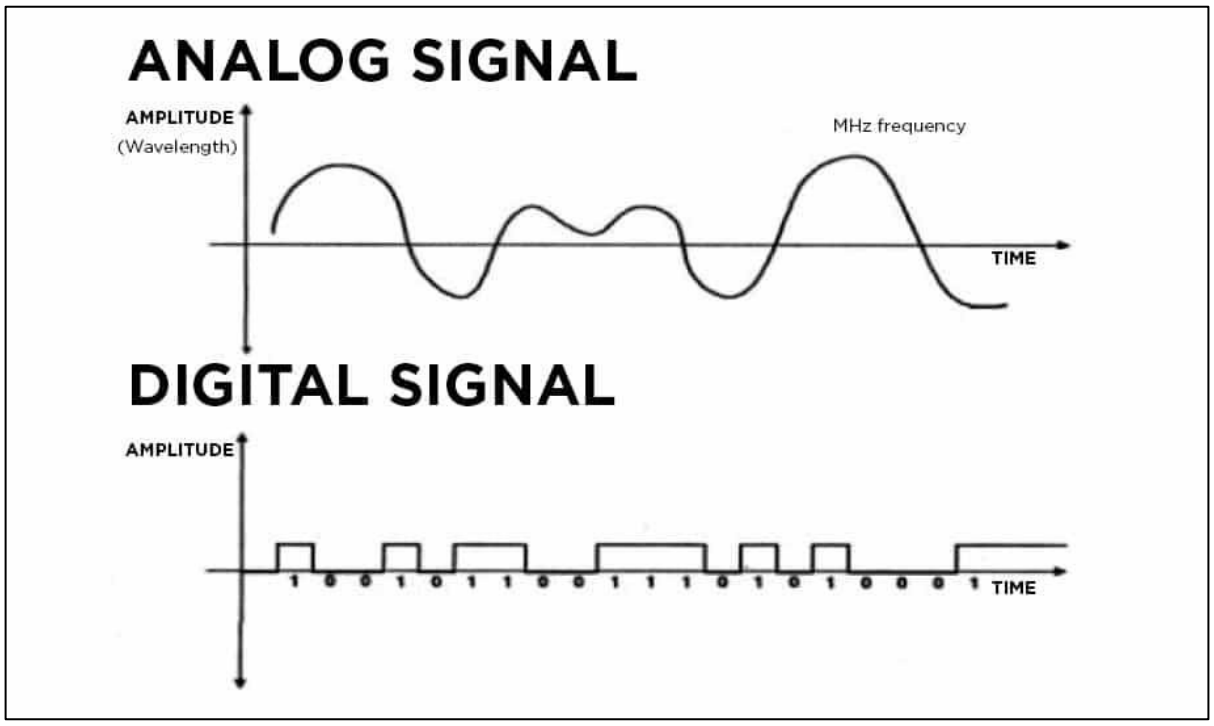

Рисунок 1.1 – Приклад цифрового та аналогового сигналів

Така обробка має безумовні переваги і перша з них це те, що її легше розповсюджувати та реалізовувати, а значить вона є більш доступною для

широкого кола людей. Аналогові системи складніші і їх переналаштування напряму пов’язане зі зміною обладнання чи компонентів. В DSP все простіше: потрібно оновити відповідний код чи програму, при чому використовувати можна і складні методи аналізу сигналу, які складні або неможливі для аналогових систем. В такий код можна інтегрувати різноманітні бібліотеки, в тому числі для роботи з ШІ. Крім того, можна досягати високої точності та стабільності фільтрації, бо фільтри, розроблені з використанням DSP, забезпечують жорсткіший контроль над точністю вихідного сигналу [4]. І останньою, суттєвою перевагою є обробка сигналів у реальному часі. При цьому, цифрова обробка сигналів є досить універсальною і використовується у аудіо, обробці зображень, телекомунікаціях, захисті даних, тощо [5].

З недоліків є те, що сигнал при дискретизації може втрачати тонкі нюанси, і звук є більш «сухішим» та можуть виникнути явища, як джиттер (небажані випадкові фазові або частотні неточності під час передачі сигналу) або шум квантування, що спотворює перетворений цифровий сигнал. Але його можна якісно обробити і на такі втрати можна погодитись, розуміючи перевагу DSP. Варто зазначити, що для роботи з DSP потрібні спеціальні знання з обробки та програмування у розробників.

DSP змінила те, як музика створюється, записується та розповсюджується, відіграючи значну роль у поширенні DAW, віртуальних інструментів та програмних плагінів для музичного виробництва. Її вплив у доступності музичного мистецтва очевидний. Бо тепер артисти усіх рівнів мають доступ до передових інструментів, які колись були винятковими для професійних студій, тим самим сприяючи різноманітному та яскравому музичному ландшафту. Зокрема, методи DSP для створення звукових ефектів, таких як еквалізація (застосування фільтрації), стиснення, реверберація (затримка), фазер, дозволяють виконувати складні маніпуляції зі звуком, і шукати власний художній почерк [6]. Незважаючи на переваги, DSP все одно критики стверджують, що залежність від технологій може

призвести до гомогенізації музики, що звук стає менш природнім в порівнянні з аналоговим, попит на який підвищився останніми роками, але це достатньо суб'єктивні речі.

1.1.1 Роль еквалайзера в цифровій обробці звуку

Еквалайзер є одним з найголовніших інструментів обробки в «арсеналі» музикантів. Він використовується в кожному проєкті незалежно від професійності людини та стилю музики. Саме він дозволяє керувати частотною характеристикою звуку завдяки можливості посилювати чи послаблювати певна частотні діапазони. Цей процес регулювання рівня певних частот називається еквалізацією. Еквалайзери використовуються і не тільки в музиці, а й в налаштуваннях голосових записів та іншого аудіоконтенту [7].

З технічної точки зору, еквалайзер є набором фільтрів, що згадувались у DSP. Ці фільтри працюють кожен у своїй частотній частині і підсилюють або зменшують амплітуду сигналу в залежності від частоти. При цьому не слід плутати тип реалізації фільтра, алгоритм та що саме він робить з сигналом, тобто його функція, форма.

У цифровій формі еквалайзери реалізуються на основі IIR-фільтру. Такі фільтри мають переваги перед FIR-фільтрами (нерекурсивний фільтр), оскільки вони використовують як поточні, так і попередні значення вихідного сигналу та теоретично мають нескінченну імпульсну характеристику, тобто їхня відповідь ніколи не «затухає». На практиці, якщо було б так, то фільтр був нестабільним, тому фактично імпульсна характеристика згасне до незначно малого рівня [8]. Кожна смуга еквалайзеру – це один такий фільтр. В обробці сигналів використовують biquad-фільтр – це рекурсивний лінійний фільтр другого порядку, що містить два полюси та два нулі. Тобто в ньому використано два попередніх вхідних та вихідних значення, що мінімізує обчислення [9]. Але є

практичним і використовується у всіх DSP-бібліотеках, як наприклад JUCE. Його першу пряму нормалізовану форму наведено у формулі 1.1.

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2], \quad (1.1)$$

де b_n – коефіцієнти, фільтра прямого зв'язку;

a_n – коефіцієнти, фільтра зворотного зв'язку.

За допомогою таких фільтрів можна реалізовувати будь яку форму фільтру, які теж в термінології звукорежисерів називають фільтрами, а саме: peak, shelf-фільтри, тощо. До того є більшість аналогових фільтрів за своєю природою є IIR-фільтрами.

Отже, еквалайзер працює з декількома фільтрами і коли аудіосигнал проходить через нього, кожен фільтр змінює амплітуду сигналу у своїй частоті чи діапазоні частот відповідно до встановлених налашутвань. Роблять вони це каскадно, тобто кожен фільтр працює над результатом попереднього. У підсумку формується новий спектр сигналу з виходів фільтрів.

Зазвичай в професійній термінології фільтром називають не реалізацію, а саме її вигляд, частотну форму. В даній роботі, увагу зосереджено тільки на основних.

Перший є піковий (дзвоноподібний) фільтр, який є надзвичайно універсальним. Його можна використовувати для точного визначення дуже вузького діапазону частот або для ширшого налаштування тонального характеру. При чому кількість частот, наскільки «дзвін» може бути широким, можна регулювати параметром Q-фактору. Підходить для різних задач, наприклад для вирізу резонансу, і використовується зазвичай в середніх діапазонах.

Піковий фільтр зображено на рисунку 1.2.

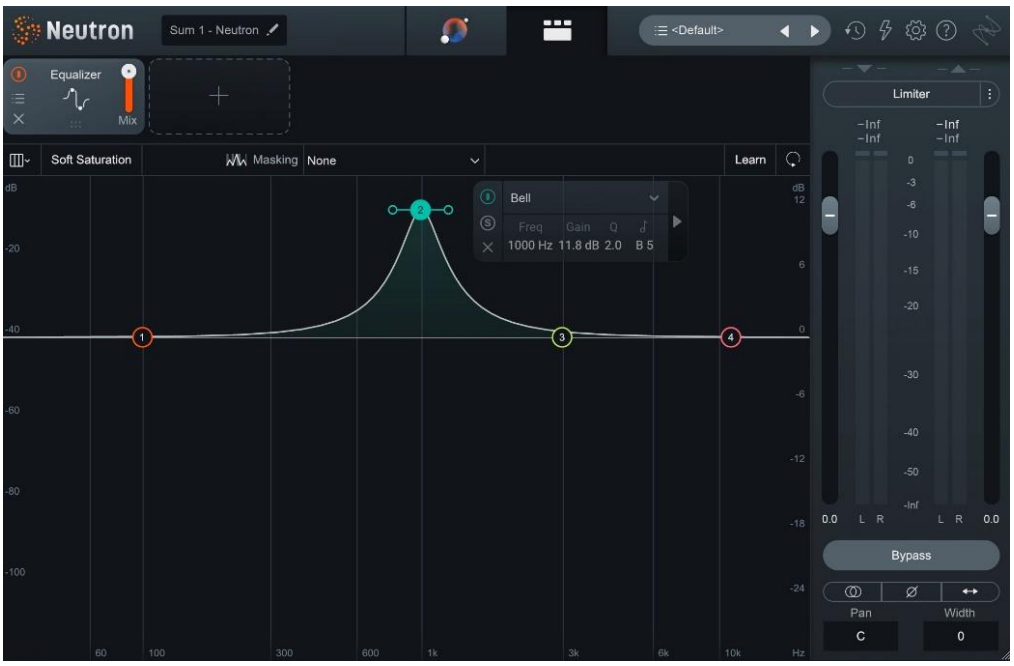

Рисунок 1.2 – Вигляд пікового фільтру

Інший фільтр називається поличковим (shelf) фільтром. Вони чудово підходять для посилення або зрізу низьких частот або, за потреби, для додавання блиску високим частотам [7]. Його зображено на рисунку 1.3.

Рисунок 1.3 – Low-shelf та high-shelf фільтри

Саме ці види фільтрів буде використано в даному проєкті, оскільки вони можуть покрити функції інших фільтрів. Наприклад, pass-фільтри відсікають цілі діапазони частот, цю задачу можуть на себе взяти і shelf-фільтр, просто значення параметрів буде відрізнятися. Також є і інші види фільтрів (смуговий, режекторний), але вони присутні не у всіх еквалайзерах.

Еквалайзери мають певний набір параметрів, що регулюють налаштування звуку, такі як:

– частота (frequency) – частота, яку потрібно, яку потрібно підсилити або зрізати, вимірюється в герцах;

– підсилення (gain) – визначає наскільки вибрані частоти обрізаються або підсилюються;

– смуга пропускання (bandwidth або Q-factor) – керує смугою пропускання, наскільки широким або вузьким буде підсилення або зрізання частот.

Також до параметрів можуть додаватися вид використаного фільтру на певній частоті.

Еквалайзери мають досить довгу історію розвитку починаючи аж з 1920 р., коли почався розвиток радіоінженерії [10]. Виділяють зазвичай два види еквалайзерів. Перший, більш старший, це графічний еквалайзер. Приклад вигляду графічного еквалайзеру наведено на рисунку 1.4.

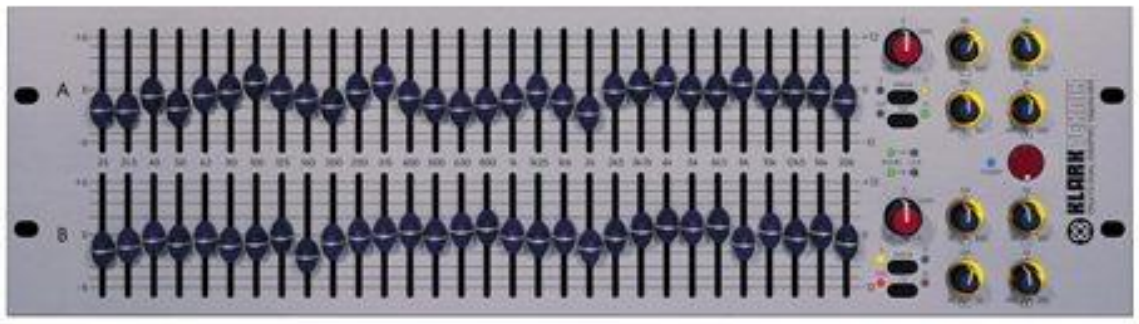

Рисунок 1.4 – Приклад графічного еквалайзеру

Він забезпечує простіший і швидший спосіб налаштування тонів у кількох частотних діапазонах одночасно. Ці еквалайзери мають повзунки, які покривають весь звуковий діапазон та дозволяють користувачам швидко налаштовувати амплітуду кожного частотного діапазону, не турбуючись про створення форм фільтрів, як це вимагав би параметричний еквалайзер. Їх часто використовують в студіях та на живих виступах.

З часом з’явилися параметричні еквалайзери, які більш тонко працюють з еквалізацією, бо на відміну від графічних, вони дозволяють керувати не тільки коефіцієнтом підсилення фільтру, але і його центральною частотою, а також шириною пропускання та мінімальним впливом на інші елементи звукового образу.

Саме такі еквалайзери використовують частіше за все у цифровій обробці, які ще можуть візуалізувати криву на частотному спектрі, їх ще називають візуальними. Приклад цифрового параметричного еквалайзеру наведено на рисунку 1.5.

Рисунок 1.5 – Цифровий параметричний еквалайзер

Також, останнім часом набули розповсюдження динамічні еквалайзери, фільтри яких реагують або на внутрішнє джерело аудіо, або на джерело вхідного сигналу сайдчейна [11].

В процесі мікшування, коли виконується балансування та налаштування різних доріжок, еквалайзер критично важливий інструмент. Бо налаштувати окрему доріжку еквалайзером не є складною задачею, а налаштувати декілька таких і поєднати в єдину композицію задача, і творча, і технічна. Частоти інструментів можуть пересікатися, що ускладнює їхнє одночасне сприйняття, тоді використовують еквалізацію.

Також він виконує і інші задачі. Ним можна коригувати частоту, підкреслювати якісь аспекти певного інструмента, можна навпаки вирізати і виправляти проблемні частоти, наприклад резонанси або різкі звуки. І ці задачі не тільки технічного характеру з обробки звуку, а й з формування тембру аудіосигналу. Можна навіть побачити цікаві техніки зі створення нових звуків або використання двох еквалайзерів одночасно, наприклад аналогового для підсилення частот та цифрового для зрізу частот.

Отже, еквалайзери пройшли багато етапів розвитку. За допомогою DSP їх реалізація стала можливою і, що не менш важливо, якісною. Це незамінний інструмент для обробки, який є не тільки технічним засобом, а й може ідентифікувати музику серед усієї іншої.

1.2 Сучасні підходи до звукової обробки

DSP змінила «правила гри» в обробці звуку зробивши гідну альтернативу аналоговим засобам. В сучасних умовах, створилося багато інструментів для створення музики, таких як DAW та різноманітні плагіни. Хоч і раніше доступ до цього мали теж тільки професіонали, зараз коли в кожного є комп’ютер, людина може створити щось для себе використовуючи ці засоби, серед яких є гідні безкоштовні альтернативи. Деякі люди стають популярними просто створивши власну студію у спальні,

як Flume [12]. При цьому навіть створивши власний плагін, якщо вона має певні знання для цього. Доступність для людей музичного інструментарію важко переоцінити. А з ним і його зручність та знижений поріг входження до користування цими інструментами. Тим не менш звукорежисери навіть зараз потребують професійної освіти, особливо в епоху розвитку технологій і зокрема ШІ.

З новими інструментами прийшли і нові техніки обробки, і полегшення старих прийомів дозволяючи складний звуковий дизайн. Наприклад, автоматизація – це техніка, яка дозволяє автоматично налаштовувати певний параметр з часом. Усі популярні DAW мають функцію кривої автоматизації, наприклад для зміни гучності в часі, одним з таких є «затухання» гучності в кінці музичної композиції. Приклад такої кривої наведено на рисунку 1.6. Автоматизація також може бути і в інших інструментах, в тому ж еквалайзері.

Рисунок 1.6 – Крива автоматизації зміни гучності

Є і багато інших цікавих ефектів автоматизації, наприклад панорамування – зміщення стереопозиції звуку з часом, завдяки якому можна створити захопливий ефект, коли джерело звуку переходить від одного динаміка до іншого [13].

Не менш важливим є якість редагування без втрат та миттєве прослуховування результатів або навіть коригування на концертах без необхідності великої кількості пристроїв, а використовуючи обчислювальну техніку. І все це забезпечує DAW та розроблені плагіни.

Завдяки цьому зросла якість в написанні музики, зокрема для фільмів та ігор. З’явилися нові технології, наприклад іммерсивного аудіо, такі як Dolby Atmos, які дозволяють створювати 3D-звукові ландшафти, забезпечуючи багатший та динамічніший досвід прослуховування [14]. Цей формат став зручним для ігор з підтримкою VR або AR для реалістичного аудіодосвіду та сприйняття джерел звуку з великою точністю [15]. Або знайшли впровадження нові техніки, як гранульований синтез для фільмів та експериментальної музики [16].

1.2.1 Цифрові аудіо робочі станції (DAW)

Завдяки з роками з’явилися і різні інструменти та ПЗ і одне з ключових – це поява звукових аудіо робочих станцій (DAW). DAW надають середовище для роботи зі звуком, аудіо та записом музики, що є по суті поєднанням традиційної звукової студії та інтерефейсу на комп’ютері. Протягом останніх двох десятиліть DAW зазнали значного прогресу, революціонізувавши процес запису. Хоча їхній основний функціонал був встановлений наприкінці 1990-х років, наступні роки зазнали значної еволюції [17]. Популярними DAW на даний момент є Reaper, Studio One, FL Studio.

На програмному рівні робоча станція реалізується як програма багатоканального зведення, що дозволяє керувати процесом запису та обробки звуку. DAW пропонують широкий спектр функцій та інструментів. Зокрема, запис аудіо та MIDI-даних з контролерів, їхнє редагування (копіювання, вставка, переміщення тощо) та мікшування – робота з багатьма доріжками одночасно (багатодоріжковий режим). Не менш важливою функцією, доступною в DAW, яка недоступна в аналоговому записі, є можливість скасувати попередню дію, використовуючи команду, подібну до команди скасування у програмі обробки текстів [18]. Зараз такі операції нікого не дивують, адже вони

існують наприклад в редакторах тексту, а раніше, в часи аналогового панування, артисти повинні були досконало знати свої партії, бо вирізати це було неможливо, а синхронізація партій вимагала професійного редактора [12]. Їх миттєве виконання економить багато часу в порівнянні з фізичним зрощуванням магнітних стрічок. Також DAW дозволяє інтегрувати різну апаратуру: айдіоінтерфейси для перетворення аналогового сигналу інструментів, MIDI-контролерів, мікшерів, тощо. Інтерфейс DAW зображено на рисунку 1.7.

Рисунок 1.7 – Приклад вигляду DAW

У минулому створення студії звукозапису вимагало значних інвестицій в обладнання та студійний простір. Сьогодні DAW дозволяє музикантам створювати високоякісні записи вдома, музичне виробництво стало доступнішим для музикантів-початківців і сприяли їхній співпраці, зокрема і дистанційно, наприклад на сервісі Splice, що є актуальним в часи пандемії COVID-19 та повномасштабного вторгнення, коли різко зросло

значення використання цифрових технологій для роботи, співпраці та комунікації між різними людьми [19].

1.2.2 Музичні плагіни

Іншим важливим інструментом цифрової обробки звуку є наявність плагінів, які підключаються до DAW. Плагін – це додаток, незалежно скомпільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення або використання її можливостей [20]. Належить до загального програмного класу додатків. Плагіни зазвичай виконуються у вигляді динамічних бібліотек.

Такими плагінами можуть бути віртуальні (цифрові) інструменти, які імітують звучання реальних музичних інструментів або створюють нові, унікальні звуки за допомогою DSP. Це можуть бути семплери, які відтворюють заздалегідь записані звуки реальних інструментів (гітара, барабани, тощо), так і синтезатори або гібридні інструменти, які генерують звук і навіть в деяких аспектах перевершують свої аналогові прототипи [21]. Такими інструментами можна керувати за допомогою MIDI-клавіатур, що є більш доступним, ніж мати багато реальних музичних інструментів. Такі інструменти стали дуже популярними у електронній музиці, оскільки там використовуються синтезатори та різноманітні ефекти, і без перебільшення стали основою електронної музики.

У цій роботі, еквалайзер виступає плагіном ефектів або FX-плагіном. Такі плагіни дозволяють змінювати, збагачувати та навіть кардинально трансформувати звучання аудіо за допомогою цифрової обробки. Вони надають обробку в реальному часі або при рендерингу. Прикладом таких плагінів ефектів є, як класичні реверберація, компресор, так і складні ефекти, як bitcrusher. Не рідкість, коли дані плагіни мають аналоговий прототип який намагаються максимально наблизити у цифровому вигляді. FX-плагіни пропонують графічний інтерфейс з елементами керування для

налаштування параметрів ефектів і дозволяють експериментувати з власним звучанням і завдяки ним оновлюються музичні стилі, один з прикладів інтерфейсу зображено на рисунку 1.8. Не дивлячись, що DAW мають вбудовані плагіни, зараз існують багато компаній, які займаються розробкою професійних плагінів з DSP, якими користуються і іменити музиканти, саундпродюсери.

Рисунок 1.8 – Приклад вигляду інтерфейсу плагіну ефектів

Музичні плагіни зазвичай використовують стандарт VST, розроблений фірмою Steinberg. Цей стандарт і дозволяє використовувати віртуальні інструменти та ефекти у DAW. Оскільки Steinberg видавали безкоштовну ліцензію фірмам на використання даної технології, вона стала основною на ринку музичного ПЗ. Відповідно до сказаного раніше, їхня робота здійснюється через хост-програму, якою виступає DAW, і дозволяє користуватись відповідним ефектом, який повертає оброблений результат.

VST3 – є третім поколінням даного стандарту, і має певні покращення, хоча кодова база з попереднім поколінням досить схожа. Серед іншого, має

динамічний розподіл введення, оновлений суфікс файлів та інші [22]. Є інші формати плагінів, такі як Audio Units (AU) від Apple чи AAX від Avid для професійних студій.

Таким чином, можна сказати, що DSP кардинально змінила сучасний саундпродакшн і створила багато корисних інструментів для роботи зі звуком у цифровому вигляді.

1.3 Актуальність використання ШІ в аудіотехнологіях

Останні роки ШІ стрімко увірвався у життя людей та всі основні професії. Він зміг полегшити та автоматизувати багато процесів або і зовсім зробити їх автоматичними. Це пришвидшило певні процеси у виробництві або знизило вимоги до працівників, оскільки їхня робота стала автоматизованою. Але виробництво має більш строгі правила або алгоритми щодо роботи на відміну від творчих галузей. І тим не менш ШІ зараз активно використовується та впроваджується і в аудіотехнологіях.

Не дивлячись на те, що можливо ШІ не так розповсюджений в аудіотехнологіях, ніж наприклад в обробці тексту чи аналізі зображень, де до речі також використовується DSP для відновлення, стиснення чи сегментації, він все одно набрав обертів. При цьому, річ йдеться не лише про задачі, пов’язані зі створенням музики, але й про інші важливі та актуальні напрямки, які потребують автоматизації або оптимізації й мають безпосередній зв’язок із обробкою звукової інформації.

Вже не перший рік дуже актуальними є інтелектуальні помічники, які можуть спілкуватись з людиною. Технологія автоматичного розпізнавання мовлення (ASR) завдяки розвитку технологій обробки природньої мови та акустичного аналізу набула широкого застосування в системах перетворення голосу на текст, а також у віртуальних асистентах, таких як Google Assistant чи Siri, щоби керувати своїми пристроями за допомогою голосових команд.

До суміжних завдань можна віднести ідентифікацію мовця – визначення особи на основі індивідуальних голосових характеристик. Таке розпізнавання може застосовуватись як у сфері безпеки (автентифікація), так і для персоналізації користувацького досвіду. Ще одним прикладом є аналіз емоційного стану мовця. Технології визначення тону та емоційної забарвленості голосу активно використовуються, зокрема, в кол-центрах для оцінки якості обслуговування клієнтів.

Одним із найяскравіших прикладів застосування сучасних технологій ШІ в галузі мовлення та аудіо є озвучування персонажів у відеоіграх класу ААА. Зокрема, поєднання технологій клонування голосу та перетворення тексту на мовлення (TTS) стало революційним кроком у розробці діалогів у 2024 році. Воно дозволяє розробникам створювати нові діалоги, які звучать реалістично та відповідають оригінальному актору озвучування, використовуючи генератор мовлення на основі ШІ або синтезатора голосу. В результаті гравці можуть насолоджуватися адаптивним сюжетом, що ще більше покращувало занурення в гру [15]. Один з лідерів такого забезпечення є Respeecher.

Це цікава технологія, бо високоякісне озвучування є дорогим і довгим, для цього потрібен актор, що може передати емоції, створить індивідуальність персонажа та ще потрібно обробити запис, щоб гарантувати найвищу якість, а ШІ змогло зробити його набагато продуктивнішим.

Іншим прикладом впровадження ШІ в аудіодизайн є розвиток адаптивних аудіосистем на основі ШІ. Ці системи можуть динамічно налаштовувати звуковий ландшафт на основі дій гравця, ігрових подій або навіть емоційних станів. Замість того, щоб покладатися на попередньо запрограмовані аудіотригери, ШІ може генерувати звукові ландшафти на льоту, реагуючи на те, як гравець переміщається в ігровому середовищі або взаємодіє з його елементами. Це призводить до більш плавного та

реактивного слухового досвіду, створюючи посилене відчуття занурення та персоналізації [23].

Не менш відомою і розповсюдженою технологією, як розпізнавання мовлення, є музичний аналіз. Наприклад, якщо людина хоче знайти мелодію чи трек, то вона її може наспівати. Сервіс від Google дозволяє таке робити, використовуючи ШІ і надає збіги з піснями в інтернеті. Зараз існує і багато інших сервісів, які спираються на зображення (спектрограми), щоб розуміти та прогнозувати музичні характеристики пісні. Вона може прогнозувати від жанру, настрою до темпу чи тональності [24]. Також можна зустріти і просто розпізнавання звуків навколишнього середовища, які ідентифікують шуми навколо. Вони можуть бути корисними наприклад, коли в палаті пацієнта чутно звуки падіння, що можуть свідчити про необхідність надання допомоги [25].

Найбільшу увагу прикуто до сервісів пов’язаних зі створенням музики, семплів, звуків, тощо. В медіапросторі існує багато прикладів генерації музики для обходу авторського права або створення музики під контент та навіть з власним текстом.

Першими є інструменти просто для створення музичних ідей. До прикладу BandLab Song Starter, який генерую ідею і її можна одразу експортувати у студійний режим. Це є зручним для початківців, які хочуть з чогось почати. Але є і такі сервіси, які пишуть музику і для більш відомих виконавців, такі як SOUNDRAW. Він може повністю створити «мінус» і навіть поєднувати жанри, особливо актуально в жанрах «хіп-хоп», оскільки музика менш розвинена і більше зосереджена на ритміці. Її можна редагувати в самому серівісі, але назвати це DAW буде перебільшення, їм бракує елементів керування для професійних звукорежисерів. Для аматорів такий сервіс дійсно є непоганим. Плюсом саме даного сервісу є тренування нейронної мережі на оригінальних семплах створених людьми саме для неї, що прибирає питання щодо авторських прав, яке є актуальним. Наприклад, коли LLM навчаються на купах тексту, статей відомих видавництв без

дозволу останніх, що знецінює роботу цих людей, прикладом став судовий позов New York Times до OpenAI [26].

Щоб покращити досвід користування для професійних музикантів, почали впроваджувати ШІ в самі DAW через плагіни. А новинкою стали появи GAW – генеративних аудіо робочих станцій, які інтегрують можливості використання повноцінної DAW та інструментів з ШІ. Прикладами таких є ACE Studio, яка спеціалізується на синтезі вокалу з ШІ і навіть навчання власних моделей. Але її функціонал стало можливим інтегрувати у власну DAW. Іншим прикладом GAW є один із перших новаторів у генерації музики – AIVA. Користувач може згенерувати композицію на основі стилю і працювати з нею через веб-сайт, але для повного доступу функціоналу, як у DAW потрібно завантажити відповідний додаток [27]. Новим досвідом стало впровадження чат-ботів у самі DAW, прикладом такого є чат-бот Gopher вбудований у FL Studio. Цей чат бот може надавати певні рекомендації щодо музичних питань. Інші чат-боти можуть навіть переглядати DAW, слухати аудіофайли та на їх основі надавати рекомендації щодо налаштувань ефектів, прикладом такого є Strip 2 [28].

Найцікавішим проєктом була WavTool. Це також GAW, яка містила чат-бот під назвою Conductor на базі GPT-4 і працювала в веб-браузері. Цей чат-бот міг розмовляти на будь-яку музичну тему і мала пряму здатність виконувати дії в DAW, наприклад генерувати MIDI-файли, а потім і семпли, синтезувати нові музичні інструменти, керувати ефектами (тим самим еквалайзером, компресією) та запускати дії при спілкуванні з ботом. При цьому функціонал цього сервісу постійно розширювався, сервіс був повноцінним інструментом та працював з хмарним збереженням, конвертував аудіо в MIDI і ще багато іншого. Нажаль, цей проєкт в кінці 2024 року, але з повідомленням, що продукт повернеться [29]. За допомогою таких чат-ботів або VST-ШІ плагінів, по суті, можна перетворити DAW у GAW.

Вище перелічені інструменти існують не тільки в контексті GAW, бо як правило це все окремі модулі. Цей функціонал можна знайти і в окремих VST-ШІ плагінах. Це і перетворення аудіо в MIDI (Samplab 2), і синтез нових структур та звуків (Combulator), генерація MIDI, тощо.

Деякі з таких інструментів можуть вже аналізувати стиль роботи композитора, щоб створювати на їх основі музику. Наприклад, грув-машини для створення бітів чи ударних можуть проаналізувати семпли користувача, генерувати їх у різних стилях та підтримують рандомізацію, щоб зробити їх більш унікальними [30].

Для створення нових звуків розроблені синтезатори з ШІ. Наприклад NSynth Super, який є частиною експерименту Magenta. Він використовує глибоку нейронну мережу для вивчення характеристик звуків, а потім створює абсолютно новий звук на основі цих характеристик. На таких звуках і була натренована модель. За словами розробників можна отримати звук, який є одночасно і флейтою, і ситаром, що дійсно дивує [31]. І це далеко не один такий приклад, є й інші подібні синтезатори, наприклад Pigment від однієї з ключових компаній в даній області Arturia.

Можна заключити, що ШІ значно розширили діапазон роботи та створення музики, вдосконаливши поточні інструменти, як і DAW та плагіни. В майбутньому, коли плагіни почнуть демонструвати значне зростання, застарілі DAW, ймовірно, почнуть неминуче переходити до нативних генеративних аудіофункцій. На цьому етапі VST-ланцюжки можуть стати менш поширеними, а більшість важливих дій будуть доступні безпосередньо в GAW [29].

Переходячи зі створення музики та звуків за допомогою ШІ, окремо буде розглянуто безпосередньо обробку звуку в плагінах, мікшингу, його якості та очищення, бо ця галузь є не менш важливою та великою для огляду.

Досить широко розповсюджені технології з обробки аудіо та видалення шуму, як активне, так і пасивне. Такі технології вже не рідкість і

почали застосовуватись в голосових дзвінках та відеоконференціях. Воно зменшує або усуває небажані звукові сигнали. Працює це шляхом аналізу та розрізнення бажаних сигналів, таких як мова, та небажаного шуму в аудіосигналі. Потім застосовуються алгоритми для мінімізації або видалення шуму, зберігаючи при цьому якість бажаного сигналу [32]. В даних методах використовуються як статичні методи, так і глибокі нейромережі. Їх перевагою є загальне покращення якості звуку, незалежно від аудіопристрою та немає необхідності у ручному налаштуванні. Звичайно, що шумоподавлення використовується і у музиці, щоб очищати запис, юо при записі можуть виникнути шуми. Ще у 2014 році був представлений багатоканальний шумоподавлювач діалогів, який використовував алгоритм LEARN. Зараз таку технологію впроваджують і інші великі компанії у плагінах, як iZotope [33]. Іншим не менш цікавим застосуванням є аудіореконструкція – відновлення старих або пошкоджених записів, де використовують різні методи, зокрема і видалення небажаних шумів.

Також ШІ впроваджують, як було сказано вище, у генерацію мовлення і зокрема у написанні пісень. Зараз часто використовується ШІ саме в аудіо, який не тільки може розмовляти, а і співати. Вони можуть визначати фальшиві ноти і повністю генерувати вокал, що використовується навіть у створенні медіа контенту, наприклад Девід Гетта на відео грав пісню з куплетами Емінема, згенерованими ШІ [34]. Це може замінити в якійсь задачі вокаліста, бо вокал досить реалістичний і має перевагу – його легко редагувати та маніпулювати нотами. Однак, це не замінює людину повністю, хоча це дешево та швидко.

Не дивлячись, що у деяких плагінах ця функція вбудована, але плагіни для розділення інструментів і отримання стемів (окремих треків запису вокалу та інструментів). Ці інструменти використовують глибоке навчання для розділення повністю зведеної пісні на окремі компоненти, такі як вокал, ударні, бас, гітара, фортепіано тощо і є дуже корисними у процесі

виробництва музики. Його можна по-різному використовувати, наприклад для створення реміксів, семплів для своєї музики, аудіореставрація, якщо потрібно покращити якість окремих інструментів, для живих виступів, оскільки забезпечують чудовий контроль над інструментами та навіть для мастерингу. Зокрема, такі функції надають онлайн-сервіси, як LALAL [30].

Окрема велика категорія, яка існує для обробки звуку це плагіни ШІ, які призначені для мікшингу та майстерингу. Ці процеси характеризуються складністю обробки всього треку та великими витратами часу, хоча для деяких проєктів можна використовувати і тривіальні рішення. Щоб зробити цей процес швидшим почали створюватися такі автоматизовані плагіни. Мастеринг відрізняється від мікшингу тим, що мастеринг проводиться на фінальному етапі після мікшингу. Процес мікшингу характеризується поєднанням та балансуванням всіх окремих треків, щоб створити цілісну музику. Це включає панорамування, еквалізацію, компресію, тощо для кожного треку окремо або групи треків. Мастеринг відбувається на остаточному міксі, щоб відшліфувати його та підготувати до випуску. Тож, ці процеси прекрасно підходять під задачу автоматизації. Вони навчені на тисячах професійно зведених треках і аналізують частотний та динамічний баланси, гучність і відповідно до цього вносить розумні корективи у значення відповідних ефектів. При цьому все це значно пришвидшує трудомісткі процеси, і зекономлений час можна витратити на інші, більш творчі завдання. Щодо автоматизованості, то запропоновані ШІ налаштування можна змінити щоб домогтися результату, який подобається саме звукорежисеру, таким чином зберігаючи творчий контроль. Найбільш відомими є iZotope зі своїми плагінами для майстерингу (iZotope Ozone) та мікшування (iZotope Neutron) окремо. Вони використовують «Assistant» для аналізу треку і пропонують налаштування для інструментів, які в них вбудовані і які також можна регулювати після обробки. Цікавим є сервіс Landr для мастерингу, в який можна додавати референсні треки та вибирати між різними стилями та рівнями гучності, а також проводити A/B-аналіз між

оригінальним та майстер-треками з однаковою гучністю. Є також і просто набори плагінів ШІ, такі як Focusrite Fast Bundle, кожен з яких вносить відповідні корективи відповідно до треку. Включає лімітер, еквалайзер, компресор, реверберацію та «Reveal» – спектральний сайдчейн [35].

Підводячи підсумки щодо впровадження ШІ в аудіообробку та взагалі в те, що пов'язане з цифровим звуком, можна сказати про велику різноманітність вирішених та оптимізованих задач. Від генерації звуків до GAW – інновації, які впроваджують ШІ показують не лише автоматизацію певних рутинних або трудомістких завдань, а й розширення меж творчого використання таких інструментів. І таких проєктів вже досить багато, а в майбутньому буде ще більше, бо ця задача має відмінну специфіку від того ж комп'ютерного зору і розвиток МН сам сприяє створенню подібних інструментів, моделей. Це актуально і потрібно всім користувачам, які можуть не здогадуватись, що при прослуховуванні певного музичного продукту, певні елементи вже зроблені за допомогою ШІ. І ці інструменти є доступними та корисними кожній категорії людей, що займаються музикою: від новачків, яким потрібна допомога, щоб досягти якісного звучання без великої кількості знань чи коли вони не можуть самостійно повністю створити музику, до професіоналів, яким важлива швидкість роботи і це робить їх більш продуктивними та залишає час на інші творчі аспекти та залишає кінцевий результат все одно за ними.

Виходячи з вищесказаного, даний проєкт еквалайзеру слідує сучасним трендам автоматизації і реалізує інтелектуального асистента для полегшення процесу мікшування. Автоматичне налаштування такого інструмента є корисним також для всіх категорій людей, що займаються музикою, як для зменшення навантаження професійних звукорежисерів, так і для підвищення якості звучання у новачків. Впровадження ШІ в такий традиційний, але критично важливий ефект як еквалайзер, є прикладом практичного використання ШІ і враховуючи активний перехід DAW до

інтеграції з ШІ, подібні інтелектуальні плагіни є кроком у напрямку нової епохи цифрового аудіовиробництва і популяризації GAW.

1.4 Постановка задачі

У результаті проведеного аналізу сучасного стану в галузі DSP, а також вивчення тенденцій застосування методів ШІ, можна сказати, що дана задача є актуальною, бо в музичному продакшні все більше уваги приділяється автоматизації. Зокрема, еквалізація – один із ключових етапів обробки та налаштування звуку, часто вимагає від музиканта високого рівня досвіду, знань в даній області, бо це досить складний і фундаментальний процес, який завжди присутній у створенні музики, мікшингу чи майстерингу. Незважаючи на велику кількість доступних еквалайзерів, кожен з яких має широкий набір параметрів і функцій, налаштування залишаються суб’єктивними. У зв’язку з цим постає актуальна задача створення інтелектуального еквалайзеру, який не замінює, але доповнює процес еквалізації, виступаючи у ролі помічника для людини. Такий інструмент повинен мати здатність аналізувати звуковий сигнал та рекомендувати значення параметрів еквалайзера на основі професійно оброблених треків. Важливо, щоб цей інструмент зберігав творчий контроль за користувачем, але при цьому спрощував технічну рутину й підвищував продуктивність роботи.

У якості технологічної основи для реалізації такого рішення доцільно використати формат VST3, що є загальноприйнятим стандартом для розробки аудіоплагінів, який забезпечує інтеграцію з DAW. Крім того, у світлі останніх досягнень в галузі МН та глибоких нейронних мереж, логічно використати нейромережеву модель, здатну прогнозувати параметри еквалайзера на основі аудіоознак.

Перелік задач, які необхідно вирішити для досягнення мети виглядає наступним чином:

– проаналізувати наукові джерела з тематики DSP та застосування ШІ в аудіотехнологіях;

– описати архітектуру програмного рішення, визначити структуру VST3-плагіна;

– розробити методику формування набору даних або знайти набір, що відповідає досягненню мети;

– вибрати і налаштувати архітектуру нейронної мережі для прогнозування параметрів еквалайзера;

– навчити модель на наборі даних та протестувати її;

– експортувати натреновану нейромережу в ONNX-формат для подальшої інтеграції в плагін;

– реалізувати в плагіні механізм отримання аудіоознаок;

– провести практичне тестування створеного плагіна.

Таким чином, мета кваліфікаційної роботи полягає у розробці VST3-плагіна інтелектуального еквалайзера, який виконує аналіз аудіосигналу та надає рекомендації щодо налаштувань еквалайзера на основі попередньо навченого ШІ. При цьому передбачається, що еквалайзер матиме стандартну архітектуру з чотирма фільтрами та можливістю ручного налаштування, а автоматизоване налаштування параметрів буде реалізовано на основі треку, що оброблюється та рекомендації перametрів, які надає попередньо навчена нейромережа.

2 МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ

Розробка плагіну еквалайзера є досить комплексною задачею та складною, оскільки поєднує в собі елементи DSP, розробку плагіну і методи ШІ та МН. Даний розділ розкриває сутність задачі і розбиває її аналіз на етапи, які будуть реалізовані в ході виконання кваліфікаційної роботи. Будуть розглянуті основні підходи до вирішення поставленої задачі, їх порівняння між собою, які є переваги та недоліки, і обґрунтовано вибір методу та інструментів що були розглянуті у процесі розробки.

2.1 Огляд аналогічних систем

Незважаючи на відносну новизну підходу, на ринку вже представлено певну кількість, як плагінів, так і платформ, які представляють собою еквалайзер або комплексне налаштування звуку включно з еквалайзером. При чому деякі орієнтуються на роботу з музичним продуктом, а деякі для роботи з аудіоконтентом, подкастами, надаючи трохи інші можливості, але також з еквалізаціэю, що зайвий раз підкреслює – цей процес є невід’ємним при роботі зі звуком загалом.

Одними з популярних інтелектуальних систем є iZotope Neutron та iZotope Ozon, які згадувалися у підрозділі 1.3. Вони позиціонуються, як розумний мікс-асистент і діють, як канальні смуги, що містять окремі модулі (еквалайзери, компресори тощо) для обробки аудіо. На базі ШІ, плагін здатний проаналізувати вхідний звук і встановити ланцюжок ефектів. При цьому ці допоміжні робочі процеси не призначені для того, щоб «зводити трек за людину», а радше скорочують час сеансу мікшування чи мастерингу, пропонуючи чудову відправну точку [36].

Хоч про технічну реалізацію сказано замало, але даний плагін працює на концепції визначення цілі, яку повинен досягнути інтелектуальний асистент. Після аналізу звуку, помічник відобразить цільову криву

тонального балансу, яка представляє частотну та динамічну інформацію, рекомендовану Neutron для треку (ціль – це туманна помаранчева лінія, а записане аудіо – це чітко визначена біла лінія), що зображено на рисунку 2.1.

Рисунок 2.1 – Інтелектуальний асистент Neutron

Щодо визначення цілі, то користувач може налаштувати її іншими елементами керування, якщо йому не подобається. При цьому ці плагіни користуються бібліотеками цілей, яка містить проаналізовані ШІ профілі, які показують до якого звуку потрібно йти, або можна завантажувати користувацькі для вказання стилю автора. До цього постачається окремий асистент для регулювання балансу всіх інструментів, теж за допомогою помічника.

Розробники стверджують, що Mix Assistant ґрунтується на припущенні, що аранжування доріжок у більшості сесій можна об’єднати в однакові групи, навіть якщо деякі з них порожні [37]. Він використовує

технологію МН, щоб автоматично призначити кожну доріжку до групи. В ході експерименту було з’ясовано, як кожна з цих груп повинна виглядати в міксі. І хоча завжди будуть приклади, що відрізняються від норми, розроблений ШІ здебільшого відповідав очікуванням. Як і у випадку з усіма допоміжними технологіями, функція балансу Mix Assistant призначена для того, щоб дати відправну точку, автоматизувати процес і дозволяє вносити зміни в його пропозиції.

Іншим достатньо відомим прикладом є Sonible smart:EQ, який є виключно еквалайзером, але який пропонує досить складні функції. Він почав впроваджувати крос-канальне мікшування і теж аналізує декілька доріжок, враховуючи контекст міксу. Центральним елементом є фільтр smart:filter – він автоматично балансує сигнал на основі цільового профілю, який можна обрати для треку. Він також має різні функції для налаштування обробки саме ШІ, такі як: «Smoothing» для більш м’якої фільтрації, опцію «Adaptive» для згладжування високодинамічних сигналів. Також можна обмежити частотний діапазон, щоб визначити, де повинні працювати інтелектуальні алгоритми та розділити інструменти на групи в треку за бажанням користувача (наприклад, за важливістю). Еквалізація може відбуватися лише в поточному треку, або всіх інших треків під заданий у групі, або комбінувати підходи. Має можливість налаштування під жанр або навіть завантажити референсний трек, якого звучання хочеться досягнути [38]. Інформації щодо деталей алгоритмів чи даних, які використовувались для розробки ШІ в ході аналізу виявлено не було.

Ще одним найбільш популярним еквалайзером є Focusrite FAST Equaliser. Він також використовує ШІ, знаходячи правильну криву та активні зони треку і налаштовуючи їх. В ньому можна обрати профіль інструмента і налаштувати еквалайзер до нього. Після аналізу спектрального вмісту та створення кривої еквалізації, є можливість до більш тонкого налаштування, обираючи тональний характер. Як і в інших плагінах, більш конкретної інформації щодо реалізації немає.

Щодо веб-сервісів з мастерингу, які використовують еквалізацію, то вони є скоріш рішенням для людей, які не мають великого відношення до музики. Також в цих сервісах іноді немає навіть мінімального графічного інтерфейсу для управління еквалізацією. Тим не менш, еквалайзер завжди знаходив активне розповсюдження, навіть в речах непов’язаних з музикою. У період розповсюдження аналогових аудіопристроїв, багато з них мали вбудовані еквалайзери. Зараз багато сервісів для створення медіаконтенту також мають еквалізацію для покращення розбірливості тексту та чіткості мовлення, що говорить. Прикладами таких сервісів є Descript або Auphonic, але деталей реалізації також в ході аналізу знайдено не було.

Отже, можна сказати, що еквалайзери з ШІ не рідкість і є досить гарні екземпляри. Кожен з них обов’язково має можливість для ручного налаштування людиною, і до застосування ШІ, і після нього. Кожен має вбудовану модель ШІ, при чому наголошено, що такі плагіни мають більше навантаження на систему та CPU, що логічно. Моделі навчені на великих об’ємах даних, професійно оброблених, референсних треках. Не зазначається, які саме записи було використано і чи не було використано чужу інтелектуальну власність для тренування цих моделей. Також не зазначається, які саме ознаки аудіоданих використані для аналізу та для передачі на вхід моделі. Декілька разів, на сторінках відповідної продукції, можна було побачити слово «спектральні», що вказує на характеристики аудіосигналу, які описують його частотний склад. Це може бути досить широкий набір характеристик, від мел-спектрограм до спектрального центроїда чи нахилу. Також можна сказати, що моделі можуть адаптувати трек на основі попередніх тренувальних даних, на основі вказаного жанру, що може свідчити про те, що використовуються різні моделі для кожного основного жанру, або на основі референсного треку, що адаптує дані і параметри налаштувань еквалайзера до стилю налаштування звуку автора. Деякі мають певні конкретні налаштування під настрій, інструменти чи інші

аспекти. Це виокремлені основні моменти, які будуть розкрито в ході аналізу окремих етапів вирішення задачі.

2.2 Підхід до реалізації плагіну

Оскільки задача зі створення плагіну є специфічною, якісних інструментів для його створення є не дуже багато. Основними вимогами для розробки будуть саме якість обробки звуку, можливість створення графічного інтерфейсу та інтеграція моделі ШІ в плагін.

Для забезпечення безперешкодної роботи з більшістю DAW, буде використано стандарт VST3. Даний формат зручний тим, що він найпоширеніший і його підтримують всі відомі DAW. Інші формати не розглядаються, оскільки розробка буде відбуватись на платформі Windows, тому формат Audio Units розроблений Apple може не підійти для тестування, як і інші формати. При чому VST3 найостанніший стандарт, який має покращення в порівнянні з попередніми версіями, тому стандарти VST2 також не будуть використовуватись для розробки. Тому при створенні і вибору технологій важливо, щоб мова та фреймворк при збірці підтримували збірку у формат VST3.

Існує достатньо багато фреймворків для створення плагінів. В більшості всі вони базуються на мові C++, бо офіційний VST3 SDK, наданий Steinberg, реалізований як інтерфейс даної мови. З цього можна і почати, що плагіни розробляються на основі офіційного SDK від Steinberg, і більшість бібліотек також реінтегрують його в себе. Саме тому він не підходить для повноцінної розробки, більшість інструментів доведеться створювати вручну, а VSTGUI запропоноване для створення інтерфейсу досить складне і є нетривіальним процесом, бо він дає тільки стартову точку. Так само він не реалізує алгоритми DSP, це реалізовано в інших фреймворках. Саме тому було створено інші більш зручні фреймворки, але на основі цього ядра. Також існує VST3 Project Generator, яка є просто

утилітою для створення проєкту плагіна з VSTGUI, але він просто полегшує старт проєкту.

Одним з найновіших і підтримуваних фреймворків є iPlug 2, який є переробкою більш раннього проєкту з додаванням графіки. Він є досить цікавим фреймворком, який ще в розробці, тому документація досить невелика, має тільки описи методів. Але проєкт підтримує всі основні формати плагінів і його можна використовувати для розробки веб-додатків та окремих програм для різних ОС. В цілому гарний варіант для розробки, але потребує окремого пошуку реалізацій методів DSP в інших фреймворках, які за словами розробників повинні гарно інтегруватися з даним фреймворком. Тим не менш, не є розповсюдженим і має досить неповну документацію, хоч і є перспективним [39].

Іншим фреймворком є RackAFX, розроблений Вілом Пірклом – доцентом кафедри музичної інженерії в Університеті Маямі та викладачем цифрових та аналогових аудіотехнологій. Він має також гарні видання щодо проєктування плагінів ефектів, які розроблено на RackAFX і є досить вичерпними. Підтримує як і можливість створення гарного інтерфейсу, так і підтримку MIDI, різних форматів плагінів, тощо. Ця розробка використовувалась для навчання студентів паралельно з впровадженням методів DSP на чистому C++ і відповідних виданнях. Однак зараз проєкт повернувся до приватної розробки і не підтримується публічно, та не підтримує деякі середовища розробки. Але з точки зору популяризації створення плагінів це є одним з найкращих джерел [40].

Наразі найпоширенішим є фреймворк JUCE для створення плагінів, який також підтримує всі доступні формати для збірки, в тому числі VST3. На відміну від більшості інших фреймворків, він має образу вбудовані методи DSP для розробки, що полегшує розробку і немає потреби шукати інші бібліотеки, можна одразу працювати. Його широко використовують й інші великі компанії, такі як Adobe, Izotope, LANDR, Netflix та інші. Він є кросплатформним, що досить зручно для більшості розробок і за допомогою

його можна робити і окремі застосунки на Windows, MacOS та інших ОС. Окрім достатньо потужної бібліотеки з DSP, яка дозволяє розгортати різні проєкти одразу, в ній можна зробити і гарний GUI, який можна зробити якісним з усіма задумами розробників, крім того має підтримку і OpenGL. Цей фреймворк має багато компонентів для розробки, може працювати з MIDI.

Одна з найважливіших речей це гарна документація, та те що фреймворк перевірений часом і має досить багато проєктів, шаблонів і відкритого коду інших розробників. Є окремий великий форум, де розробники діляться досвідом чи задають питання і є досить активним, а також спонсорує «Конференцію розробників аудіо». Оскільки його використовують і iZotop, можна зробити висновок, що в нього можливо інтегрувати ШІ. Також фреймворк має і гарний проєктний генератор у різних IDE та певні шаблони проєктів, які можна протестувати. Наразі це є одним з найдоступніших і підтримуваних відкритих фреймворків і має ліцензію, як для безкоштовного розповсюдження, так і для продуктів, які є закритими і комерційними [41].

Є також і певні інші фреймворки, наприклад Jamba – легкий VST3 фреймворк, але по суті він не надає унікальних інструментів, яких не існувало би в інших фреймворках [42]. Або VST.NET, який працює на платформі .NET, але який має можливості збірки тільки плагінів VST2 формату, чи AudioPlugSharp, який немає достатньої підтримки і не має реальних проєктів використання. Обидва реалізовані мовою C#. Також з інших мов на яких програмують плагіни можна зустріти Rust з бібліотекою nih-plug, яка навіть має приклади використання.

Одними з найцікавіших рішень є мова програмування Faust та середовище візуального програмування plugdata. Faust – це функціональна мова програмування для синтезу звуку та обробки аудіо з сильним акцентом на проектуванні синтезаторів, музичних інструментів, аудіоефектів тощо, створена в дослідницькому відділі GRAME-CNCM, яка має власний

компілятор і навіть веб-середовище розробки, що зараз достатньо актуально. За словами розробників мова дозволяє «перекладати» будь-яку специфікацію DSP Faust на широкий спектр неспецифічних для предметної галузі мов, таких як: C, JAVA, LLVM IR, WebAssembly тощо. У цьому відношенні її можна розглядати як альтернативу C++, але вона набагато простіша та інтуїтивно зрозуміліша у вивченні [43]. При цьому ця мова має і комерційне використання при чому реальних проєктів достатньо багато. Але мова недостатньо відома і має функціональну парадигму, що може ускладнювати поріг входження.

Plugdata – це середовище візуального програмування для прототипування та навчання. Це середовище також інтегрується з JUCE, Pure Data та іншими. Є кросплатформним та його можна інтегрувати у DAW. Це цікавий інструмент для людей, які не вміють програмувати, але знають будову мікросхем та мають технічну базу в електроніці [44]. Їх проблема в тому, що не зрозуміло чи можливо в них інтегрувати ШІ, тому ці засоби потребують додаткового аналізу.

Проблема сучасних фреймворків така, що не кожен має гарну документацію чи набір реально розроблених, сучасних проєктів. При цьому майже всі мають підтримку VST3, що вказує на поширеність стандарту. Безумовно можна стверджувати, що JUCE один з найбільш практичних фреймворків, який має приклад розробки на нього, а також велику кількість послідовників. Він зайняв досить міцну позицію при виборі технологій програмування плагінів, і це не дивно, бо в ньому є все, щоб створити плагін від початку і до кінця, як і допоміжні матеріали, які можна зустріти або в документації, або на популярних форумах. Тому вибір JUCE для розробки є гарним рішенням для проєктування плагіну, хоча і потребує достатньо гарних знань мови C++ та DSP, але реалізація цього модулю на достатньому рівні, щоб не звертатися до інших фреймворків, які займаються обробкою звуку.

2.3 Генерація набору даних

Достатньо складним етапом при вирішенні задач пов’язаних з музикою, її генерацією або обробкою є пошук відповідного набору даних. Бо існують багато способів описати звук різними ознаками, які обираються безпосередньо під задачу. Саме тому далеко не завжди вдається знайти відповідний набір даних.

В більшості задач, як основу беруть публічні аудіодатасети, які містять в собі певні музичні уривки та іноді позначені темп, тональність чи інші характеристики музики для вирішення певної задачі, наприклад класифікації жанру, тощо. Відомі також і інші види представлення аудіофайлів, наприклад MIDI-файли, як у наборі даних MAESTRO, які використовують для генерації музики або не менш ефективний і зручний спосіб представлення ABC-notation, який був нещодавно був використаний для тренування моделі для генерації нот у моделі NotaGen [45]. Існують також і інші способи представлення аудіофайлів, які наприклад використовувались для створення музики для ранніх ігор і має назву Music Macro Language. Також, були експерименти з представленням музики у форматі схожому до XML, який називався MML (Music Markup Language). І хоч ці способи достатньо непогані для машинного представлення музики, вони не набули широкого використання у МН. Тому знайти дані для даного плагіну достатньо складно, оскільки можливо знайти дані, які би містили в собі аудіоознаки певних записів, але вони точно не містять потрібних значень, які використовуються в еквалайзері, як частота підсилення та добротність та які би вказували, що вони покращують запис і знайдені професійними звукорежисерами під час мікшування. Тож, можна зробити висновок, що під конкретну задачу з налаштування параметрів еквалайзеру набору даних немає і його потрібно розробляти окремо.

У попередньому розділі при огляді музичних систем з ШІ згадувався такий проєкт, як NSynth. Не дивлячись на те, що це віртуальний

інструмент (синтезатор), який використовує нейронну мережу для вивчення характеристик звуків, а потім створює абсолютно новий звук на основі цих характеристик, цей проєкт задіяв метод генерації нових звуків для тренування моделі, тобто створення власного набору даних. Було записано 16 оригінальних звуків у діапазоні 15 тональностей, а потім введено в алгоритм NSynth для попереднього обчислення нових звуків [31]. Іншим прикладом генерації даних в роботі зі звуковими сигналами є робота щодо дослідження моделі МН для автоматичного застосування ефекту реверберації до аудіосигналу [46]. В даному дослідженні також існувала проблема створення великої вибірки даних імпульсних характеристик приміщення (RIR), оскільки це займає багато часу, але необхідно було зберегти автентичність даних імпульсів. Для цього було записано імпульси в реальній кімнаті та змодельовані у віртуальній кімнаті. Після створення змодельованих імпульсів вони порівнювалися з реальними імпульсами для обрахунку оптимальних коефіцієнтів поглинання кімнати.

Таким чином було знайдено оптимальні коефіцієнти, а віртуальні імпульси були дійсно наближені до реальних. Це дозволило згенерувати набір даних для дослідження, який є достатньо наближеним до реального з економією часу. Тобто метод генерації даних є досить універсальним, оскільки долає проблему малої кількості даних та дозволяє створювати саме потрібні дані для моделі, а не шукати їх в репозиторіях, обирати потрібні та очищати їх. До того ж у випадках, коли даних не існує взагалі, а для їх ручного створення потрібен людський ресурс, у випадку даного проєкту звукорежисери, такий вихід є дуже вигідним.

Складність такого методу полягає лише у наближенні даних до реальних, що потрібно перевірити наприклад різницею між реальними прикладами даних та згенерованими, щоб мінімізувати різницю. Отже, для даної роботи буде задіяно саме створення даних за допомогою моделі ШІ, і тренування фінальної моделі для передбачення параметрів еквалайзеру.

Не рідкість, коли можна помітити в еквалайзерах, що вони можуть застосовувати референсні треки користувача для тренування і наближення мікшування до його стилю. Такі приклади еквалайзерів розглядались у попередньому розділі, такі як Landr та Sonible smart:EQ. Ці еквалайзери мають таку можливість, але не розкривають деталей реалізації. Можна сказати, що як і в попередній згаданій роботі з реверберацією, йде порівняння референсного сигналу з вхідним, і на його основі будується стиль.

Не дивлячись на те, що деталей немає, але є достатньо нова робота, яка також розглядає перетворення стилю мікшування вхідної доріжки на стиль мікшування еталонної пісні [47]. Це робиться за допомогою попередньо навченого кодера, який витягує інформацію, пов’язану з аудіоефектами еталонного музичного запису. Витягнувши потрібну інформацію щодо стилю мікшування, він передається вхідному аудіо, а навчання проводиться шляхом різниці між двома записами. З чого можна зробити висновок, що навчання моделей шляхом різниці записів між еталонним та вхідним (сирим) є відомою практикою.

Об’єднавши ідеї щодо генерації даних та порівнянні еталонного та сирого треків, можна вивести загальну ідею генерації даних для роботи, а не їх пошуку, в якій оптимальні параметри еквалайзера, які потрібні для навчання вихідної моделі, можна отримати шляхом порівняння еталонного та сирого треків. Таким чином можна буде шляхом підбору параметрів еквалайзеру досягти максимально оптимальних значень, щоб наблизитись до досвіду мікшування інших людей. Для досягнення такої мети потрібна модель, яка би змогла реконструювати перетворення запису.

Підходи можуть бути різні, як і подання аудіотреку до моделі ШІ. Наприклад, аудіосигнал можуть представляти у вигляді спектрограм – це візуальне представлення спектра частот сигналу, що змінюється з часом. Він Вона має декілька вимірів: одна вісь представляє час, а інша вісь – частоту. Третій вимір, що вказує амплітуду певної частоти в певний момент часу,

представлений інтенсивністю або кольором кожної точки на зображенні [48].

Приклад звичайної спектрограми з її порівнянням з іншими представленнями аудіосигналу наведено на рисунку 2.2.

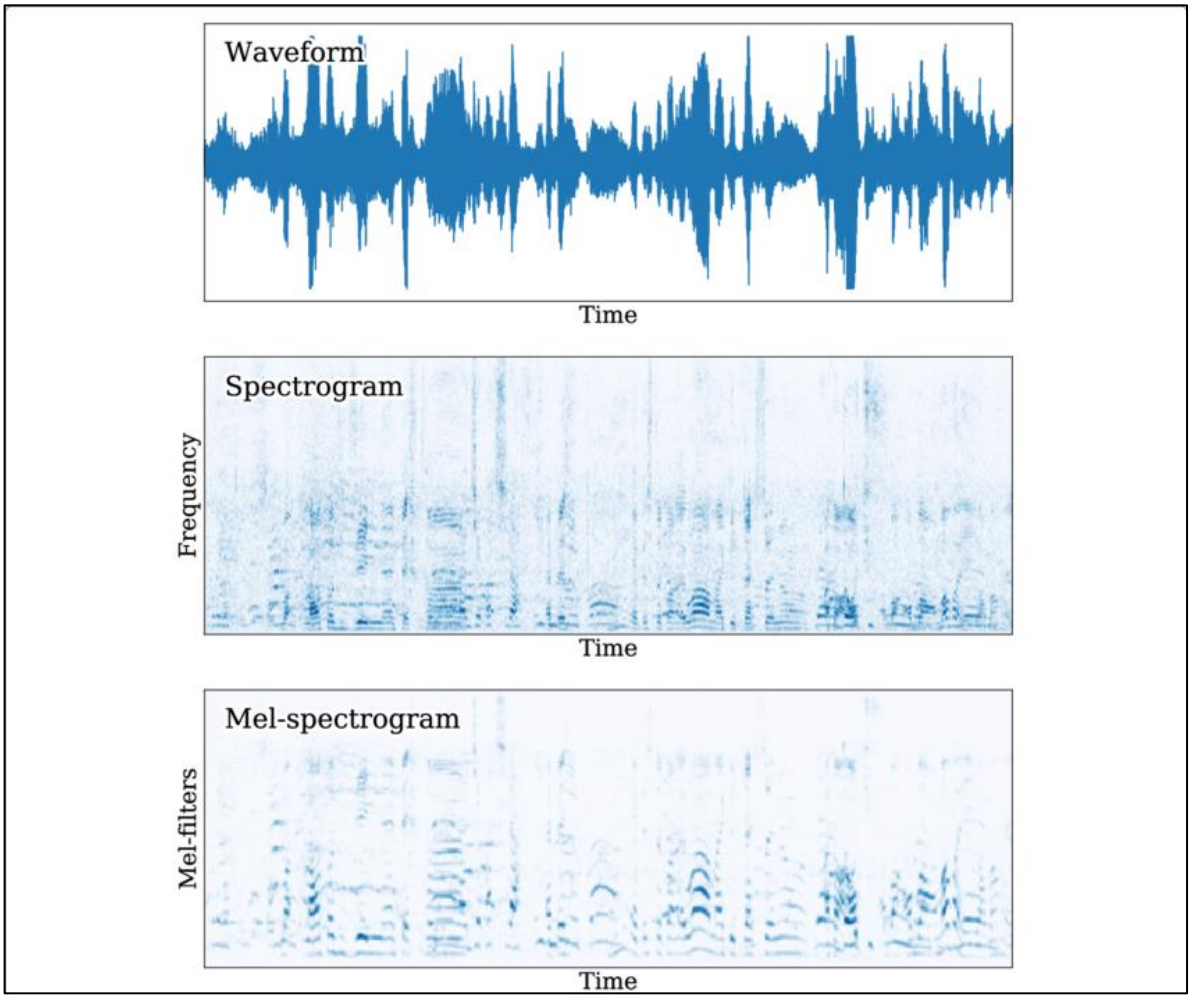

Рисунок 2.2 – Порівняння спектрограми з іншими аудіо-представленнями

В задачах МН зазвичай використовують модифіковану спектрограму, яка називається мел-спектрограмою. Вона має дві важливі відмінності порівняно зі звичайною спектрограмою. Перша це те, що інтенсивність кольору вимірюється шкалою децибелів, замість амплітуди. Інша, від якої і пішла назва даної спектрограми, є те, що використовується шкала Мела. Дослідження показали, що люди не сприймають частоти в лінійному

масштабі. Натомість краще виявляється різниця в нижчих частотах, ніж у вищих. У 1937 році Стівенс, Фолькманн і Ньюманн запропонували одиницю вимірювання висоти звуку, за якої рівні відстані за висотою звуку звучали б для слухача однаково віддалено. Це називається шкалою Мела, яку зображено на рисунку 2.3.

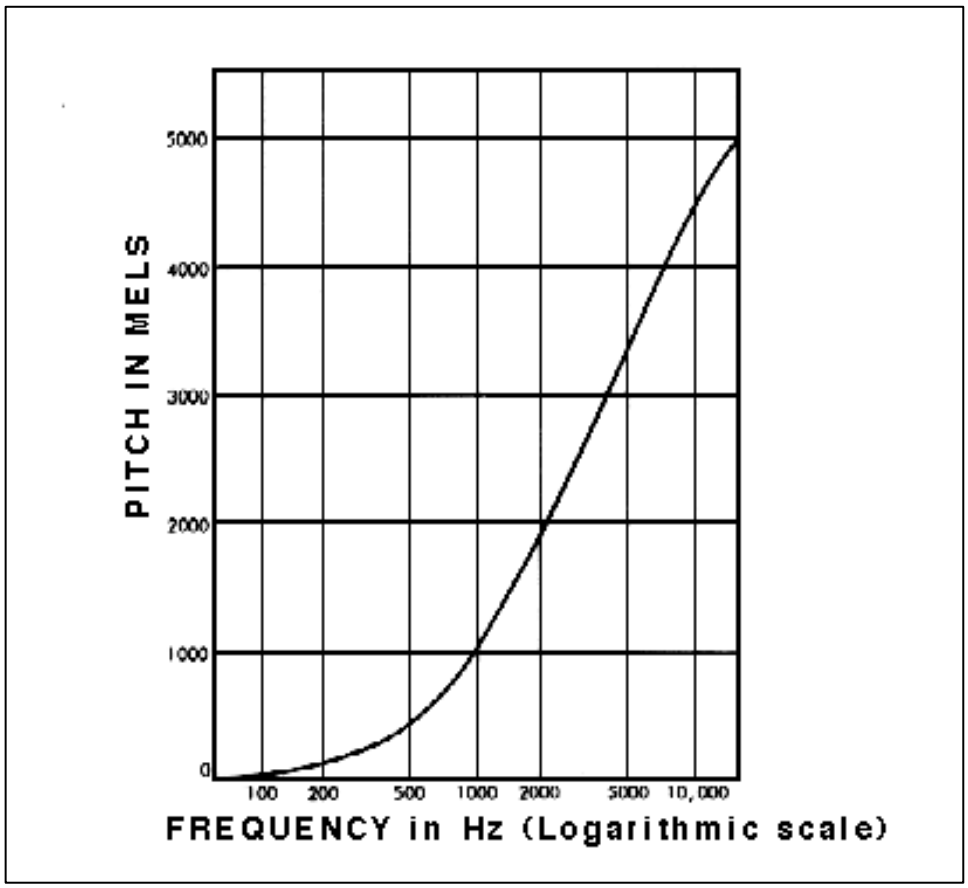

Рисунок 2.3 – Шкала Мела

Саме такі мел-спектрограми використовуються у згорткових нейронних мережах, бо вони імітують сприйняття звуку людиною, є кращою для задач, де важливе сприйняття. Алгоритм побудови є багатоетапним із застосуванням інших математичних методів, таких як швидке перетворення Фур'є, тощо. Приклад побудови мел-спектрограм наведено на рисунку 2.4 [49].

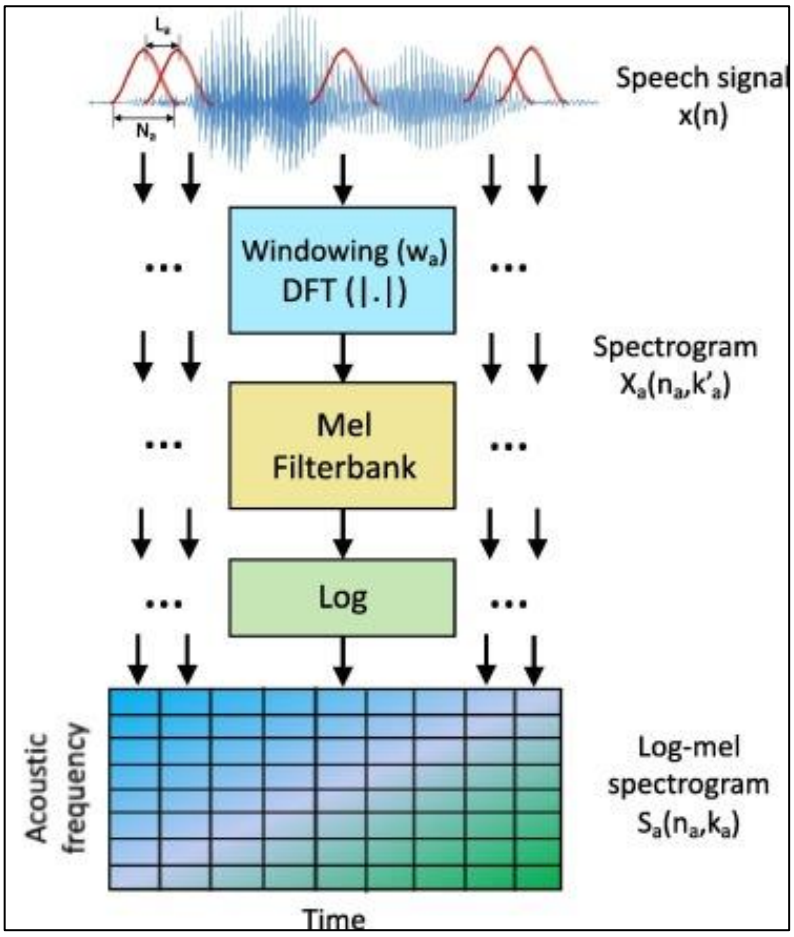

Рисунок 2.4 – Алгоритм знаходження спектрограм Мела

Іншим представленням аудіосигналу є мел-частотні кепстральні коефіцієнти (MFCC). Вони можуть використовуватись для тембрального опису та порівняння тембрів. Вони стискають загальний спектр до меншої кількості коефіцієнтів, які разом описують загальний контур спектра. Їх значення отримують шляхом виділення і нормалізації сигналу, його кадрування, застосування перетворення Фур'є для перетворення їх з часової області в частотну та застосуванні фільтрів Мела для обчислення логарифмічного значення величини потужності, як і для побудов спектрограми. Важливим кроком є обчислення спектру результатів, отриманих на попередньому кроці, шляхом застосування дискретного косинусного перетворення (ДКП), що призводить до кепстральних коефіцієнтів, представлених рівнянням 2.1 [50].

$$c(n) = \sum_{0}^{M-1} \log_{10}\big(s(m)\big) \cos\left(\frac{\pi n(m - 0.5)}{M}\right), \qquad (2.1)$$

де c(n) – MFC коефіцієнт;

n – індекс коефіцієнта;

M – кількість мел-фільтрів;

m – індекс мел-фільтра;

$s(m)$ – енергія у m-му мел-фільтрі.

Ці коефіцієнти є зручними, оскільки вони можуть описати достатньо великий фрагмент аудіо при цьому задіяючи невелику кількість коефіцієнтів (зазвичай 13), але може достатньо абстрагувати спектр, що певні нюанси в еквалізації будуть втрачені. Хоча вони зручні для роботи в МН, зберігають відповідність людському слуху та більш компактні. В задачах з розпізнавання мовлення такі коефіцієнти достатньо часто використовуються.

Тому для порівняння записів між собою цих ознак буде достатньо для мінімального опису, але ефективного опису аудіосигналу. Для порівняння можна використовувати різні підходи. Наприклад, сіамську нейронну мережу (Siamese neural network), яка використовує однакові ваги, працюючи одночасно з двома різними вхідними векторами для обчислення порівнянних вихідних векторів. Використовувати можна CNN мережі для порівняння саме мел-спектрограм, але якщо потрібно більше описових характеристик для аудіосигналу цього може бути недостатньо. Хоча як більш легка альтернатива ця модель може розглядатись. Або можна розглянути за таким принципом інші згорткові нейронні мережі, як U-Net для сегментації зображень і знаходити різницю спектрограм. За логікою пошуку різниці між ознаками можна використовувати різні моделі. Часто в аудіотехнологіях з ШІ пов’язані моделі Transformer або Autoencoder. Але вони менш контрольовані і можуть потребувати додаткового аналізу та інформації для створення.

Натомість більш гнучким підходом є навчання з підкріпленням – це клас методів машинного навчання, де агент взаємодіє з середовищем, отримуючи за свої дії винагороди і спостерігаючи нові стани. Основна мета агента є навчання політиці вибору дій, яка максимізує сумарний сигнал винагороди. Середовище зазвичай моделюється як процес Маркова з визначеним простором станів, простором дій та функцією винагороди.

Якщо правила роботи середовища відомі, можна використовувати методи динамічного програмування, які поступово обчислюють найкращі стратегії для кожного можливого стану. Але в реальних задачах часто неможливо точно знати модель середовища або вона надто складна. У таких випадках застосовують методи без моделі (model-free), які навчаються безпосередньо на основі отриманих нагород. Одним із найвідоміших методів є Q-learning – це off-policy алгоритм, який навчається оцінювати найкращі дії, навіть якщо сам агент діє інакше, тобто оцінює оптимальні дії незалежно від досвіду. Він намагається поступово побудувати таблицю (або модель), яка для кожного стану та дії зберігає оцінку їхньої корисності (Q-функцію). Згодом агент обирає ті дії, які мають найвищі значення. Інший подібний метод – SARSA, який є on-policy алгоритмом. Він навчається на основі дій, які сам агент реально виконує [51]. Обидва ці методи працюють без знання внутрішньої структури середовища та використовують або Q-таблицю, або її наближення.

З появою глибокого навчання виникла нова галузь – Deep Reinforcement Learning. Наприклад, Deep Q-Network (DQN) об’єднує Q-learning з нейромережею, яка замінює таблицю Q-функції і може працювати у складних середовищах з великою кількістю ознак. Щоб навчання було стабільнішим, DQN використовує реплей-пам’ять (перегравання попереднього досвіду) та цільову мережу. Інший клас методів – actor-critic. Тут агент має дві частини: актор, який вибирає дії та критик, який оцінює ці дії. Такий підхід дозволяє ефективно поєднувати методи, що базуються на політиці, та методи, що базуються на оцінці значень. Actor-critic добре

працює в умовах, де простір дій є неперервним, і використовує градієнти політики для оновлення стратегії. Серед сучасних і популярних методів вирізняється Proximal Policy Optimization (PPO) – алгоритм, який обмежує різкі зміни в політиці, що забезпечує стабільність навчання. Також вартий уваги AlphaZero – алгоритм, який поєднує навчання з підкріпленням і пошук за деревом (MCTS), навчається без попередніх знань і досягає надлюдських результатів у складних іграх, як-от шахи чи го, завдяки грі проти самого себе.

RL також достатньо поширений у музиці. Останнім часом його почали використовувати в задачах генерації музики. Наприклад проєкт Magenta, який неодноразово згадувався в даній роботі, розробила модель RL Tuner, яка допомагає моделі LSTM навчатися музичної теорії для генерації мелодій [52]. Або інша модель RL-Duet, яка в режимі реального часу створює акомпанемент на основі мелодії, яку грає людина. Навчальна модель оцінює, наскільки згенерована нота гармонійно поєднується з мелодією, що дозволяє агенту імпровізувати разом з виконавцем [53].

Тож навчання з підкріпленням активно використовується у музиці і його застосування є коректним. Безумовно агентна модель може мати недоліки, оскільки вона більша за обсягом, складніша та важча у обчислювальному, але вона гнучка та зрозуміла для потреб генерації. В ній можна налаштувати власний простір дій, як агент буде впливати на середовище, експериментувати з метриками нагород, тощо.

В межах даного проєкту для генерації даних було обрано DQN архітектуру агента. З вищерозглянутого оптимальним буде стратегія саме порівняння референсного та сирого аудіозаписів, які будуть подаватися на вхід. В результаті буде отримано всі потрібні параметри еквалайзеру. За чіткою метрикою агент буде отримувати нагороду і концентруватися на діях, що істотно зменшують різницю з референсом, замість «випадкового» застосування фільтрів. Також можна застосувати ε-greedy стратегію, яка балансує процеси дослідження на експлуатації агентом дій, бо так він може

застрягнути у локальному максимумі і пропустити потенційно кращі варіанти. При цьому можна зменшувати це значення з початку навчання.

Простір дій буде дискретним для кожного з параметрів еквалайзеру для чотирьох різних фільтрів. Це буде менш обчислювально витратно і спростить задачу без суттєвих втрат нюансів еквалізації, оскільки самі параметри еквалайзеру мають фіксований крок до десятих у певних межах. Моделлю буде нейромережа, яка буде видавати Q-значення для кожної дії та під кожний параметр. При цьому в DQN є дві мережі – основна та цільова. Основна змінюється кожен крок і прогнозує Q-значення для поточного стану, а цільова використовується для обчислення значень у функції витрат. Цільова може оновлюватись певну фіксовану кількість кроків або частково змішувати ваги. Таким чином DQN алгоритму не потрібна явна модель переходів чи складні математичні рівняння, оскільки агент повинен просто застосувати дію та отримати винагороду, без застосування складних методів DSP, окрім фільтрів, якими агент змінює аудіосигнал. Зручним є і те, що вона використовує буфер досвіду та окремі моделі параметр, що пришвидшує збіжність та дозволяє паралельно оптимізувати параметри еквалайзера, тому навіть додавання нового фільтру не буде проблемою.

Враховуючи вище сказане, навчання з підкріпленням є достатньо універсальним та декомпозитивним методом, в якому можна налаштувати все для отримання оптимальних параметрів еквалайзеру на основі різниці між еталонним та сирим треками. В подальшому при реалізації будуть обрані та розроблені функції винагород під дану задачу. Не дивлячись на обчислювальну складність та час навчання, в порівнянні з іншими підходами, DQN агент є більш доступним в реалізації, легкості налаштування та гнучкості і розумінні, бо PPO мають більше гіперпараметрів та складніша в реалізації і обчисленнях. Натомість кероване навчання не підходить, бо немає параметрів еквалайзеру, тож таку модель буде розроблено в подальшому для прогнозування параметрів на основі згенерованих даних. Grid-пошук є також обчислювально дуже складним,

хоч і простим. Тому дане рішення є оптимальним, як і зі сторони ефективності генерації набору даних, так і зі сторони обчислювальної складності, часової складності та реалізації даної моделі.

Рішення з використанням моделей, які аналізують якість звуку, таких як HAAQI-Net, Whisper від OpenAI, SpeechQ та інших подібних моделей не розглядаються. Безумовно, це було би зручно, оскільки на вхід можна подавати лише один запис і змінюючи параметри еквалайзеру отримувати конкретну оцінку якості запису і відштовхуючись від неї вказувати правильний напрям для агента. Це також полегшило задачу, оскільки не потрібно тренувати окрему модель для прогнозування параметрів. Але оцінка якості звучання – це оцінка, яка позбавлена будь-якої творчої складової. Тобто, вона орієнтується саме на покращення звуку, а не на поліпшення еквалізації і творчої складової, бо іноді в таких процесах є важливим підсилити частоту, яка можливо буде створювати певний шум, але це буде певним творчим рішенням.

У висновку, за допомогою такої моделі можна отримати кожен параметр для кожного з чотирьох фільтрів і при цьому можна витягнути з сирої аудіодоріжки лише необхідні параметри, які будуть потрібні для побудови supervised моделі для тренування і застосування її у плагіні у вигляді ONNX моделі, а не ті які використовуються для побудови агента, достатньо просто поставити у відповідність параметрам або ознакам сирої аудіодоріжки знайдені оптимальні параметри еквалайзеру, які наблизять її до референсного стану.

2.4 Вибір моделі МН для прогнозування параметрів

Після генерації набору даних агентом потрібно створити окрему модель, яка зможе на згенерованих даних передбачати параметри еквалайзеру для оптимального налаштування звуку. В даному етапі є важливим вибір ознак для тренування моделі, які будуть подаватися на вхід

та репрезентувати сирий трек, оскільки їх багато і важливо знайти баланс між достатньою кількістю ознак і тим наскільки точно вони зможуть описати вхідний аудіотрек для підбору параметрів під нього, щоб не перевантажувати модель та разом з тим не ускладнювати її роботу плагіну, в який вона буде інтегрована.

Не менш важливим є момент експорту моделі, бо якщо модель не підтримує експортування в ONNX формат, її неможливо буде інтегрувати. Звідси випливає те, що моделі, які можуть підходити для вирішення даної задачі з підбору параметрів неможливо буде використати. Для експорту гарно підходять нейронні мережі, які створено за допомогою бібліотек pyTorch та tensorFlow, бо вони мають підтримку даного формату і він є одним з найпоширеніших у застосуванні моделей в інших мовах. Також бібліотека sklearn має власну підтримку експортування моделей у формати pkl та ONNX. Тож, при виборі моделей у першу чергу будуть використані саме ці бібліотеки. Інші алгоритми, наприклад градієнтного бустингу (XGBoost), може бути складно експортувати.

Також при експорті слід враховувати вбудовану обробку вхідних параметрів, таких як нормалізація та те, що не всі моделі мають змогу для побудови кількох виходів. В таких випадках буде створено моделі під кожний параметр, що ускладнює інтеграцію і робить роботу плагіну більш важкою, хоча все залежить від ваги моделі. З іншого боку, створення моделі під параметр дає змогу комбінувати різні моделі і знаходити оптимальні рішення, наприклад для параметрів підсилення використовувати один вид моделі, а для частоти інший. У випадку pyTorch моделей такої проблеми немає – модель експортується одна з виходом під усі параметри.

2.4.1 Вибір ознак для тренування моделі

Є багато способів репрезентувати аудіодоріжку чи фрагмент. Існує багато ознак, які створено для різних типів задач. У попередньому підрозділі

було розглянуто основні способи, у вигляді спектрограм, мел-спектрограм та MFCC. Як основу представлення аудіофрагменту на вхід моделі МН буде використано мел-спектрограмні характеристики, а саме середнє та стандартне відхилення енергії у відповідних мел-смугах. Це зумовлено тим, що спектрограми зазвичай не використовуються, а амплітуда є менш корисною, аніж шкала децибелів. А в порівнянні з MFCC, який частіше використовується в задачах розпізнавання мовлення, мел-спектрограмні характеристики зберігаються спектральну форму і безпосередньо відображають енергію на мел-шкалі частот. MFCC занадто стискає спектр через використовуване дискретне косинус-перетворення і передає менше нюансів саме музичних. Другий момент це те, що MFCC не можливо пов’язати з певною смугою чи частотним діапазоном, оскільки ці коефіцієнти навпаки найбільш компактно намагаються представити аудіофрагмент у кількості до тринадцяти коефіцієнтів. Через таке узагальнення даних і використання фільтрації та перетворень дані коефіцієнти менше підходять для задачі, хоча їх використовують у задачах класифікації мовлення або жанру, бо там важливі саме глобальні особливості.

Іншими корисними ознаками є відображення сумарної енергії сигналу у відповідних діапазонах частот. Це дозволяє зрозуміти, які ділянки спектра переважають у звуці. Наприклад, басова музика матиме багато енергії у низькому діапазоні. Разом із ними використовуються і частки цієї енергії для кожної смуги від загальної суми енергій, що показує баланс звуку. І окремо створено числові значення відношення енергії між ділянками спектра. Вони дають інформацію про нахил спектра, що напряму пов’язане з тим, які частоти можливо треба підсилити чи зменшити через еквалізацію.

Великим блоком йдуть різноманітні спектральні характеристики. Це описові числові ознаки, які пов’язані з енергією сигналу та її розподілом по частотах. Вони є дуже важливими у задачах, пов’язаних із обробкою тембру, тону, «забарвлення» аудіо, чим і займається еквалайзер. Було використано

достатню кількість таких характеристик. Першим є центроїд, що є середньозваженою частотою спектра, а отже центром ваги спектра, де зосереджена енергія. Він допомагає зрозуміти чи потрібно посилювати або послаблювати верхні або нижні частоти. Відповідно, якщо він високий, то звук має багато енергії у верхніх частотах.

Плоскість спектру оцінює, наскільки спектр схожий на «білий шум» (плоский) або має виражені піки. Він може бути корисним для налаштування верхніх та середніх частот, і розуміння чи є сигнал тональним або шумовим. Низьке значення вказує на більш гармонійний звук. Спектральний контраст є різницею між енергією «піків» та «провалів» у частотних смугах. Якщо контрасту багато, то звук є із чітко виділеними гармоніками, якщо менше, то він більш заповнений, рівний. Еквалайзер саме і застосовується для керування такою контрастністю, тож ця ознака є корисною. Поріг спадання (rolloff) – це частота, нижче якої знаходиться заданий відсоток від загальної спектральної енергії, наприклад 85% [54]. Може вказувати на розтягнутість спектру і безумовно корисний при налаштуванні верхніх частот. Нахил спектру є прямим індикатором тонального балансу, що є також ключовою інформацією для еквалізації. Якщо нахил йде вниз, то переважають низькі частоти. Гострота спектру є зважене середнє частот з урахуванням енергії і чим вище його значення, тим більше звук концентрується у високих частотах.

Окремо розглянуто ознаки, які дають статистику форми спектру. Це спектральна дисперсія, асиметрія спектра та куртозність. Дисперсія показує відхилення частотної енергії від її середньої частоти, що дає уявлення про ширину спектра – чи звук сфокусований чи розкиданий по всьому спектру. Асиметрія вказує в якій половині зосереджено більше енергії, що корисно для налаштування середніх діапазонів. Та куртозність, яка допомагає виявити, де в спектрі зосереджена енергія сигналу, та чи є у спектрі піки або широкий розподіл.

Останньою ознакою було обрано RMS енергію – це середньоквадратична енергія сигналу. Вона показує загальну гучність або потужність звукового сигналу в певному відрізку часу. На відміну від простої амплітуди, RMS гладша, менш чутлива до окремих піків та вона краще передає сприйнятну гучність на слух, ніж просто максимум. Це є більш допоміжною ознакою разом зі спектральними, щоб модель бачила загальний контекст гучності фрагменту аудіо.

Даний набір ознак є збалансованим і дає повну картину частотної структури, енергії та форми сигналу, що є достатнім для передбачення параметрів. Головний акцент зроблено на ознаках, які могли би не просто передати наскільки сигнал гучний, чи є у ньому шум та інші більш узагальнені ознаки, а саме ознаки, які можуть описати більш гнучко та музично фрагмент, що буде подаватися на вхід.

2.4.2 Порівняння моделей МН

Як було сказано вище, акцент на виборі моделі повинен бути не тільки на якості, оскільки високої якості досягнути достатньо складно з обмеженими обчислювальними ресурсами, а й на експорті моделі чи моделей у плагін. Саме тому в першу чергу увагу буде акцентовано на моделях з бібліотеки sklearn, які дають достатньо широкий вибір.

Першою моделлю є RandomForestRegressor. Це класична і достатньо широко використовувана ансамблева модель МН, яка складається з великої кількості дерев рішень, кожне з яких навчається на випадковій підмножині даних. Всі дерева працюють незалежно, а їхні результати усереднюються, щоб отримати фінальний прогноз. Це є також класичним прикладом ансамблевого навчання з методом беггінгу.

Для даної задачі вона підходить через свою простоту, вміння працювати без масштабування чи препроцесінгу. А також її можна навчати з кількома вихідними параметрами, що полегшить процес інтеграції у

плагін. Також її перевагами для даної задачі є робота з числовими ознаками різного масштабу і вона може добре працювати на невеликих та середніх наборах даних (у випадку згенерованого набору 10 тис. екземплярів). Ця модель також менше схильна до перенавчання, через те що метод ансамблевий, на відміну від звичайного дерева рішень.

Також було створено окремо моделі під кожен параметр еквалайзера для порівняння з результатами моделі з кількома виходами. Це зроблено для того, щоб зрозуміти чи є сенс створювати та використовувати окремі моделі, тим самим ускладнюючи процес інтеграції. Окремі моделі створено з тими ж самими параметрами, що й модель з кількома вихідними параметрами, і набори даних використані теж однакові. В ході тестування виявилося, що ніяк змін в результатах між окремими моделями та однією моделлю не виявлено, вони точно співпадають, і всі моделі без проблем експортуються. Надалі створення моделей з бібліотеки sklearn буде тільки з кількома виходами, бо їх зручніше інтегрувати. При цьому протестувавши оригінальну модель і таку ж саму експортовану ONNX модель було виявлено несуттєві зміни у результатах передбачень, що не є суттєвою проблемою враховуючи інший формат моделі.

Іншим методом з цієї бібліотеки є GradientBoostingRegressor, який також є методом ансамблевого навчання, але вона працює за принципом бустінгу, коли всі дерева будуються послідовно. Суть даної моделі, що кожне наступне дерево намагається виправити помилки попередніх. Фінальним прогнозом є сума всіх дерев. Ця модель також може давати кілька вихідних параметрів і гарно працює з табличними числовими даними. Є думка, що цей метод точніший за випадковий ліс, особливо коли є нелінійні зв’язки в даних.

Інші два методи XGBoost та LightLGB є також ансамблевими методами, але з окремих бібліотек, відповідно для їх експорту потрібні інші інструменти. Великим недоліком є складність експорту моделей з багатьма

вихідними ознаками, тому кожна модель тренувалась та експортувалась окремо, що робить інтеграцію в плагін важчим.

XGBoost – це вдосконалена реалізація градієнтного бустингу, орієнтована на високу продуктивність і точність та є одна з найефективніших моделей для задач регресії. Ця модель користується популярністю на таких сервісах, як Kaggle, бо забезпечує високу якість результатів. Він схожий на метод з бібліотеки sklearn, також використовує градієнтний бустинг над деревами рішень і підходить до задач з табличними даними. В своїй реалізації він навчається на помилках інших дерев, і мінімізує похідну функції втрат. Також в цей метод додано регуляризацію та обрізку дерев, що допомагає уникати перенавчання і контролювати складність моделі [55].

Інша обрана модель LightGBM також є моделлю ансамблевого типу, яка реалізує градієнтний бустинг над деревами рішень. Але вона має одну важливу особливість, що відрізняє її від інших: на відміну від класичних алгоритмів, які ростять дерева рівень за рівнем (level-wise), LightGBM використовує стратегію leaf-wise. Це означає, що на кожній ітерації розгалужується листок, розбиття якого дає найбільше зменшення функції втрат. Такий підхід дозволяє отримати глибші дерева та часто кращу точність, але потребує контролю для уникнення перенавчання. Також вона використовує цікаві техніки, як GOSS, що залишає всі приклади з великими градієнтами (тобто ті, на яких модель помиляється найбільше), а з решти – випадково вибирає підмножину. Це дозволяє зменшити обсяг даних для навчання без втрати точності [56].

Так як нейронні мережі, які побудовані за допомогою pyTorch або TensorFlow без проблем експортуються в ONNX форматі та мають гарну їх підтримку, буде розроблено декілька архітектур для тестування. У випадку використання таких моделей можливою проблемою є недостатня кількість даних для тренування, що може погіршити їх результати. Також особливістю підготовки даних до тренування є потреба в їх попередній

нормалізації на вхід нейромережі, що може ускладнити процес імпорту і потребувати або окремої реалізації нормалізації даних на вхід моделі у плагіні, або інтеграція нормалізації у ONNX модель перед входом нейромережі. Моделі бустингу, які в основі містять дерева рішень цього частіше не потребують, оскільки дерева розбивають простір ознак за пороговими значеннями, тому можуть автоматично обробляти різні масштаби ознак без втрати якості. В ході тестування моделей було також створено модель RandomForest з нормалізацією даних і підтверджено, що підвищення якості моделі ніякого немає, що дозволяє видалити попередній крок у підготовці даних та експорті моделі.

Було створено декілька простих архітектур нейромереж для тренування та тестування на створених даних. Всі моделі приймають на вхід однакову кількість ознак і видають на виході 12 параметрів під кожний фільтр. Перша архітектура є багатошаровою нейронною мережею з двома ключовими блоками та одним пропускним з’єднанням. Вхідний шар перетворює вхідний вектор у проміжне представлення розмірності 512. Це достатньо типове число і є ступенем двійки, що зручно для обчислень на апаратному рівні, і є достатньо великим, щоб розширити простір початкових ознак для виявлення прихованих закономірностей, але і не надто велике, щоб спричинити перенавчання, оскільки використовується регуляризація. Менша кількість може не забезпечити достатній простір для моделювання складних залежностей. Тому кількість нейронів є оптимальною для такої архітектури.

Перший блок двічі обробляє 512-вимірний тензор, але в кінці додає до результату початкові значення цього ж шару (пропускне з’єднання). Такий прийом допомагає мережі навчитися змінювати вхідну репрезентацію лише там, де це дійсно потрібно, а також уникнути затухання градієнтів і дає змогу мережі бути більш стійкою до перенавчання у першій частині з великою кількістю параметрів. Другий блок вже немає пропускного з’єднання, оскільки він понижує розмірність вектору. Цей блок є наступним

етапом – поступове концентрування найважливішої інформації у меншому просторі, щоб надмірно не розширювати модель. У вихідному блоці виконується лінійне відображення і відбувається перехід до кількості вихідних параметрів. Dropout перед останнім шаром не використано, бо зайве зашумлення на цьому етапі може завадити точності. Така модель повинна бути ефективною, бо вона має достатню внутрішню ширину, має пропускне з'єднання та кожний етап ущільнює значення мережі. Ця мережа може мати досить велику вартість обчислень, але нею можна знехтувати, якщо якість прогнозу буде виправданою.

Друга модель є також глибокою повнозв'язною мережею з кількома шарами. Вона має глибшу архітектур на відміну від попередньої та активно використовує пакетну нормалізацію. Всі блоки побудовані схожим чином: першим йде лінійний шар, потім пакетна нормалізація, функція активації типу Relu та регуляризація у вигляді виключення нейронів. Структура блоків однакова, але значення лінійних шарів та шару Dropout поступово змінюються, а точніше зменшуються. Пакетна нормалізація є досить корисною, оскільки зменшує чутливість до початкових параметрів і дозволяє використовувати вищу швидкість навчання та загалом прискорює конвергенцію моделі. Ймовірність відключення нейронів також зменшується поступово, щоб на початках запобігти перенавчанню, а в кінці менше зашумлювати дані. Недоліки має такі, як і в попередній архітектурі: велика кількість параметрів та вища обчислювальна складність.

Останньою є архітектура з використанням механізму self-attention (самоуваги), що дозволяє мережі навчатися зважувати кожну ознаку стосовно інших у рамках одного пакету даних. Структура першого блоку з вилученням ознак повторює логіку блоків попередньої архітектури, і потрібно отримати інформативне представлення ознак аудіо. Далі йде шар з механізмом уваги, а саме MultiheadAttention. Він представляється певною кількістю голів (heads), що є окремим набором механізмів уваги, який незалежно обробляє вхідні дані, числовий вектор певної розмірності, і

перетворює їх у три набори векторів за допомогою ініціалізованих матриць ваг: запитів (Query), ключів (Key) та значень (Value). Для кожної такої голови обчислюється скалярний добуток між запитами та ключами, масштабований квадратним коренем розмірності. Після застосовується softmax-функція для отримання ваг уваги, які визначають, які частини вхідних даних є найважливішими. В кінці вихідні вектори з усіх голів конкатенуються і проходять крізь лінійний шар для формування фінального результату [57]. Таким чином мережа може вивчити, які компоненти вхідного вектора слід підсилити чи ослабити, залежно одна від одної. Наприклад, якщо деякі спектральні ознаки корелюють із відповідним значенням gain або Q, механізм уваги може встановити високий ваговий коефіцієнт між ними. Після даного блоку також використано пропускне з'єднання, щоб у випадку якщо цей блок нічого корисного не додає, то мережа може зберегти вихід попереднього блоку без змін. Останній блок просто поступово згортає велике представлення ознак до вихідного числа прогнозів для кожної ознаки. Такий механізм має мати здатність виявляти складні взаємозв'язки та вплив декількох ознак одночасно на параметр еквалайзеру і може дати гарні результати, ніж звичайна MLP. Безумовно, такий механізм ще більше ускладнює обчислення, ніж попередні моделі, через аналіз та обрахунок взаємозв'язків у декількох головах, тому було знижено розмірність шарів у моделі.

Всі моделі було протестовано на однаковому наборі даних, у розмірі 15 тисяч записів для тренування та тестування, а ключовою метрикою для оцінки обрано R^2 (коефіцієнт детермінації) – це показник якості регресійної моделі, який показує, наскільки добре модель пояснює варіацію цільової змінної. Чим вищий коефіцієнт, тим вищий відсоток варіації вхідних даних вона пояснює. Для більш точного результату дана метрика буде обрахована для кожного вихідного параметру окремо, тобто для кожного параметру фільтру еквалайзера. Дані оцінки будуть обраховано на етапі саме тестування моделі, через ризик перенавченості моделей. Також разом з

оцінкою було зроблено спробу експорту навчених моделей. У результаті всі моделі мають підтримку експорту у ONNX через різні інструменти, але певні моделі, як і зазначалося раніше (XGBoost, LightGBM), не мали змогу бути експортованими з декількома виходами. Протестовані моделі ONNX мали незначні відхилення у результатах від оригінальних моделей з яких їх було експортовано, що не вплинуло на якість передбачення. Результати тестувань наведено у таблиці 2.1.

Таблиця 2.1 – Результати тестувань моделей

Модель / Параметр	Random Forest	Gradient Boosting	XGBoost	LightGBM	Residual NN	Batch NN	Attention NN
low_freq	0.31	0.30	0.31	0.30	0.35	0.37	0.37
low_gain	0.71	0.73	0.73	0.72	0.75	0.75	0.76
low_q	0.05	0.05	0.05	0.05	0.05	0.07	0.08
low_mid_freq	0.16	0.17	0.17	0.17	0.25	0.26	0.27
low_mid_gain	0.50	0.52	0.51	0.51	0.59	0.60	0.60
low_mid_q	0.001	0.003	0.0	0.0	0.0	0.0	0.01
high_mid_freq	0.10	0.08	0.09	0.09	0.13	0.15	0.17
high_mid_gain	0.33	0.34	0.34	0.34	0.39	0.37	0.39
high_mid_q	0.004	0.0	0.0	0.0	0.0	0.003	0.0
high_freq	0.006	0.0	0.0	0.0	0.0	0.0	0.0
high_gain	0.72	0.72	0.72	0.72	0.77	0.78	0.78
high_q	0.0	0.004	0.0	0.0	0.01	0.01	0.01

Як видно з результатів, для більшості параметрів всі моделі показали незадовільний результат. Нейромережі показують стабільно вищі результати в порівнянні з усіма ансамблевими методами, хоч відрив у результатах невеликий.

Найкращою моделлю за тестуванням є нейронна мережа, яка включає механізм уваги, її результати для кожного параметру наведено на рисунку 2.5.

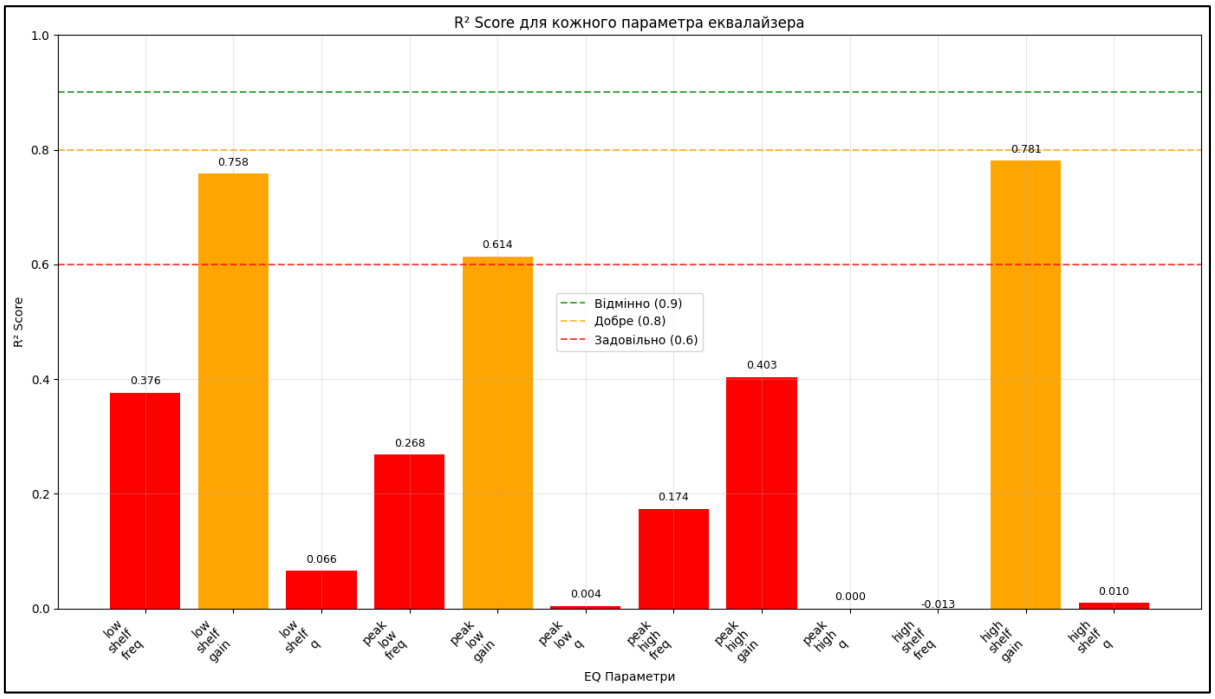

Рисунок 2.5 – Результати тестування нейромережі з механізмом уваги

Найвищі значення загалом спостерігаються для параметра підсилення, особливо для крайніх фільтрів. Це дуже добре, оскільки даний параметр є основним та опорним для налаштування звуку, хоча результати не достатньо хороші. Значення смуги пропускання (добротності) передбачити складно всім моделям. Це означає, що ні ансамблі, ні нейромережі практично не можуть пояснити варіацію цих параметрів на основі доступних вхідних даних. Тому можливо додавання додаткових ознак вирішило б цю проблему, хоча жодна ознака напряму не пов’язана з даним параметром, тому очевидно, що передбачити його найскладніше.

Параметр добротності є найменш очевидним навіть при налаштуванні еквалайзера людиною, а значить і найбільш суб’єктивним, оскільки параметр підсилення має простіший зворотній зв’язок: якщо переважають низькі частоти, значить їх знижують, а параметр добротності не має такого очевидного налаштування.

Друга проблема це те відсутність достатньої кількості даних. Оскільки додавання додаткових записів покращило якість передбачення, можливо

збільшення кількості даних вирішило б проблему з передбаченням параметра підсилення, як мінімум.

У висновку можна сказати, що протестовані архітектури нейронних мереж є хоч і не набагато, але більш точними. Найкращі результати показала архітектура з механізмом уваги, але вони несуттєво відрізняються від архітектури з нормалізацією. Тому при виборі для впровадженні моделі буде використана одна з них, враховуючи їхні особливості інтеграції та обчислювальну складність. Оскільки нейромережа з використанням уваги дає кращі результати, але є і більш складною з точки зору обчислень, перевагу при впровадженні може отримати більш швидка нейромережа.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАГІНУ

В даному розділі буде описано весь шлях програмного створення плагіну, який буде опиратися на вибір інструментів та методів з попереднього розділу. Реалізація буде охоплювати створення 4-х полісного параметричного еквалайзеру з основними параметрами, а також візуального інтерфейсу для нього. Далі буде висвітлено програмну реалізацію моделі агента для створення набору даних для навчання фінальної моделі, яку буде описано окремо. Наприкінці цього розділу будуть розглянуті інструменти для експорту та інтеграції моделі ШІ у базовий плагін еквалайзеру.

3.1 Розробка VST3-плагіна еквалайзера

У ході дослідження методів вирішення поставленої задачі, для розробки VST3 плагіну було обрано мову C++ та фреймворк Juce, який має весь необхідний набір інструментів для створення плагіну: як DSP методи, так і можливості збірки плагіну у специфічний формат. В даному підрозділі буде розглянуто розробку саме базового функціоналу плагіну, без використання моделі ШІ.

Разом з плагіном постачається програма Projucer, яка зручно створює різноманітні проєкті та збирає їх у різних середовищах програмування, у випадку даної роботи це Microsoft Visual Studio 2022, яка є однією з найвідоміших IDE, особливо по відношенню до мови C++. Після збірки проєкту у середовищі, Projucer створює проєкт з мінімальним функціоналом та шаблонами для програмування плагіну, а також імпортує потрібні бібліотеки, зокрема окремо підключено бібліотеку juce_dsp.

Код у даному плагіні за замовченням розбитий на дві частини. Першою є робота з логікою плагіну та звуковими блоками, друга відповідає за візуальний інтерфейс плагіну. Перший блок розробки, який буде розглянуто пов’язано саме з реалізацією логіки у файлах PluginProcessor,

тобто ту частину, яка відповідає за обробку аудіосигналу, роботу фільтрів еквалайзера, оновлення стану параметрів і підготовку даних для графіків у GUI. За замовченням створюється два файли: заголовковий, для оголошення класів, функцій, змінних, тощо, та сам файл реалізації.

В першу чергу відбувається ініціалізація плагіну. Створюється контейнер (дерево стану) для доступних у плагіні параметрів: чотири фільтри по три параметри (frequency, gain, Q) та прапор чи є фільтр активним. Імена параметрів формуються через серії функцій-утиліт, щоб зручно звертатись до кожної смуги по індексу і викликається функція, яка повертає перелік усіх параметрів фільтрів. Ініціалізуються вектори частотної осі, який заповнений значеннями від 20 Гц до 20 кГц за логарифмічною шкалою, щоб мати рівномірний розподіл точок та масив під кожний фільтр (магнітуда), в які записуються значення амплітуди фільтра (коефіцієнтів) для кожної точки частотного блоку. Таким чином, оновлюється крива фільтру, яка відображає в якій частоті вона підсилює або послаблює сигнал, щоб зрозуміти вплив кожного фільтру окремо. При ініціалізації створюються IIR-фільтри, які було оглянуто в першому розділі для кожної частотної смуги і заповнюються відповідні значення амплітуди фільтрів. За замовченням фільтри знаходяться у стані bypass, тобто не працюють. Для кожного параметру кожної смуги встановлюється listener, який дозволяє реагувати на зміни параметрів. І в кінці викликається функція, яка формує всю криву GFC – криву, для опису графіка частотної характеристики підсилення чи ослаблення сигналу.

Логіку можна розділити на декілька основних компонентів. Перший це сам клас плагіну з ініціалізацією, яка описана вище. Другий блок це AudioProcessorValueTreeState – дерево стану. При ініціалізації формується вектор об’єктів (всі параметри для кожної смуги), які і додаються до цього дерева. Всього виходить по 4 параметра для фільтру, які є стандартними для еквалайзеру з додаванням bypass-прапору. Це дерево забезпечує механізм автоматичного збереження або відновлення стану, нативне відправлення

повідомлень про зміну параметрів, а також інтерфейс для прив’язки слайдерів GUI до конкретних параметрів.

За допомогою DSP методів ініціалізується об’єкт filterChain, що представляє собою послідовність з чотирьох створених фільтрів і кожна смуга у процесі обробки аудіоблоку отримує частину сигналу і застосовує IIR-фільтр, якщо він не у стані bypass.

Важливим компонентом є аналізатор, який створено у окремому файлі зі своєю логікою та використовується у основному процесі аудіооброки. Їх створено два – один збирає дані до застосування фільтрів, а інший після, тобто оброблений сигнал. Вони використовуються у методі processBlock, що є ключовим методом, в якому обробляється кожен аудіоблок і застосовуються фільтри. Відповідно аналізатори викликаються перед обробкою аудіоблоку та після, щоб мати користувачеві змогу побачити, як застосовані фільтри змінюють вхідний аудіосигнал, що є корисним при аналізі частот та налаштування звуку.

Іншим компонентом є процес зміни параметрів еквалайзеру та взаємодія його з GUI. Основним методом в даному процесі є parameterChanged, який викликається автоматично, коли один з параметрів у tree змінюється користувачем. Допоміжним методом визначається до якої смуги належить параметр. Після цього змінюються коефіцієнти потрібного фільтра в окремому методі updateFilter. Метод отримує поточні значення фільтру з дерева стану і залежно від отриманого індексу створює нові коефіцієнти фільтра. Після оновлення фільтра у ланцюзі, переписується оновлена магнітуда у відповідний масив. Наприкінці цього методу викликається метод updatePlots. В цьому методі оновлюється GFX на основі тих фільтрів, які на час роботи є активними, а точніше їх значень амплітуди, щоб отримати загальну амплітудно-частотну характеристику. В кінці методу викликається метод sendChangeMessage, який повідомляє, що потрібно оновити графік частотної кривої, таким чином забезпечуючи зв’язок з GUI.

Окремо було створено параметр, який вказує чи працює фільтр в ланцюгу інших фільтрів. Це потрібно щоб вимикати непотрібні фільтри і вони не перетворювали звук у частині, де їх застосовано. Логіка проста і працює, як прапор з булевим значенням. Від того, чи працює фільтр залежить не тільки зміна звуку, але й зміна графіку частотної кривої на який вона впливає. Коли фільтр вимикається чи навпаки вмикається це також відображується у GUI.

Вище було описано всі основні компоненти, які створювалися окремо. Функції, які існували в логіці за замовченням майже не потребували змін. Це такі функції, які призначені для збереження та відновлення налаштувань, підтримка поточної конфігурації каналів (моно або стерео), методи для роботи з пресетами, які не використовувались. В методах, які запускають плагін або зупиняють обробку також зроблено мінімальні зміни, включно з налаштуванням аналізатора. В методі з обробки аудіо також зроблено мінімальні зміни для застосування фільтрів до аудіоблоків.

Інший файл PluginEditor реалізує роботу інтерфейсу програми. Його задача це створити всі елементи керування, відображати потрібні графіки та зв’язати GUI з параметрами DSP і оновлювати графіку. У цьому класі є внутрішній клас FilterEditor, який представляє одну смугу еквалайзера і включає всі потрібні елементи керування, наприклад ручки для змін параметрів та графік частотної характеристики фільтру. Безпосередньо в основному класі створюється чотири об’єкти цього класу і генеруються елементи управління, а також підписується на ChangeListener для реагування на зміни параметрів. Коли відбувається зміна від користувача, то викликається метод для перебудови графіків всіх смуг і основної, використовуючи метод з логіки плагіну та перемальовується GUI. Метод малювання задає дизайн інтерфейсу, а також малює графіки аналізаторів та криві кожної смуги разом із загальною. Також в цьому коді були і методи за замовченням, як наприклад resized, який реагує, коли користувач змінює розмір вікна. Головне у інтерфейсі, що він дозволяє динамічну

синхронізацію для миттєвого відображення змін у фільтрах. Приклад інтерфейсу плагіну наведено на рисунку 3.1.

Рисунок 3.1 – Візуальний інтерфейс еквалайзеру

Після створення плагіну, він успішно зібрався у VST3 форматі та був протестований у DAW. Всі функціональні елементи, як візуальні, так і застосування DSP методів та фільтрів працює успішно. Це вказує на те, що у всіх популярних DAW такий плагін зможе стабільно працювати.

3.2 Розробка агентної моделі для створення набору даних

Проблема пошуку даних для тренування моделей, які працюють з аудіоознаками є актуальною проблемою. В ході теоретичних досліджень було проаналізовано, що вирішення такої проблеми є створення даних за

допомогою програмних засобів чи моделей ШІ. Відповідно для тренування фінальної моделі потрібно створити дані. Для цього було вирішено використати навчання з підкріпленням та створити інтелектуального агента для обробки аудіо записів.

Зовнішнє середовище роботи агента буде представлено, як пари аудіофайлів – сирий, початковий запис та еталонний, еквалізований запис, до стану якого агент буде покращувати сирий. Оскільки для створення таких пар потрібні аудіофайли, які оброблено людиною для її власних потреб і при релізі певної роботи не викладається сирий трек, такі пари аудіофайлів знайти складно.

Для формування таких пар аудіофайлів було використано набір даних GTZAN. Цей набір використовується для жанрової класифікації і містить в собі колекцію з 10 жанрів, кожен з яких представлений аудіофайлами по 100 у кожному жанрі. Також, окремо він має мел-спектрограму під аудіофайл та навіть таблицю CSV з певними вилученими аудіоознаками [58]. З огляду на специфіку задачі, у роботі використовуються лише аудіофайли. Для формування пар еталонний-спотворений запис були штучно змінені референсні аудіозаписи з датасету, що приймаються за зразкові. З метою отримання спотворених версій до еталонних сигналів застосовувались еквалайзерні фільтри з випадково згенерованими параметрами. Кожен отриманий варіант зберігався окремим аудіофайлом. Таким чином можна генерувати нескінченну кількість даних, оскільки спотворювати запис можна багато разів і такий запис буде унікальним, а референсний трек залишається таким самим.

Опираючись на методи дослідження задачі, було реалізовано агентну модель з алгоритмом Deep Q-Network з покращенням у вигляді пріоритизованого буфера досвіду, м’якого оновлення цільових мереж та техніки Double DQN і функціонує, як генератор набору даних. Компоненти агента зображено на рисунку 3.2.

Рисунок 3.2 – Структурні компоненти агента

Агент працює повністю автономно, він самостійно вибирає дії, накопичує досвід та навчається без втручання ззовні. Користувач лише вказує гіперпараметри, але сам процес вибору дій і навчання відбувається автоматично. Сам агент насамперед реагує на поточний стан (вектор ознак, створений на основі пари записів) та обирає дію за ε-політикою. З огляду на те, що мережа обирає дію на основі поточного стану, він демонструє реактивну поведінку, бо не оцінює дії на декілька кроків вперед, як проактивні агенти [59]. Але він використовує буфер з накопиченим досвідом які дії призводили до якої винагороди, тому такого агента можна охарактеризувати, як реактивного агента з навчанням через досвід.

Даний агент чітко орієнтований на ціль максимізувати якість сирого аудіофайлу відносно референсного за метриками, які закладено у функцію винагороди. Дія обирається шляхом оцінки Q-функції, яка наближає відкладену нагороду.

Середовищем є набори пар аудіофайлів – сире та еталонне. Під час навчання агент випадково обирає одну з цих пар, завантажує обидва файли і працює з ними. Ці файли попередньо підготовлюються для отримання вектору ознак. Вони нормалізуються і вирівнюються за довжиною, щоб кількість обрахованих значень була однаковою. Вектор ознак складається з порівняльних ознак між аудіозаписами і саме цей вектор подається на вхід нейронним мережам агента, як поточний стан середовища. Вектор має усереднені енергетичні ознаки по діапазонах еквалайзеру, які використовуватимуться і для тренування фінальної моделі, кожного запису та різниці між ними. Далі витягуються спектральні ознаки аудіофайлів. Можна задавати самостійно кількість використаних ознак, але в даному навчанні використовувались група спектрально-акустичних ознак, такі як: мел-спектрограми у логарифмічному масштабі, MFCC, chroma-коефіцієнти (відповідає енергії напівтону у фреймах) та спектральні ознаки, такі як контраст, центроїд та плоскість спектру. Для цих ознак береться середнє значення, стандартне відхилення по кожному файлу та значення

різниці даних ознак між пар аудіо, які усереднюються. Це дає можливість перейти від сирих часових даних до зручного для нейромереж числового вектора. Функція повертає вектор ознак, який потрапляє на вхід моделі, як відповідає за певний параметр еквалізації.

Середовище створює початковий стан представлений даним вектором на основі пари записів і застосовується дія (вибір параметрів еквалайзеру), де агент обирає індекси, які декодуються в конкретні параметри фільтрів. Після застосування середовище отримує метрику якості і повертає нагороду та новий стан сирого запису.

Простір дій даної моделі є дискретним. Для цього було створено окремий клас EQActionSpace, який містить у собі масиви всіх можливих дискретних значень. Для кожного з чотирьох фільтрів є три діапазони параметрів з фіксованим кроком, налаштовані відповідно до того, як вони будуть використовуватись в плагіні. В цьому класі є два важливі методи. Перший повертає словник для агента з параметром та кількістю можливих індексів для цього параметру, щоб будувати відповідні Q-виходи. Інший метод створений для декодування словника з індексами дій обраних агентом в масиві дискретних значень. В результаті повертається інший словник, але вже з конкретними реальними значеннями, який використовується для налаштування фільтрів, які застосовуються до аудіофайлу, що обробляється.

Агент використовує моделі для обрахунку Q-значень, тобто вигідність вибраної дії, ймовірна нагорода за її виконання. Для кожного параметра агент має нейромережу для передбачення Q-значення для кожної дії з дискретного простору, що означає вихідний шар мережі буде такої розмірності, як і кількість дискретних дій. Це можна розглянути, як ансамбль нейронних мереж. Архітектура схожа на модель при виборі фінальної моделі для передбачення параметрів еквалайзера: послідовна, багатошарова, повнозв’язна модель. Вона також має чотири прихованих шари, звужуючу архітектуру і структуру: лінійний шар, нормалізація,

функція активації і регуляризація. В даній моделі використано LayerNorm, який нормалізує вектори всередині одного зразка (на відміну від BatchNorm, що по батчу), що дозволяє уникнути помилки з розмірами батчів. Для початкової ініціалізації ваг використано He-ініціалізацію, що підходить для мереж з ReLU активацією та допомагає зберегти стабільність дисперсії активацій і градієнтів під час навчання, а зсуви (bias) ініціалізовано, як нуль [60].

Основна та цільова мережі мають однакову архітектуру, але ваги можуть відрізнятися в процесі навчання, бо лише основна модель оновлюється після кожного кроку, а цільова – ні. Ці дві моделі використовуються для реалізації техніки Double DQN. Ця техніка спрямована на усунення проблеми переоцінювання Q-значень, яка часто виникає у звичайному DQN. У DQN максимізація Q-значень у формулі оновлення призводить до систематичного переоцінювання дій через те, що одна й та сама мережа використовується і для вибору дії, і для оцінки її якості. Ідея даної техніки в тому, щоб розділити процес вибору дії (основна мережа) і оцінки її якості (цільова мережа) між двома окремими мережами. Це дозволяє зменшити упередженість максимізації і стабілізувати навчання [61].

Використано м’яке оновлення цільової мережі, яке відбувається кожні 100 кроків (не епізодів) навчання. Така стратегія є достатньо поширеною у проєктуванні агнетів і дозволяє [62] поступово оновлювати ваги цільової мережі шляхом плавного змішування її параметрів з параметрами основної мережі, замість того, щоб повністю копіювати ваги основної мережі, що зменшує нестабільність навчання.

Для створення пріоритезованого буферу визначено окремий клас PrioritizedReplayBuffer, що може містити в собі до 20000 перехідних станів для багаторазового використання попередніх прикладів. Коли стан після кожного кроку, коли агент обрав дію й отримав винагороду, потрапляє у буфер він отримує пріоритет, рівний поточному максимальному. Це

зроблено для того, щоб агент більше навчався на нових, актуальних станах. На відміну від звичайного буфера, в якому кожний стан має рівну ймовірність бути обраним для навчання, у пріоритезованому буфері для формування батчу обчислюється ймовірність вибору кожного переходу, яка обрахована, як значення пріорітету піднесене до ступеню α, яке складає 0.6 поділене на їх суму. За допомогою цієї розподіленої ймовірності обирається встановлена кількість екземплярів (у випадку даної моделі 128) для навчання моделі. Після виконання кроку навчання нові пріоритети (на основі TD-помилки) оновлюються, і якщо у переході буде велика помилка, то він буде обиратися частіше. Окремо обраховується значення вагів важливості прикладів, оскільки ймовірність вибору переходу з великою помилкою більше, то це викривляє статистику і модель може переобладнатися під ці часті приклади. Щоб так не сталося станам обраним з високою ймовірністю зменшується вплив помилки в функції втрат. Натомість приклади з меншою ймовірністю будуть впливати на втрату більше [63].

Нагорода для дії агента обрахована в окремій функції і є комплексно, що складає одразу декілька метрик, щоб оцінити наскільки агент покращив звук, наблизився до референсного файлу завдяки своїм діям. Основна логіка для всіх метрик однакова: обраховується значення покращення, наближення між обробленим записом і референсним записом. До прикладу, обрахується спектральна відстань між сирим та референсним файлами та обробленим та референсним файлами, воно вимірюється як середньоквадратична помилка між спектрограмами в децибелах. Відповідно, якщо значення помилки між обробленим та референсним записами стала менша ніж значення між сирим та референсним записами, то запис наблизився до зразкового. Значення покращення обраховано, як відношення різниці дистанцій до дистанції між сирим та референсним треками. Таким чином, якщо значення покращення більше нуля, то звук дійсно став ближчим до еталонного. Окрім спектральної дистанції

використано перцептивну схожість, яка враховує MFCC з ДКП, середньоквадратичну помилку мел-спектрограми та багатомасштабну спектральну збіжність, які показують відстань між записами і перетворюються у значення схожості (чим більше значення, тим більша схожість). Ці метрики об'єднанні у співвідношенні між собою, оскільки вони можуть дати метрику, яка узагальнює слухове сприйняття людини, яке врахують і зміну тембру, і енергії, і спектральні провали. Інші три метрики також обчислюються схожим чином і було розглянуто при виборі ознак для тренування фінальної моделі: баланс енергії по частотним діапазонам, спектральний центроїд та плоскість. Всі ці значення покращення об'єднано у винагороду, яка наведена у формулі 3.1.

$$R(s,a) = \min\left(\max\left(\begin{matrix}0.25 * \Delta_{spectral} + 0.30 * \Delta_{perceptual} + \\ 0.20 * \Delta_{balance} + 0.15 * \Delta_{centroid} + 0.1 * \Delta_{flatness}\end{matrix}, -1\right), 1\right), \quad (3.1)$$

де $R(s,a)$ – нагорода агента за дію a у стані s;

Δ – значення покращення метрики.

Таким чином отримана функція описує зміни і схожість між записами використовуючи різні ознаки. Значення нагороди також обмежено діапазоном, що важливо для стабільності навчання.

Оскільки всі частини було розглянуто, можна повністю ознайомитись з агентною моделлю. Компонентами агенту є вже вище перелічені буфер досвіду, Q-функція, яка апроксимується нейромережею для певного параметру, цільова нейромережа, політика та процедура навчання.

Вибір дії агента (select_action) є методом, який вирішує, як із поточного стану отримати індекси для всіх параметрів еквалайзера. На вхід він отримує вектор числовиї ознак, які описують поточний стан. Для вибору дії було використано політику ε-жадібної стратегії з додаванням шуму [64]. Шум гарантує, що агент час від часу відхиляється від своєї поточної стратегії, дозволяючи йому накопичувати різноманітний досвід. ε-жадібна

стратегія полягає в наступному: з ймовірністю ε (спочатку дорівнює одиниці і поступово зменшується) агент просто обирає випадкові індекси параметрів, що дозволяє досліджувати простір дій. В іншому випадку агент експлуатує (exploitation) те, що вже навчився, шляхом обчислення Q-значення в усіх дискретних діях і бере ту, яка гарантує максимальне значення. Якщо агент досліджує простір, то є два варіанти: з ймовірністю 0.7 агент просто обирає випадковий індекс, інакше агент прораховує вектор Q-значень, обирає індекс дії за найбільшим, але додає невеликий шум – випадкове ціле число в певному діапазоні. В цьому і полягає стратегія зашумленого дослідження, і не дивлячись на те, що прогноз не випадковий, в нього додається шум для більшої досліджуваності. В результаті цього метод повертає словник з індексом параметру, які потім перетворюються у реальні значення параметрів.

Центральним методом агенту є метод learn, який і реалізує всі вищеперераховані техніки. Він викликається кожного разу, коли агент додав новий перехід у буфер, але реальне оновлення мереж відбувається не одразу, а тільки коли в буфері накопичилося достатньо зразків (у випадку даної реалізації розмір батчу містить 128 переходів). На початку процесу навчання з буферу береться батч з кортежами списків станів, дій, нагород, вагів станів, які перетворюються в тензори. Після підготовки даних починається основний цикл, який проходиться по кожній мережі. Спочатку використовується DoubleDQN метод, в якому спочатку зі основної мережі обирають індекси наступних дій, а потім із цільової мережі беруть відповідні Q-значення. Це зменшує переоцінювання Q-значень. В результаті формується цільове Q-значення, шляхом додавання нагороди до Q-значення з цільової мережі. Тобто, агент розуміє, що ця нагорода є максимальною при правильному підборі індексів дій і до неї потрібно прагнути. Обчислюється поточне значення Q і формується значення TD-помилки, яке обраховується як різниця між цільовим значенням Q та поточним, передбаченим основною мережею. Якщо помилка велика, це значить, що поточна мережева

апроксимація дуже відрізняється від того, що модель спостерігає в середовищі разом із цільовою мережею. Ці помилки буде використано для оновлення станів у буфері, і якщо значення помилки велике, то шанс бути обраним станом у наступному батчі стає вищим. Обчислюється помилка, як середнє значення добутку вагів, які компенсують втрату, якщо приклад має високий пріоритет та MSE між цільовим та поточним Q-значенням.

Далі йде процес оновлення вагів в мережі. Очищуються попередньо накопичені градієнти і обчислюються нові, тобто відбувається зворотне поширення помилки, коли для кожного параметра моделі обчислюється градієнт, який показує, як зміна цього параметра впливає на значення функції втрат. Якщо норма градієнтів занадто велика, її буде обмежено задля уникнення вибуху градієнтів. Після цього оптимізатор AdamW на основі знайдених градієнтів оновлює ваги моделі параметру в напрямку мінімізації похибки. Також використано планувальник швидкості навчання, яка зменшує швидкість навчання, якщо функція втрат не зменшується, щоб запобігти стогнації.

Після проходження основного циклу оновлюються значення пріоритету у буфері за індексами, які були у батчі для навчання. Якщо лічильник досягнув цільового значення, то цільова мережа оновлюється відповідно до основної з механізмом м’якого оновлення. Окремо оновлюються значення ε та β (ступень компенсації при оновленні вагів у батчі).

Останнім етапом є тренування агента. Спочатку готується екземпляр агента, простір дій, шляхи до даних і параметри, які керують тривалістю навчання, умовою ранньої зупинки та допоміжні для збереження результатів. Формується список пар аудіофайлів, які було знайдено у відповідних директоріях. Створюється два словника, в яких в першому зберігаються метрики якості, узагальнена оцінка якості, найкраща нагорода за епізод та значення параметрів еквалайзеру. У другому буде формуватися сам набір даних з ознак, які обрано з необробленого аудіофайлу та

параметрів еквалайзера, які найкраще обробляють сирий запис до референсного. Основний цикл йде по епізодах, в кожному з яких обирається випадкова пара аудіофайлів. При завантаженні файли зчитуються відповідно до частоти дискретизації, нормалізуються та приводяться до однакової довжини. Формується початковий стан, який репрезентовано вектором ознак двох файлів та ініціалізовуються локальні змінні.

В циклі по кроках в кожному епізоді, агент обирає дію беручи вектор ознак і повертаючи індекси дій, які обираються згідно розглянутої політики. Індекси перетворюються у реальні значення і ці значення застосовуються до сирого аудіо за допомогою відповідних фільтрів еквалайзеру. Спектрограми сирого, референсного та найбільш схожого до референсного, обробленого аудіофайлів можна побачити на рисунку 3.3.

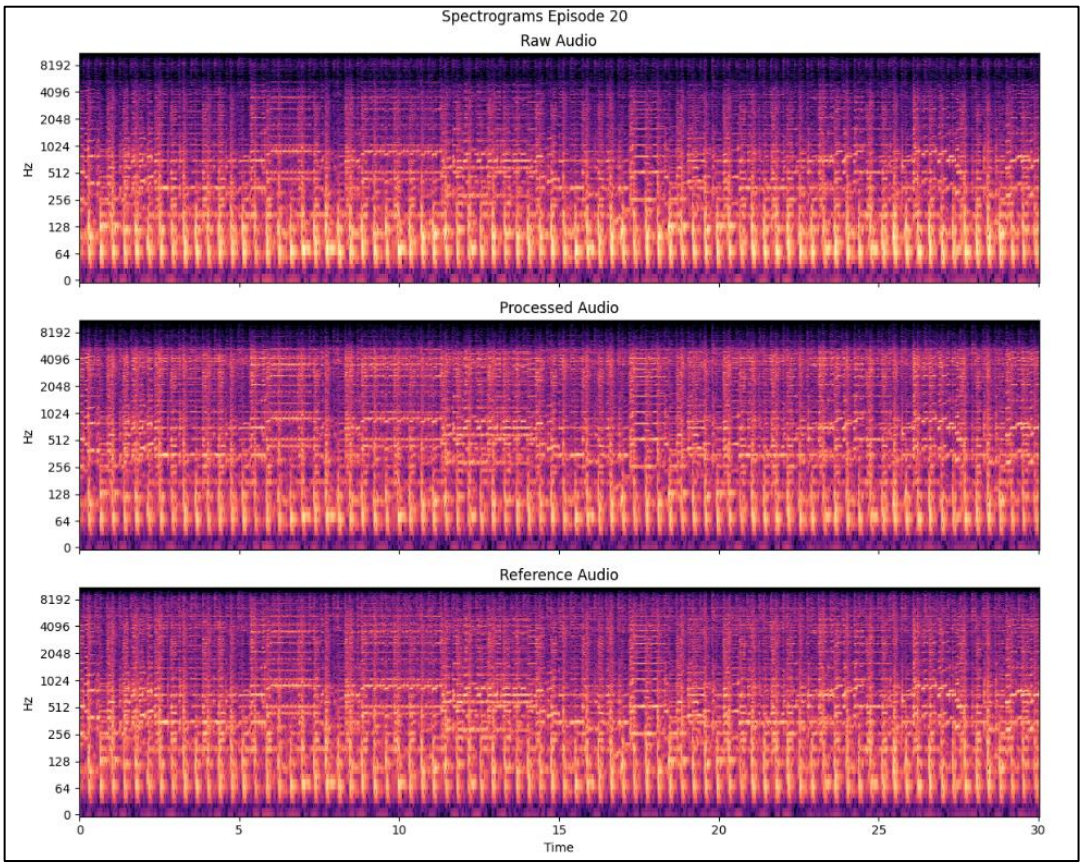

Рисунок 3.3 – Спектрограми сирого, обробленого та референсного файлів

Далі обчислюється метрика нагороди і якщо вона найкраща за попередню, оновлюються відповідні значення нагороди та параметрів еквалайзера. Будується наступний стан агента на обробленому аудіо і все це зберігається у буфері. Після чого відбувається оновлення агента функцією learn, розглянутою вище, що оновлює ваги основних мереж агента і відбувається перехід до наступного стану. Після завершення епізоду збирається відповідна статистика та будуються графіки. Також наявна техніка ранньої зупинки, якщо якість не покращується. Приклад епізоду тренування агенту наведено на рисунку 3.4.

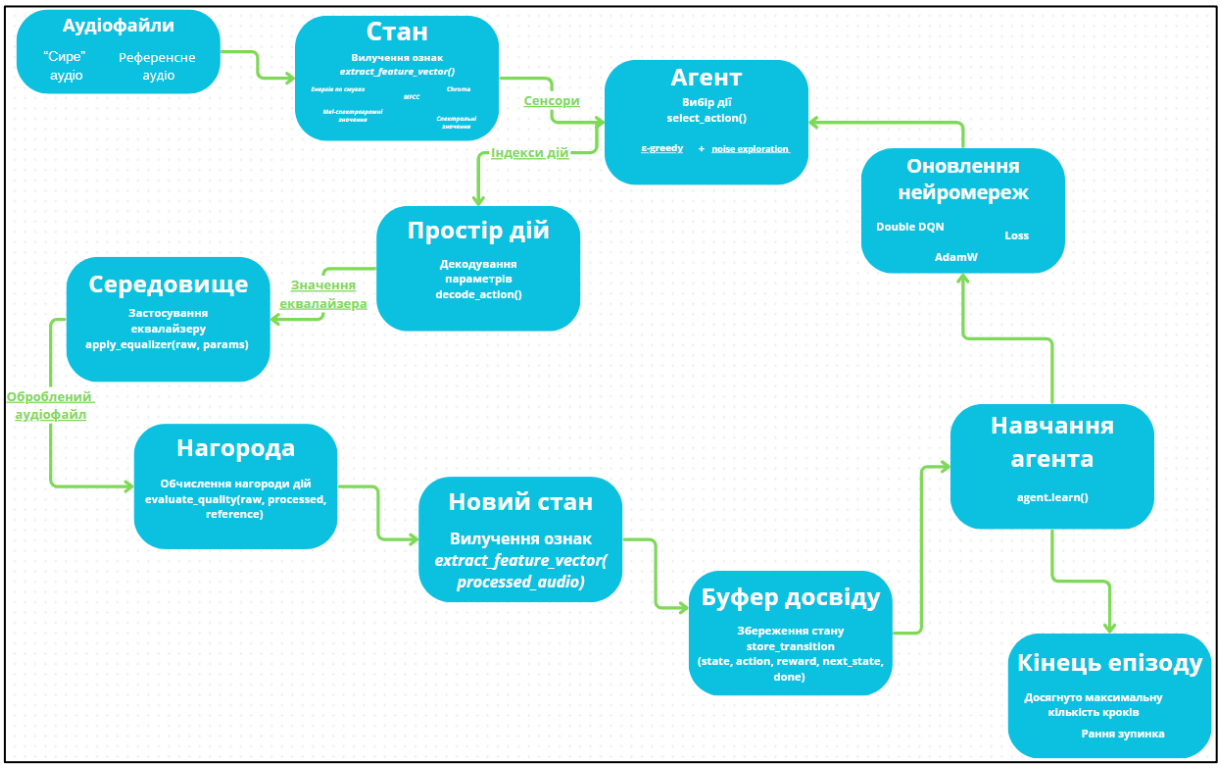

Рисунок 3.4 – Приклад епізоду навчання

Тренування запускалося з параметрами 50 епізодів по 30 кроків. Зміна нагород агента при тренуванні зображена на рисунку 3.5, а значення похибки по всіх моделях на рисунку 3.6. Додатково показано графік зміни значення ε по відношенню до кількості кроків в епізодах на рисунку 3.7.

Рисунок 3.5 – Графік нагород агента

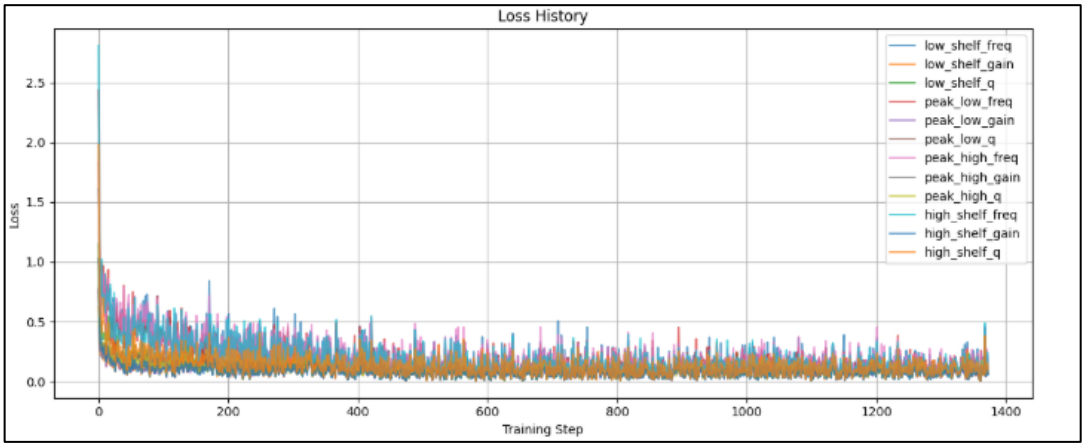

Рисунок 3.6 – Графік функції втрат для кожної моделі

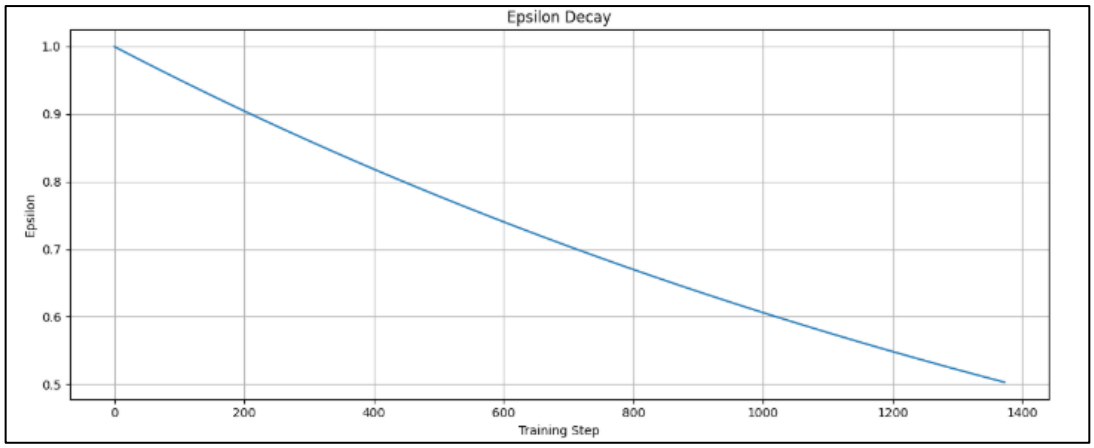

Рисунок 3.7 – Зміна значення ε на тренуванні агента

З огляду на результати можна сказати, що нагороди не є стабільними при навчанні. Хоч більшість результатів нагород є більше нуля, іноді значення різко змінюються, що можна пояснити або вибором поганих індексів дій моделі, через стратегію, або через аудіозапис, який може значно відрізнятися від попередніх, до яких модель звикла. Значення втрат є невисокими і стабільно знижуються і знаходяться зі значенням нижче 0.5, що є гарним результатом. Значення ε лінійно падає, і на момент 50 епізоду політики обирала кожну другу дію випадково.

3.3 Реалізація моделі для передбачення параметрів еквалайзеру

В методах вирішення задачі було зазначено, що як фінальну модель для інтеграції в плагін буде використано одну з моделей нейронних мереж, відкинувши розглянуті варіанти щодо ансамблевих методів, які показували гірші результати. Для інтеграції в плагін було обрано модель з чотирьох шарів з пакетною нормалізацією. Вибір зумовлений швидкістю моделі та меншою вагою. Також в ході проведення перетренувань моделей виявилося, що дана модель може показувати себе навіть краще, аніж модель з шаром уваги. Тому такий вибір зберігає, і якість, і швидкодію.

Реалізовано архітектуру нейронної мережі було за допомогою бібліотеки pyTorch. Ця бібліотека має гарну підтримку при експорті моделей, що стало одним з ключових факторів її вибору, а також повний інструментарій для створення різноманітних архітектур. Також використовувалися відомі бібліотеки pandas та numpy для обробки CSV-файлів та масивів, matplotlib для побудови графіків. Для розділення набору даних на тренувальну та тестову вибірки, обрахунку метрик для моделі та нормалізації вхідних ознак було використано бібліотеку scikit-learn, яка надає повний інструментарій для аналізу даних та створення моделей МН. Для експорту та перевірки роботи моделі використовувались бібліотеки ONNX та onnxruntime завдяки яким і буде інтегровано модель до плагіну.

На початку тренування моделі було завантажено дані. Екземплярів даних для тренування моделі було використано 15 тис. Відповідно цей набір мав 52 ознаки, з яких 40 є вхідними ознаками, які було розглянуто при виборі моделі (спектральні характеристики, тощо), решта – цільові ознаки для кожного параметру еквалайзеру. При підготовці даних не було виявлено пустих значень. Підготовлені дані, перед використанням їх в моделі, нормалізуються. Нормалізація є практично обов'язковою процедурою для навчання більшості моделей нейронних мереж. У випадку ознак, які було створено, вони мають досить різні масштаби, тому є обов'язковим етапом приведення їх до єдиного діапазону, щоб не спотворювати навчання мережі, і градієнтий спуск разом з вагами мережі оновлювалися більш рівномірно, що сприяє стабільнішій збіжності, а також добре поєднується з такими методами, як пакетна нормалізація [65]. Тому для нормалізації вхідних даних використовувався RobustScaler, а для нормалізації цільових значень StandardScaler. RobustScaler менш чутливий до викидів, що важливо для аудіоданих, які можуть мати екстремальні значення.

Архітектура нейронної мережі є багатошаровою та повнозв'язною. Її призначенням є передбачення числових параметрів для налаштування еквалайзеру, тобто вирішення задачі регресії. Вона має п'яти прихованих шарів та вихідний шар.

В даній архітектурі присутні п'ять прихованих шарів разом з вхідним та вихідним шаром. Така архітектура дасть змогу моделювати нелінійні залежності високої складності. Основою є лінійний шар, який виконує лінійне перетворення вхідних даних: вагові коефіцієнти множаться на вхідні значення, до суми додається зсув. Такий шар є базовим елементом повнозв'язних нейронних мереж. Сам по собі він не вводить нелінійності, тому для побудови складних моделей зазвичай після нього застосовують функції активації. В першому шарі використано 512 нейронів.

Перед застосуванням функції активації використовується пакетна нормалізація – техніка, яка покращує швидкість і стабільність навчання

нейронних мереж шляхом нормалізації вхідних даних для кожного шару. Вона запроваджує додаткові операції у мережу, щоб зменшити внутрішній коваріаційний зсув (зміни розподілу вхідних даних для шару під час навчання). Це прискорює навчання і дозволяє використовувати більший крок навчання, а також мережа стає менш чутлива до початкових налаштувань. Нормалізація використана після кожного шару, крім останнього.

Як функція активації використано ReLU. Це доволі популярна функція. Вона вводить нелінійність, необхідну для навчання складних патернів і є дуже простою у реалізації, а тому швидка в обчисленні. Якщо до неї надходить від'ємне вхідне значення, то вона повертає нуль, якщо ні – повертає число, яке прийшло на вхід. На відміну від сигмоїди чи tanh, ReLU не має проблеми затухаючого градієнта на додатній частині. Проблема необмеженого виходу, коли вихід додатного значення через функцію активації не змінюється, а значить може зростати без верхньої межі, вирішено саме впровадженням пакетної нормалізації перед нею. Це допоможе усунути проблему з дуже великими числами у внутрішніх шарах нейромережі.

Шар регуляризації Dropout застосовується після функції активації. Це також є поширеною технікою для запобігання перенавчанню. Під час навчання шар випадковим чином відключає певний відсоток нейронів у мережі з ймовірністю виставленою ймовірністю. Ці нейрони тимчасово не беруть участі у прямому проході та оновленні ваг, і це змушує не залежати надмірно від окремих нейронів. Рівень Dropout зменшується по мірі зменшення розмірності шарів. На ранніх шарах багато нейронів, і ймовірність перенавчання вища, тому використано більший коефіцієнт. На глибших шарах, в яких поступово зменшується розмірність, використано менший коефіцієнт, бо там менше нейронів.

Останній шар лінійний і переводить вектор розмірності 64 у 12 виходів під кожний параметр еквалайзера. Оскільки задача цієї мережі є

регресія, то функції активації не застосовуються, щоб значення були без обмежень. Архітектуру нейромережі зображено на рисунку 3.8.

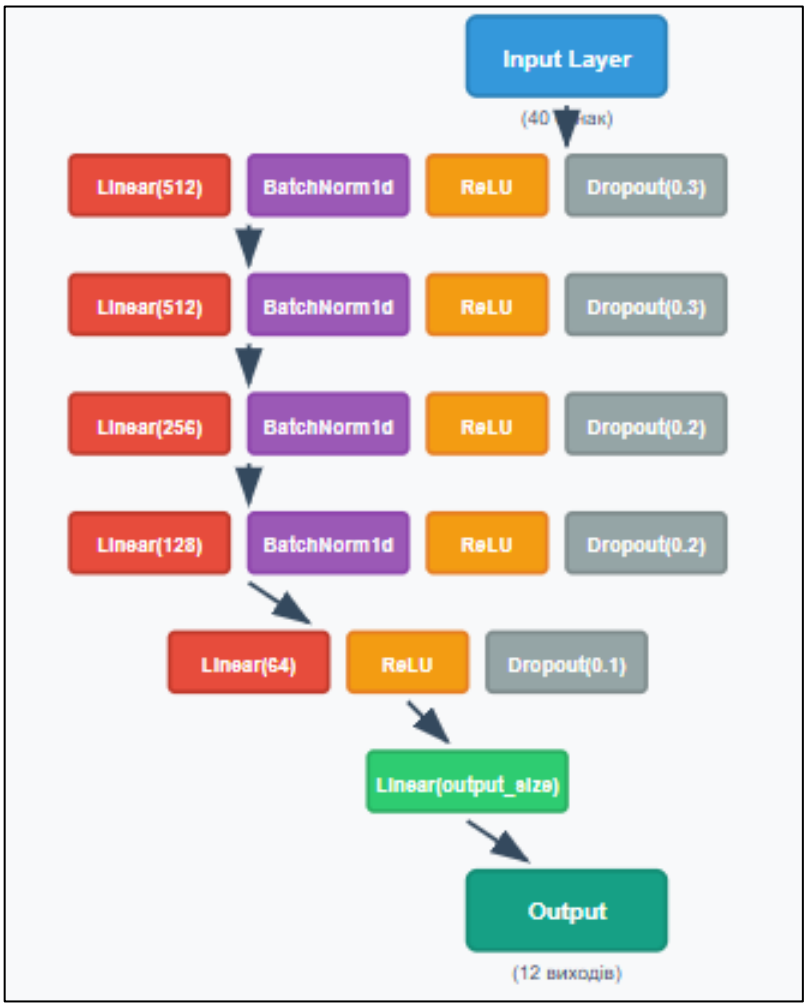

Рисунок 3.8 – Архітектура нейронної мережі

Дана архітектура гарно підходить до задачі, має достатню кількість шарів та нейронів, щоб уловити складні залежності між ознаками та цільовими мітками і може підходити до різних наборів ознак. Навчання відбувається на тренувальній вибірці разом з перевіркою на валідаційній вибірці. Тренування відбувалось з використанням GPU, що дозволяє прискорити навчання, оскільки графічні процесори мають тисячі ядер, оптимізованих для одночасного виконання великої кількості простих операцій, зокрема матричних множень і векторних обчислень, які є основою

нейронних мереж [66]. При тренуванні використана комбінована функція втрат в яку входять метрики MSE (середньоквадратична помилка) та MAE (середня абсолютна помилка). Функція MSE є чутливою до великих відхилень, оскільки квадратичне зростання втрати при значних помилках істотно впливає на загальний результат, підвищуючи роль поодиноких викидів. Натомість MAE забезпечує більш рівномірну реакцію на помилки середнього масштабу та є стійкішою до викидів. Проте вона характеризується меншою чутливістю поблизу нуля, що може зумовити уповільнення оновлення ваг при малій похибці. Саме тому поєднавши ці значення, отримано компроміс, де якщо є дані зашумлені, то абсолютна помилка пом’якшить функцію втрати.

Оптимізатором для навчання було обрано AdamW. На відміну від звичайного Adam, який включає регуляризацію (L2) ваг безпосередньо в процес оновлення градієнтів, даний оптимізатор відокремлює регуляризацію (weight decay) ваг від кроку оновлення градієнтів. Це дозволяє уникнути некоректного застосування регуляризації, яке в Adam може призводити до менш ефективного запобігання перенавчанню. Також це дозволяє точніше контролювати регуляризацію допомагає підтримувати стабільність протягом усього процесу навчання [67].

Додатково використано регулювання швидкості навчання (CosineAnnealingWarmRestarts). Основна його ідея в тому, що замість лінійного або степеневого спаду швидкості навчання (learning rate), застосовується косинусна траєкторія. Це забезпечує поступове, гладке зменшення швидкості навчання від максимального значення до мінімального, що допомагає уникнути різких стрибків і стабілізує процес оптимізації. Кожний пройдений цикл спадання швидкості навчання, вона відновлюється до початкової і цей цикл стає довшим. Для даної моделі один цикл складає 50 епох, які в подальшому стають в два рази довшими. Така корисна техніка дозволяє не застрягати в локальних мінімумах і повторно

відновлювати швидкість, шукаючі кращі мінімуми функції втрат, особливо при великій кількості епох.

Для обмеження норми градієнтів використано Gradient Clipping – це техніка, яку використовують під час навчання нейромереж для запобігання проблемі вибухаючих градієнтів, коли їхні значення під час зворотного поширення помилки зростають експоненційно. Це може призводити до нестабільного оновлення ваг і руйнувати навчання моделі. У глибоких мережах, при великій кількості шарів таке може відбуватись через багаторазове перемноження похідних функцій активації. Щоб запобігти цьому використовується ця техніка, яка не шкодить процесу, оскільки якщо градієнти мають малі значення, то нічого не зміниться.

Крім того використано ранню зупинку навчання, якщо значення функції втрати на валідації не покращується, тренування припиняється, щоб уникнути перенавчання або марної витрати ресурсів.

Після тренування, модель оцінюється на тестовому наборі. Оцінки по кожній вихідній ознаці наведено на рисунку 3.9. Значення статистики R^2 для моделі загалом зображено на рисунку 3.10.

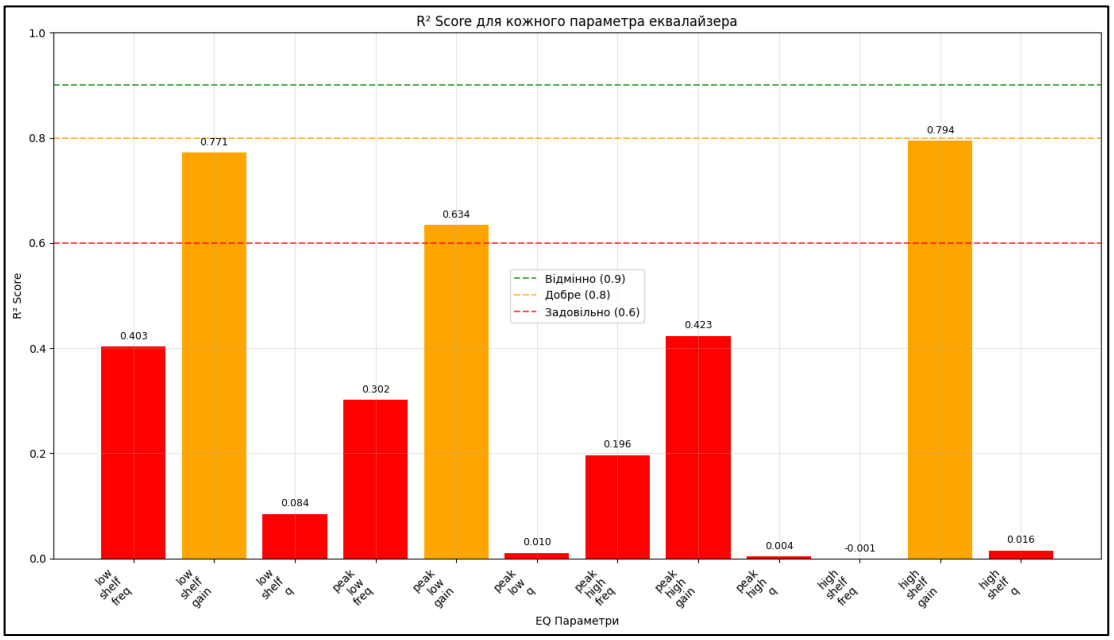

Рисунок 3.9 – R^2 оцінка для кожного параметра

```
   Статистика R² по параметрах:
-------------------------------------
Середній R² (нормалізовані): 0.3031
Мінімальний R²:              -0.0011
Максимальний R²:             0.7943
Стандартне відхилення R²:    0.2893
```

Рисунок 3.10 – Статистика моделі за метрикою R^2

За результатами передбачення, яку наведено на рисунку 3.11, різниця між деякими параметрами велика, що відповідає метриці R^2. Варто зазначити, що хоч результат передбачення добротності був низьким, різниця між значеннями є не дуже великою спираючись на тестування декількох прикладів вона коливалася у межах значень 0.2–0.3. Натомість параметри частоти в результаті тестувань коливаються іноді досить у великому діапазоні, що є погано, особливо у середніх частотах, де звук насичений і сприймається людиною більш гостро, на відміну від високих, пам’ятаючи про те, що слух людини логарифмічний.

Параметр	Оригінал	Передбачення	Різниця
low_shelf_freq	423.500	410.644	12.856
low_shelf_gain	8.900	8.461	0.439
low_shelf_q	1.470	1.188	0.282
peak_low_freq	4739.700	2446.954	2292.746
peak_low_gain	-0.400	-0.880	0.480
peak_low_q	1.110	1.105	0.005
peak_high_freq	10030.400	9411.877	618.523
peak_high_gain	-6.000	-0.676	5.324
peak_high_q	1.390	1.127	0.263
high_shelf_freq	13583.000	16043.652	2460.652
high_shelf_gain	-3.200	-0.508	2.692
high_shelf_q	1.470	1.092	0.378

Рисунок 3.11 – Результат передбачення параметрів еквалайзера

Останнім важливим етапом є експортування моделі в ONNX формат. Одразу постає проблема з нормалізацією вхідних даних, оскільки модель очікує саме такі ознаки на вхід, а після отримання результату також потрібно повернути дані до нормального масштабу. В такому разі при інтеграції моделі доведеться повторно імплементувати скелери. Щоб цього не робити і не мати додаткових залежностей в плагіні, потрібно додати склери всередину єдиної моделі ШІ. Для цього створено окрему обгортку зі скелерами. В ній визначається відповідний скелер і реєструється, як буфер – це такий тензор, який входить у стан моделі, але не буде мати градієнта. У підсумку в моделі буде отримано постійні тензори з нормалізацією на вході. Після нормалізації, ознаки передаються базовій мережі. Відповідно, щоб отримати вихідні параметри в правильному масштабі потрібно денормалізувати дані.

Модель експортовано за допомогою окремого метода. В ньому є багато налаштувань, зокрема модель експортована на CPU, щоб уникнути можливих проблем із GPU-залежностями. ONNX-експортувач потребує зразковий вхід із правильною формою, в якому вказуємо розмірність вхідного тензору, а також назви вхідного та вихідного тензорів. Всі параметри мережі, як наприклад ваги, будуть всередині моделі, тому немає потреби в окремих файлах. Версія для експорту була обрана 13, яка не є новою, але підтримує всі потрібні операції та підтримується рантаймом. Сам механізм експорту проводить попередню оптимізацію, виконавши всі зведення сталих підвиразів, що зменшує розмір.

Після експорту ONNX модель була протестована на працездатність та порівняна зі звичайною моделлю на тестових даних. Для цього створюється нова сесія для імпортованої моделі і подається на вхід масив numpy, оскільки ONNX працюють саме з ними. Після отримання результатів передбачення вони порівнюються з оригінальною PyTorch моделлю. В ході тестування значних розбіжностей між моделями не виявлено, що показано

на рисунку 3.12. Модель генерує варіативні результати – це вказує на те, що модель не застрягла, і всі параметри знаходяться в очікуваних діапазонах.

Параметр	ONNX	PyTorch	Різниця
eq_low_shelf_freq	61.0092	61.0092	0.000019
eq_low_shelf_gain	-15.6990	-15.6990	0.000001
eq_low_shelf_q	1.5454	1.5454	0.000000
eq_peak_low_freq	2004.9464	2004.9474	0.000977
eq_peak_low_gain	-7.1621	-7.1621	0.000000
eq_peak_low_q	1.0876	1.0876	0.000000
eq_peak_high_freq	9045.0879	9045.0879	0.000000
eq_peak_high_gain	1.4775	1.4775	0.000001
eq_peak_high_q	1.0959	1.0959	0.000000
eq_high_shelf_freq	15692.5195	15692.5186	0.000977
eq_high_shelf_gain	-6.6432	-6.6432	0.000002
eq_high_shelf_q	1.1366	1.1366	0.000000

Рисунок 3.12 – Порівняння результатів роботи моделей

Дана модель пройшла всі тестування на коректність і готова до імпорту в плагін за допомогою ONNX-рантайму. Фінальний розмір склав 1.77 Мб.

3.4 Інтегрування моделі в плагін

Після тестування працездатності моделі в Python за допомогою ONNX Runtime, було перейдено до основного проєкту плагіну, реалізованого на C++.

ONNX – це бібліотека, яка є кросплатформеним високопродуктивним рушієм для виконання інференсу моделей МН у форматі ONNX. Для використання, її було завантажено через менеджер пакетів NuGet в проєкт. Після завантаження, у програмі Projucer, яка використовується для управління проєктом, вказано шляхи до даної бібліотеки, щоб при збірці плагіну фреймворк її бачив у області дій і зміг виконувати та збирати

відповідний код. Також необхідно завантажити саму модель у плагін у вигляді бінарного файлу і при використанні додати до неї шлях в проєкті.

Після виконання цих дій, у коді створюється глобальний контекст логування, налаштовується оптимізація і кількість потоків та вбудовується сам об’єкт інференсу. На вхід також потрібно підготувати відповідні тензори і створити сесію та запустити сесію для роботи моделі. Коли модель спрацьовує, вона отримує відповідний вектор з параметрами еквалайзера, які передаються у функцію, щоб їх встановити.

Оскільки інференс може працювати певний час, особливо складні моделі, важливо зробити можливість його роботи паралельною, щоб не затримувати роботу основної програми. Також, щоб покращити роботу моделі можна зафіксувати розміри тензорів, щоб прискорити часову оптимізацію.

Іншим етапом інтеграції є підготовка вхідних ознак для тензору на вхід моделі. В коді для підготовки відповідного набору даних використовувалися ознаки, які обчислені за допомогою бібліотеки librosa. Альтернативою могла би стати подібна бібліотека, але не всі бібліотеки на C++ мають певні з методів для вилучення ознак, хоча основні ознаки у вигляді мел-спектрограм, MFCC у більшості є. До того ж у бібліотеках можуть міститись відмінності у реалізації обчислення певних ознак, що може вплинути на результат передбачення. Саме тому основним рішенням стало використати функцію з Python у C++. Це достатньо складний процес, оскільки збільшується навантажуваність, вага плагіна і ускладнюється його розгортання та процес інтеграції. Для вбудови функції є декілька підходів, один з яких використання інтерпретатора Python у код плагіна. Це громіздкий процес, оскільки потребує знання технології, а обсяг коду збільшується.

Для спрощення цього процесу було використано бібліотеку pybind11, яка дозволяє більш легко інтегрувати функцію з меншою кількістю коду. Це легка бібліотека , що працює лише з заголовками та надає доступ до типів

C++ у Python і навпаки, головним чином для створення зв’язків Python з існуючим кодом C++. Вона дозволяє легко і з мінімальними зусиллями зв’язати (байндити) функції, класи та інші елементи C++ з Python, забезпечуючи прямий виклик C++ коду з Python без необхідності писати складний обгортковий код. Підхід подібний, тому потрібно у сам проєкт додати шляхи до відповідного модулю та самого Python. Тим не менш, обсяг коду вийшов значно меншим, виключаючи «сирий» код для роботи з інтерпретатором. Ще однією альтернативно може стати мікросервіс на Python, який буде відповідати за обчислення ознак, але архітектура також ускладнюється і може виникнути затримка через мережевий обмін, хоча варіант більш надійний.

Отже, після підключення інтерпретатору в код, скрипт з вилученням ознак імпортується і зберігається, як член класу з логікою, оскільки його потрібно часто викликати. Після цього, якщо користувач хоче викликати модель, то збирається буфер аудіоданих на 30 секунд (такі записи використовувались для вилучення ознак), для цього використано окремо створену функцію. Цей фрагмент нормалізується та передається на вхід Python-функції для вилучення ознак, де на виході отримано числовий вектор. Цей вектор перетворено на тензор на вхід моделі, який вона обробляє та видає відповідні параметри еквалайзера, що відповідно встановлюються.

Таким чином, отримано фінальний плагін, який забезпечує не тільки базову функціональність налаштування звуку, але й налаштування завдяки вбудованій моделі ШІ.

4 ВИКОРИСТАННЯ ПЛАГІНА В ЦИФРОВИХ АУДІО РОБОЧИХ СТАНЦІЯХ

4.1 Приклади використання плагіну

Роботоспроможність даного плагіну тестувалася в DAW під назвою Studio One 6 на платформі Windows 11. Оскільки плагіни досить чуттєві до помилок, то через помилку вони не тільки не зможуть працювати, а й навіть достроково завершити роботу DAW. При тестуванні плагін стабільно працював у середовищі, що може свідчити про високу переносимість до інших DAW з підтримкою VST3. Але оскільки використовувався ONNX Runtime може знадобитися його завантаження користувачем на пристрій, на якому він користується плагіном.

Даний плагін можна використовувати автономно, як базовий інструмент обробки звуку, так і з моделлю ШІ. Перший сценарій дозволяє керувати кожним з параметрів, як і в стандартних еквалайзерах. Другий забезпечує автоматизацію налаштування параметрів. Це може бути корисно у різних сценаріях використання і для різних інструментів. Можна обробляти звичайні музичні інструменти, синтезатор. Також це може бути корисним при налаштуваннях голосу для подкасту, при аналізі в реальному часі.

Варто зазначити, що даний плагін лише пропонує параметри налаштування і вони можуть бути скориговані користувачем на власний розсуд, або змінені користувачем повністю через базовий функціонал. Він є орієнтиром і його завжди можна скоригувати вручну. При аналізі музичного фрагменту бажано брати кульмінаційний момент, щоб модель зрозуміла значення максимальної енергії, гучності для треку, що буде враховано при обчисленні ознак і наданні коректних рекомендацій.

Такий плагін дозволить швидко налаштувати звук для стріму або вказати напрям еквалізації музичного фрагменту для музикантів-

початківців, які опановують даний інструмент і який забезпечить все необхідне для його початкового налаштування. При встановленні плагіну можна порівняти значення до еквалізації з використанням ШІ та після, що може бути корисним і допомогти відчути різницю та зрозуміти, що може подобатися та не подобатися у підібраних значеннях. Це забезпечується шляхом вбудови аналізатора сигналу при програмуванні плагіну.

4.2 Аналіз роботи ШІ

ШІ-помічник з налаштування безумовно має свої переваги з огляду на розвиток галузі, як музичної, так і МН.

По-перше, ШІ може проаналізувати запис миттєво і з першого разу, і за різноманітним набором ознак запропонувати оптимальні параметри суттєво зменшуючи час на ручний пошук точок поділу і регулювання підсилення, тощо. Він є доступним і зрозумілим для новачків, які користуються даним плагіном і не потребує глибоких знань з еквалізації.

З поширенням медіаконтенту ці сильні сторони знаходять застосування у масовому мікшингу, при роботі з великою кількістю подкастів, музикою для відеоконтенту, економлячи час людям, що його створюють. Тобто, плагін може бути не тільки корисним для виготовлення аудіо безпосередньо. Багато людей створюють свої власні домашні записи, але не мають знань чи бажання навчатися складним процесам налаштування, і даний плагін може визначати проблемні діапазони і одразу коригувати їх, щоб підвищити чіткість. Також для звукорежисерської практики рекомендації ШІ параметрів може бути демонстрацією принципів чи загальних закономірностей еквалізації і порівнювати автоматичне налаштування з ручним.

В плагіна є певні межі застосування. Перше з яких стосується особистих вподобань, художніх смаків, які іноді не можуть бути зведено до єдиної оптимальної кривої, оскільки ШІ пропонує базове, оптимальне

рішення, але не замінює звукорежисера. Так само для певних музичних жанрів автоматизація може втратити чи не виділити певні характерні частоти чи діапазони. Записи з кліппінгом чи артефактами можуть спричинити некоректні пропозиції, оскільки модель навчена на записах, які їх не мають.

У висновку, ШІ як інструмент-помічник у еквалізації здатен виконувати гарно виконувати задачі у стандартних сценаріях мікшування й мастерингу, при цьому прискорюючи робочий процес та рівень входження для початківців. Разом з тим його застосування доцільне тільки у поєднанні з творчим контролем людини та розумінням власних жанрових і технічних вимог.

4.3 Огляд подальших покращень

В ході вирішення задачі щодо розробки плагіну з’явилися ідеї щодо покращення функціональності, як зі сторони еквалайзера, як інструменту, так і з огляду використання ІШ.

Механічні покращення еквалайзеру спрямовані щодо інтерфейсу, додавання інтерактивного, графічного редактора кривої, збільшення кількості фільтрів, використання пресетів і можливість зберігати пресети створені ШІ.

Наступним етапом щодо розширення функціоналу є жанрова орієнтованість ШІ. Для цього можливо впровадити одну з інснуючих моделей для класифікації жанру і опираючись на даний жанр проводити еквалізацію з використанням нейромережі, яка навчена саме на файлах даного жанру. Таким чином збільшиться гнучкість і різноманітність сценаріїв використання.

Наступним є впровадження створеного агента в плагін, як модель, яка здатна підбирати параметри на основі референсного треку користувача. Трек, який користувач сам налаштував може бути використаним для

розуміння його авторського стиля еквалізації, особливо якщо робота йде переважно в одному жанрі. У такий спосіб еквалайзер стане більш персоніфікованим.

Можливе розширення архітектури нейромережі, використання типів нейромереж, які прогнозують одразу кілька варіантів еквалізації, тощо. Окремо варто зазначити, що кількість даних для нейромереж повинна бути більшою для виявлення закономірностей. Для цього можна збирати користувацькі правки щодо зміни параметрів для періодичного донавчання.

Такі покращення можуть зробити плагін більш зручним та гнучким, і пристосованим навіть для професійного налаштування за рахунок розширення функціоналу, великого набору даних.

ВИСНОВКИ

Основною метою роботи було автоматизувати налаштування параметрів еквалізації, для досягнення якої було створено плагін еквалайзера VST3 на базі штучного інтелекту, який здатний рекомендувати налаштування частоти, коефіцієнта посилення та добротності на основі аудіохарактеристик.

На відміну від основних комерційних продуктів, які мають закритий вихідний код і не пропонують жодного розуміння їхніх алгоритмів, ця робота пропонує повністю відкриту, модульну реалізацію. Плагін відповідає базовим потребам з еквалізації та додатково інтегрує модель ШІ для зручного налаштування.

Цей проєкт спирається на результати робіт в галузі інтелектуальної обробки сигналів та систем штучного інтелекту. Він використовує дослідження з вилучення ознак, DSP та застосування ШІ при роботі з аудіофайлами, що на даний момент є одним з основних шляхів досліджень у МН.

Даний проєкт пропонує нові вирішення щодо автоматизації в обробці звуку, створення аудіопродуктів та зокрема генерації даних з використанням агентної моделі для створення параметрів еквалайзеру.

Цей проєкт має перспективність щодо подальшого покращення і дослідження використання ШІ в музиці. Це як покращення моделей для прогнозування параметрів, створення більших наборів даних з додатковими ознаками. Впровадження системи класифікації жанру і еквалізації на основі даних конкретного жанру.

Плагін та його вихідний код можна включити до курсів з цифрової обробки сигналів та аудіоінструментів на основі ШІ. Набір даних, навчені моделі та приклад коду JUCE слугують практичними навчальними матеріалами, що сприяють розумінню студентами як реалізації DSP, так і інтеграції машинного навчання в реальні аудіозастосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Терміни. *Офіційний сайт ТОВ "Свен Центр"*. URL: https://sven.ua/ua/service/glossary/ukr/6/ (дата звернення: 12.05.2025).

2. Marven C. A simple approach to digital signal processing. Oxford : Texas Instruments. 235 с.

3. Ужинський М. Ю. Основи цифрового звукозапису : метод. вказівки до виконання практ. та самост. роботи для студентів спец. 025 «Муз. мистецтво» освітньої програми «Муз. мистецтво. Комп'ютерно-електронна музика» / Рівнен. держ. гуманіт. ун-т, І-т мистецтв, Каф. естрадної музики. Рівне: РДГУ, 2022. 37 с.

4. DSP: advantages and disadvantages you need to know!. *RF Wireless World | Your Comprehensive Resource for RF, Wireless, Telecom, and Electronics*. URL: https://www.rfwireless-world.com/terminology/fpga-dsp/advantages-and-disadvantages-of-dsp (дата звернення: 12.05.2025).

5. A beginner's guide to digital signal processing (DSP) | analog devices. *Mixed-signal and digital signal processing ICs | Analog Devices*. URL: https://www.analog.com/en/lp/001/beginners-guide-to-dsp.html (дата звернення: 12.05.2025).

6. O'Mara W. J., Susskind A. K. Notes on analog-digital conversion techniques. *Mathematical tables and other aids to computation*. 1958. Т. 12, № 64. С. 317. URL: https://doi.org/10.2307/2002430 (дата звернення: 12.05.2025).

7. Nick Messitte. Principles of equalization: how to EQ your mix. *iZotope*. URL: https://www.izotope.com/en/learn/principles-of-equalization.html (дата звернення: 12.05.2025).

8. Bores Signal Processing - Introduction to DSP - IIR filters - digital filter equation. *Wayback Machine*. URL: https://web.archive.org/web/20160321174258/http://www.bores.com/courses/intro/iir/5_eq.htm (дата звернення: 12.05.2025).

9. Dobrev D. P., Neycheva T. D. Intuitive approach to active digital filter design. part IV: principle of active biquads. *2024 XXXIII international scientific conference electronics (ET)*, м. Sozopol, Bulgaria, 17–19 верес. 2024 р. 2024. С. 1–6. URL: https://doi.org/10.1109/et63133.2024.10721547 (дата звернення: 12.05.2025).

10. AAX/AU/VST audio plugins from black rooster audio. *Black Rooster Audio*. URL: https://blackroosteraudio.com/en/blogreader/a-brief-history-of-equalization (дата звернення: 12.05.2025).

11. Pq R. What are the different types of EQ and filters?. *403 Forbidden*. URL: https://www.iconcollective.edu/types-of-eq (дата звернення: 12.05.2025).

12. The digital audio workstation and the change it gave to music. *lukethines*. URL: https://lukethines.wordpress.com/2013/06/13/the-digital-audio-workstation-and-the-change-it-gave-to-music/ (дата звернення: 12.05.2025).

13. What is automation in music? A complete guide - blog | splice. *Blog | Splice*. URL: https://splice.com/blog/what-is-automation-in-music/ (дата звернення: 12.05.2025).

14. Emerging technologies and trends in the sound industry | tecnare sound systems. *Tecnare Sound Systems*. URL: https://www.tecnare.com/article/emerging-technologies-and-trends-in-the-sound-industry/ (дата звернення: 12.05.2025).

15. Essential sound effects tools for AAA game development in 2024. *AI Voice Generator | Advanced Text-to-Speech (TTS)*. URL: https://www.respeecher.com/blog/essential-sound-effects-tools-for-aaa-game-development-in-2024 (дата звернення: 12.05.2025).

16. Innovative sound design: trends shaping the future of music production - orange candy music. *Orange Candy Music -*. URL: https://www.orangecandymusic.com/innovative-sound-design-trends-shaping-the-future-of-music-production/ (дата звернення: 12.05.2025).

17. HISTORY OF AUDIO, analog to digital production | timeline. *TravSonic Audio Production*. URL: https://www.travsonic.com/history-of-audio-recording-analog-digital/ (дата звернення: 12.05.2025).

18. Lindquist M. J. Unit two, part two: pro tools recording software and mixing a session. *Pressbooks*. URL: https://mlpp.pressbooks.pub/audioproduction/chapter/unit-two-pro-tools-recording-software/ (дата звернення: 12.05.2025)..

19. Online learning methods for effective communication between teachers and students / V. Grebenyuk та ін. *Measurement methodologies to assess the effectiveness of global online learning*. 2022. С. 289–309. URL: https://doi.org/10.4018/978-1-7998-8661-7.ch011 (дата звернення: 12.05.2025).

20. Sterne J. Plug-in | electricity, power, efficiency | britannica. *Encyclopedia britannica*. 2014. URL: https://www.britannica.com/technology/plug-in (дата звернення: 12.05.2025).

21. Yun Y., Cha S.-H. Designing virtual instruments for computer music. *International journal of multimedia and ubiquitous engineering*. 2013. Т. 8, № 5. С. 173–178. URL: https://doi.org/10.14257/ijmue.2013.8.5.16 (дата звернення: 12.05.2025).

22. VST 2 чи VST 3 – в чому різниця і що краще? | AREFYEV Studio. *Професійне зведення та мастеринг | AREFYEV Studio*. URL: https://arefyevstudio.com/uk/2019/01/26/vst-2-abo-vst-3-v-chomu-riznicya-i-shho-krashhe/ (дата звернення: 12.05.2025).

23. The role of sound design in immersive gaming experiences | pingle studio. *Pingle Studio*. URL: https://pinglestudio.com/blog/the-role-of-sound-design-in-immersive-gaming-experiences (дата звернення: 12.05.2025).

24. Benchmark of the best AI for music analysis in 2025. *Bridge.audio*. URL: https://www.bridge.audio/blog/benchmark-of-the-best-ai-for-music-analysis-in-2025/ (дата звернення: 12.05.2025).

25. Audio analysis with machine learning - unlocking sound insights | tensorway. *Tensorway - AI Development Company | We Reinvent Companies with Artificial Intelligence*. URL: https://www.tensorway.com/post/audio-analysis-with-machine-learning (дата звернення: 12.05.2025).

26. Judge allows 'New York Times' copyright case against OpenAI to go forward. *NPR*. URL: https://www.npr.org/2025/03/26/nx-s1-5288157/new-york-times-openai-copyright-case-goes-forward (дата звернення: 12.05.2025).

27. Generative audio workstations: AI vsts & the future of daws. *AudioCipher*. URL: https://www.audiocipher.com/post/generative-audio-workstation (дата звернення: 12.05.2025).

28. FL studio 25 now includes gopher AI text support in the DAW. *AudioCipher*. URL: https://www.audiocipher.com/post/fl-studio-gopher (дата звернення: 12.05.2025).

29. WavTool: the world's first text-to-music AI DAW uses GPT-4. *AudioCipher*. URL: https://www.audiocipher.com/post/ai-daw (дата звернення: 12.05.2025).

30. The best AI music production tools: a complete & expert guide. *Tracklib.com*. URL: https://www.tracklib.com/blog/ai-music-production-tools (дата звернення: 12.05.2025).

31. NSynth super. *NSynth Super*. URL: https://nsynthsuper.withgoogle.com/ (дата звернення: 12.05.2025).

32. Understanding noise suppression: a comprehensive guide to audio quality enhancement. *Tencent RTC*. URL: https://trtc.io/blog/details/noise-suppression (дата звернення: 12.05.2025).

33. Industry trends: machine learning for commercial audio production. *IEEE Signal Processing Society*. URL: https://signalprocessingsociety.org/newsletter/2019/11/industry-trends-machine-learning-commercial-audio-production (дата звернення: 12.05.2025).

34. Borg J. Impact of AI on music production industry | pibox resources. *Pibox*. URL: https://pibox.com/resources/ai-in-music-production/ (дата звернення: 12.05.2025).

35. Top 16 AI plugins and tools for mixing and mastering music. *Production Music Live*. URL: https://www.productionmusiclive.com/blogs/news/top-7-ai-plugins-and-tools-for-mixing-and-mastering-music-in-2023 (дата звернення: 12.05.2025).

36. IZotope neutron 4: a guide to mixing with the plugin - blog | splice. *Blog | Splice*. URL: https://splice.com/blog/how-to-mix-izotope-neutron/ (дата звернення: 13.05.2025).

37. Behind the technology of mix assistant. *Access Denied*. URL: https://www.izotope.com/en/learn/behind-the-technology-of-mix-assistant (дата звернення: 13.05.2025).

38. Jackson H. Sonible Smart EQ 3 Review – Does AI Actually Work? *Whipped Cream Sounds*. URL: https://www.whippedcreamsounds.com/smart-eq-3-review/ (дата звернення: 13.05.2025).

39. IPlug 2: documentation. *iPlug2 - C++ audio plug-in framework | iPlug2.github.io*. URL: https://iplug2.github.io/docs/ (дата звернення: 14.05.2025).

40. Download FX projects – will pirkle audio technology. *Will Pirkle Audio Technology*. URL: https://www.willpirkle.com/fx-book/project-gallery/ (дата звернення: 14.05.2025).

41. JUCE 8 adds important new features and enhancements for audio application and plugin development. *audioXpress*. URL: https://audioxpress.com/index.php/news/juce-8-adds-important-new-features-and-enhancements-for-audio-application-and-plugin-development (дата звернення: 14.05.2025).

42. Jamba. *Jamba*. URL: https://jamba.dev/ (дата звернення: 14.05.2025).

43. Faust documentation. *Faust Documentation*. URL: https://faustdoc.grame.fr/ (дата звернення: 14.05.2025).

44. A visual programming environment for audio experimentation, prototyping and education · plugdata. *A visual programming environment for audio experimentation, prototyping and education · plugdata*. URL: https://plugdata.org/ (дата звернення: 14.05.2025).

45. NotaGen: advancing musicality in symbolic music generation with large language model training paradigms. *arXiv.org*. URL: https://arxiv.org/abs/2502.18008 (дата звернення: 16.05.2025).

46. Tamulionis M., Sledevič T., Serackis A. Investigation of machine learning model flexibility for automatic application of reverberation effect on audio signal. *Applied sciences*. 2023. Т. 13, № 9. С. 5604. URL: https://doi.org/10.3390/app13095604 (дата звернення: 17.05.2025).

47. Music mixing style transfer: a contrastive learning approach to disentangle audio effects / J. Koo та ін. *ICASSP 2023 - 2023 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, м. Rhodes Island, Greece, 4–10 черв. 2023 р. 2023. URL: https://doi.org/10.1109/icassp49357.2023.10096458 (дата звернення: 17.05.2025).

48. Spectrogram reading - what are spectrograms?. *Wayback Machine*. URL: https://web.archive.org/web/19991008000849/http:/cslu.cse.ogi.edu/tutordemos/SpectrogramReading/spectrogram.html (дата звернення: 17.05.2025).

49. Gallardo-Antolín A., Montero J. M. On combining acoustic and modulation spectrograms in an attention LSTM-based system for speech intelligibility level classification. *Neurocomputing*. 2021. Т. 456. С. 49–60. URL: https://doi.org/10.1016/j.neucom.2021.05.065 (дата звернення: 18.05.2025).

50. Toward an automatic quality assessment of voice-based telemedicine consultations: a deep learning approach / M. Habib та ін. *Sensors*. 2021. Т. 21,

№ 9. С. 3279. URL: https://doi.org/10.3390/s21093279 (дата звернення: 19.05.2025).

51. SARSA reinforcement learning algorithm: a guide | built in. *Built In*. URL: https://builtin.com/machine-learning/sarsa (дата звернення: 21.05.2025).

52. Tuning recurrent neural networks with reinforcement learning. *Magenta*. URL: https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning (дата звернення: 22.05.2025).

53. RL-Duet: online music accompaniment generation using deep reinforcement learning / N. Jiang та ін. *Proceedings of the AAAI conference on artificial intelligence*. 2020. Т. 34, № 01. С. 710–718. URL: https://doi.org/10.1609/aaai.v34i01.5413 (дата звернення: 23.05.2025).

54. Spectral_features. *musicinformationretrieval.com | Instructional notebooks on music information retrieval*. URL: https://musicinformationretrieval.com/spectral_features.html (дата звернення: 26.05.2025).

55. What is XGBoost algorithm?. *HowDev*. URL: https://how.dev/answers/what-is-xgboost-algorithm (дата звернення: 27.05.2025).

56. Chauhan N. S. LightGBM demystified: understanding the math behind the algorithm. *Intelligent Machines*. URL: https://www.intelligentmachines.blog/post/lightgbm-demystified-understanding-the-math-behind-the-algorithm (дата звернення: 27.05.2025).

57. Attention Is All You Need / A. Vaswani та ін. *31st Conference on Neural Information Processing Systems*. 2017. URL: https://arxiv.org/abs/1706.03762v7 (дата звернення: 28.05.2025).

58. Andrada. GTZAN dataset – music genre classification. *Kaggle*. URL: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification (дата звернення: 29.05.2025).

59. What is the difference between reactive and proactive AI agents? - Zilliz Vector Database. *Zilliz: Vector Database built for enterprise-grade AI*

applications. URL: https://zilliz.com/ai-faq/what-is-the-difference-between-reactive-and-proactive-ai-agents (дата звернення: 29.05.2025).

60. Newsletter B. A. Weight initialization in neural networks: xavier & he strategies. *Business Analytics Review | Business Analytics Newsletter | Substack*. URL: https://businessanalytics.substack.com/p/weight-initialization-in-neural-networks (дата звернення: 30.05.2025).

61. Double deep q-learning: an introduction | built in. *Built In*. URL: https://builtin.com/artificial-intelligence/double-deep-q-learning (дата звернення: 31.05.2025).

62. What are target networks in DQN?. *Milvus | High-Performance Vector Database Built for Scale*. URL: https://milvus.io/ai-quick-reference/what-are-target-networks-in-dqn (дата звернення: 31.05.2025).

63. Actor prioritized experience replay / B. Saglam та ін. *Journal of artificial intelligence research*. 2023. Т. 78. С. 639–672. URL: https://doi.org/10.1613/jair.1.14819 (дата звернення: 01.06.2025).

64. What is the role of exploration noise in reinforcement learning?. *Milvus | High-Performance Vector Database Built for Scale*. URL: https://milvus.io/ai-quick-reference/what-is-the-role-of-exploration-noise-in-reinforcement-learning (дата звернення: 01.06.2025).

65. GeeksforGeeks. Why do we have to normalize the input for an artificial neural network? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/why-do-we-have-to-normalize-the-input-for-an-artificial-neural-network (дата звернення: 01.06.2025).

66. Can PyTorch neural network model have the same code for the CPU and GPU processing? - EITCA Academy. *EITCA Academy*. URL: https://eitca.org/artificial-intelligence/eitc-ai-dlpp-deep-learning-with-python-and-pytorch/advancing-with-deep-learning/computation-on-the-gpu/can-pytorch-neural-network-model-have-the-same-code-for-the-cpu-and-gpu-processing/ (дата звернення: 02.06.2025).

67. Pykes K. AdamW optimizer in pyTorch tutorial. *DataCamp*. URL: https://www.datacamp.com/tutorial/adamw-optimizer-in-pytorch (дата звернення: 02.06.2025).