Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Рекомендовано до захисту:
протокол засідання кафедри
№ _____ від _____._____.2025 р.
Завідувач кафедри ІПСіТ
_____ Олешко О. І.
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему Використання нейронної мережі для автоматичної транскрипції аудіофайлів у нотний запис

Захищено на засіданні ЕК № _____
протокол № _____ від _____._____.2025 р.
Оцінка _____ / _____
Голова ЕК
_____ Фесенко Г. В.
(підпис) (прізвище та ініціали)

Виконала:
студентка 4 курсу, групи КС-42
Спеціальності:
122 Комп'ютерні науки
(шифр і назва спеціальності підготовки)
Гребенюк Єлизавета Миколаївна
(прізвище, ім'я та по батькові, підпис)
Керівник канд. тех. наук, доцент,
Олешко О. І.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент канд. фіз.-мат. наук, доцент,
Споров О. Є.
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра: інтелектуальних програмних систем і технологій
Рівень вищої освіти (освітньо-кваліфікаційний рівень): бакалавр
Спеціальність: 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри ІПСіТ

_____ Олег ОЛЕШКО
підпис ім'я, прізвище

"_____" _____ 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гребенюк Єлизавета Миколаївна
(прізвище, ім'я, по батькові студента)

1. Тема роботи: «Використання нейронної мережі для автоматичної транскрипції аудіофайлів у нотний запис»

керівник роботи Олешко Олег Іванович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "16" квітня 2025 року № 4101-5/962

2. Строк подання студентом роботи 2 червня 2025 року

3. Перелік питань, які потрібно розробити:

ознайомитись з особливостями цифрового представлення музичних аудіофайлів та нотного запису, провести огляд існуючих методів автоматичної транскрипції музики, познайомитися з алгоритмом роботи та методиками навчання нейронних мереж для задач обробки аудіосигналів, створити концептуальну модель системи автоматичної транскрипції з використанням нейронної мережі, провести деталізоване проєктування модуля розпізнавання нот із аудіофайлів, здійснити експериментальне дослідження точності та ефективності обраного підходу, сформулювати висновки щодо перспектив подальшого розвитку дослідження.

4. План роботи

№ з/п	Назви етапів роботи
1	Формулювання мети та повного переліку завдань для виконання кваліфікаційної роботи (КР).
2	Пошук та аналіз інформаційних джерел за темою КР.
3	Визначення особливостей аудіосигналів музичних творів і принципів їх нотного представлення.
4	Розробка концептуальної моделі нейронної мережі для задачі транскрипції.
5	Детальне проєктування системи автоматичної транскрипції аудіофайлів.
6	Проведення експериментального дослідження ефективності моделі.
7	Підсумок отриманих результатів та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.

5. Дата видачі завдання 10 жовтня 2025 року

Студент _____ Є. М. Гребенюк
(підпис) (ініціали, прізвище)

Керівник роботи _____ О. І. Олешко
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота містить 73 сторінки тексту, 3 розділи, 32 рисунки, 4 таблиці, 7 формул, 2 додатки, оформлених на 3 сторінках. До роботи включено перелік з 27 інформаційних джерел.

У роботі розглянуто проблему автоматичної транскрипції музичних аудіофайлів у нотний запис із використанням нейронних мереж. Основну увагу приділено аналізу цифрового представлення аудіосигналів, огляду сучасних методів транскрипції та побудові концептуальної моделі відповідної системи.

Метою дослідження є розробка концептуальної моделі та створення прототипу інформаційної системи, що виконує автоматичну транскрипцію музичних аудіофайлів у нотний запис з використанням методів глибокого навчання.

Методи дослідження ґрунтуються на машинному навчанні та цифровій обробці сигналів. Аудіо трансформується в лог-мел спектрограми, які подаються на вхід CNN; після навчання модель декодує дані у MIDI-формат і генерує нотний запис.

У роботі використано такі програмні та апаратні засоби: мову програмування Python, бібліотеки *librosa*, *pretty_midi, music21*, а також датасет MAESTRO як джерело аудіо- та нотних даних для навчання та тестування моделі.

У результаті дослідження створено прототип AMT-системи на базі CNN для трансформації фортепіанної аудіо у MIDI й нотний запис. Реалізовано обробку спектрограм, генерацію нот та візуалізацію. Новизна полягає в адаптації CNN до AMT із балансом точності й ефективності для практичного застосування.

Ключові слова: АУДІОСИГНАЛ, ТРАНСКРИПЦІЯ, НЕЙРОННА МЕРЕЖА, НОТНИЙ ЗАПИС, ГЛИБОКЕ НАВЧАННЯ, ЦИФРОВА ОБРОБКА СИГНАЛІВ, СПЕКТОГРАМА.

ABSTRACT

The qualification work consists of 73 pages of text, 3 chapters, 32 figures, 4 tables, and 7 formulas, 2 appendices on 3 pages. The list of references includes 27 sources.

The explanatory note addresses the problem of automatic transcription of musical audio files into sheet music using neural networks. Special attention is given to the analysis of digital representations of audio signals, the review of modern transcription methods, and the development of a conceptual model for the transcription system.

The objective of this study is to develop a conceptual model and implement a prototype of an information system that performs automatic transcription of musical audio files into sheet music using deep learning techniques.

Research methods are based on machine learning and digital signal processing. Audio data is converted into log-Mel spectrograms, which serve as input to a convolutional neural network (CNN); after training, the model decodes the output into MIDI format and generates corresponding sheet music notation.

Software and hardware tools used in the project include the Python programming language and the *librosa*, *pretty_midi*, and *music21* libraries. The MAESTRO dataset was used as a source of audio and symbolic music data for model training and evaluation.

As a result of the study, a CNN-based AMT system prototype was developed to convert piano audio into MIDI and sheet music. The system performs spectrogram processing, note generation, and visualization. The novelty lies in adapting a convolutional neural network to the AMT task while maintaining a balance between transcription accuracy and computational efficiency, making the solution suitable for educational and applied use.

Keywords: AUDIO SIGNAL, TRANSCRIPTION, NEURAL NETWORK, SHEET MUSIC, DEEP LEARNING, DIGITAL SIGNAL PROCESSING, SPECTROGRAM.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АМТ – автоматична музична транскрипція.

АЦП – аналого-цифрове перетворення.

МН – машинне навчання.

НМ – нейронна мережа.

ПЗ – програмне забезпечення.

ЦТ – цифрові технології.

ШІ – штучний інтелект.

ASR (Automatic Speech Recognition) – автоматичне розпізнавання мовлення.

BCE (Binary Cross-Entropy) – функція втрат бінарної крос-ентропії.

CNN (Convolutional Neural Network) – згорткова нейронна мережа.

DAW (Digital Audio Workstation) – цифрова звукова робоча станція.

DFT (Discrete Fourier Transform) – дискретне перетворення Фур’є.

DL (Deep Learning) – глибоке навчання.

DNN (Deep Neural Networks) – глибока нейронна мережа.

DSP (Digital Signal Processing) – цифрова обробка сигналів.

FFT (Fast Fourier transform) – швидке перетворення Фур’є.

GRU (Gated Recurrent Unit) – вентильні рекурентні вузли.

HMM (Hidden Markov Models) – приховані марковські моделі.

LSTM (Long Short-Term Memory) – нейронна мережа довгої короткочасної пам’яті.

MFCC (Mel-Frequency Cepstral Coefficients) – мел-частотні кепстральні коефіцієнти.

MIDI (Musical Instrument Digital Interface) – цифровий інтерфейс музичних інструментів.

MP3 (MPEG-1/2 Audio Layer III) – формат аудіофайлу зі стисненням.

NMF (Non-negative Matrix Factorization) – невід’ємне матричне розкладання.

PCM (Pulse Code Modulation) – імпульсно-кодова модуляція.

RNN (Recurrent Neural Network) – рекурентна нейронна мережа.

STFT (Short-Time Fourier Transform) – короткочасне перетворення Фур’є.

WAV (Waveform Audio File Format) – формат аудіофайлу з хвильовою формою.

A – ля.

B – сі-бемоль.

C – до.

D – ре.

E – мі.

F – фа.

G – соль.

H – сі.

Гц – герц, одиниця частоти.

дБ – децибел, одиниця рівня гучності.

ВСТУП

У сучасному світі розвитку цифрових технологій (ЦТ) та штучного інтелекту (ШІ) автоматизація творчих процесів, зокрема у сфері музики, набуває дедалі більшого значення. Одним із актуальних напрямів сучасної музичної інформатики є задача автоматичної транскрипції аудіосигналів у нотний запис – процесу, за якого музичний твір, представлений у вигляді звукових коливань, перетворюється на формалізовану нотацію. Даний процес і досі залишається науковим викликом, оскільки поєднує методи цифрової обробки сигналів, комп'ютерного зору, ШІ та музичної теорії.

Музична транскрипція – це процес перетворення аудіозапису музичного твору в символічне представлення, таке як нотний запис або MIDI-файл. Він є необхідним для збереження, аналізу та відтворення музичних композицій. Традиційно транскрипція виконується вручну та вимагає значних зусиль і часу. Розвиток ЦТ відкрив можливість автоматизації згаданого процесу, що стало предметом досліджень у галузі автоматичної музичної транскрипції (AMT).

Сучасні методи AMT базуються на використанні алгоритмів машинного навчання (МН) та глибоких нейронних мереж (НМ) (DNN). Згідно з дослідженнями, найефективнішими архітектурами у задачах розпізнавання нот з аудіосигналів є згорткові нейронні мережі (CNN) та трансформери. Наприклад, у роботі Zhang, Q., Chen, X., Liu, Y. та ін. (2022) було представлено гібридну модель CNN-Transformer для транскрипції фортепіанної музики. У ході експериментів модель досягла F1-метрики 0.8551 на датасеті MAPS та 0.9042 на датасеті MAESTRO. Автори також зазначили, що поєднання згорткових шарів для витягнення спектральних ознак з механізмом уваги (attention mechanism) трансформера покращило якість транскрипції в порівнянні з класичними підходами на базі лише CNN або RNN [1].

Іншим прикладом підходу до вирішення задачі AMT є дослідження Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C. Z. A., Dieleman, S.,

та ін. (2018), у якому було реалізовано модель Onsets and Frames. У моделі були поєднані згорткові та рекурентні НМ та акцентовано увагу на одночасному виявленні початку нот (onsets) і їхньої тривалості (frames). Було досягнуто значного підвищення F1-метрики на датасеті MAPS. Отримані результати також підтверджують доцільність застосування комбінованих архітектур глибокого навчання (DL) у підвищенні точності AMT-систем [2].

AMT має широке практичне застосування в різних галузях музичної діяльності. Вона використовується у розробці програмного забезпечення (ПЗ) для композиторів та аранжувальників. Наприклад, у цифрових робочих станціях (DAW), які підтримують автоматичне генерування нот на основі імпортованого аудіо. У сфері музичної освіти системи AMT допомагають студентам аналізувати виконання та порівнювати його з нотним записом у реальному часі. Також транскрипція відіграє важливу роль у цифровому архівуванні старих або ненотованих музичних творів, оскільки полегшує їхню каталогізацію та подальший аналіз. У застосунках, таких як ScoreCloud або AnthemScore, функція миттєвого перетворення звуку у нотний запис дає змогу швидко зафіксувати музичну ідею без необхідності ручного нотування [3, 4].

Незважаючи на значний прогрес AMT-систем, існують виклики, пов’язані з транскрипцією поліфонічної музики, варіаціями темпу та динаміки, а також шумами в аудіозаписах. Дослідники продовжують працювати над удосконаленням моделей, в тому числі шляхом комбінування акустичних моделей з мовними моделями музики. Завдяки такому методу враховується музичний контекст і покращується точність транскрипції [5].

Об’єкт дослідження – процес автоматичної транскрипції музичного аудіосигналу у нотний запис.

Предмет дослідження – методи та алгоритми DL, зокрема НМ, які застосовуються для автоматичної транскрипції музики.

Мета дослідження – розробка концептуальної моделі та прототипу інформаційної системи, що виконує автоматичну транскрипцію музичного аудіофайлу у нотний запис на основі DL.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі методи і програмні рішення у сфері музичної транскрипції;

– вивчити особливості цифрового представлення музичних сигналів та нотного запису;

– дослідити принципи побудови та навчання НМ для задач обробки аудіо;

– спроєктувати концептуальну модель системи автоматичної транскрипції;

– реалізувати прототип модуля транскрипції та перевірити його працездатність;

– провести експериментальне дослідження ефективності моделі;

– сформулювати висновки щодо перспектив подальшого розвитку дослідження.

Актуальність дослідження зумовлена стрімким зростанням обсягів цифрової музичної інформації та необхідністю її ефективної обробки, аналізу і структурування. Музична індустрія все більше покладається на автоматизовані інструменти для створення, редагування та аналізу аудіоконтенту. У зв’язку з чим виникає потреба в системах, які можуть автономно інтерпретувати аудіосигнал. Особливо важливо, щоб дані системи перетворювали звук у формалізоване представлення, такий як нотний запис.

Новизна роботи полягає в інтеграції сучасних методів DL із завданням автоматичного нотування і передбачає застосування навченої нейронної моделі для розпізнавання нот з аудіофайлів без участі людини. Отримані результати можуть бути використані в подальшому як основа для створення прикладного ПЗ в сфері музичної індустрії та освіти.

1 АНАЛІЗ СТАНУ РОЗРОБОК У ГАЛУЗІ АВТОМАТИЧНОЇ МУЗИЧНОЇ ТРАНСКРИПЦІЇ

1.1 Автоматична транскрипція музики як науково-технічна проблема

1.1.1 Поняття та значення автоматичної музичної транскрипції

Автоматична музична транскрипція (АМТ) – це процес перетворення аудіозапису музики в музичну нотацію (наприклад, нотний запис або MIDI-файл) без участі людини. Інакше кажучи, метою АМТ є автоматичне розпізнавання музики – аналіз акустичного музичного сигналу і отримання з нього символічного представлення (партитури).

Необхідність автоматичної транскрипції зумовлена практичними потребами музикантів та дослідників. Ручне розшифрування «на слух» розвиває навички, але потребує високої підготовки. До того ж значна частина записів, особливо народних або сучасних, не мають нотного оформлення. У таких випадках автоматичні інструменти суттєво полегшать отримання нот, зекономивши час і зусилля.

Перші програми АМТ з’явилися наприкінці XX століття й стали основою подальших досліджень. Розвиток розпізнавання мовлення (ASR) також сприяв прогресу в розпізнаванні музики завдяки подібності сигналів. У 2000-х поширились статистичні й машинні методи, а з 2010-х – глибоке навчання, яке значно підвищило точність поліфонічної транскрипції. Проте універсальної АМТ-системи, здатної стабільно створювати точну партитуру для будь-якої музики, досі не існує. Завдання залишається відкритим, особливо в умовах багатоголосся та змішаних інструментів.

Розшифрувати мелодію одноголосого інструмента з аудіо не складно для досвідченого музиканта, однак створити повну партитуру оркестрового твору «на слух» значно важче. Аналогічно, сучасні АМТ-системи добре працюють із монодією, але транскрипція поліфонічної музики з багатьма голосами чи інструментами досі є складною й відкритою задачею [6].

1.1.2 Підзадачі автоматичної музичної транскрипції

Процес автоматичної транскрипції музики складається з кількох послідовних або паралельних підзадач [6]:

1) виявлення висот (multi-pitch detection) – визначення всіх нот, що звучать одночасно в кожен момент часу;

2) виявлення моментів атаки/затухання (onset/offset detection) – точне виявлення початку та завершення кожної ноти;

3) оцінка гучності та тривалості – вимірювання амплітуди та ритмічна квантизація нот;

4) розпізнавання інструментів – ідентифікація джерела звучання за тембром;

5) виділення ритмічної структури – визначення темпу, розміру та ритмічного малюнку;

6) перетворення на нотний запис – створення нотної нотації (наприклад, MIDI) з усіма параметрами.

На рисунку 1.1 наведено узагальнену схему процесу перетворення аудіосигналу у нотне представлення. Вона включає три етапи: вгорі – часовий сигнал аудіо (хвильова форма), посередині – спектрограма з виявленими висотами звуків (позначено кольором), внизу – відновлений нотний запис твору. Як приклад використано фрагмент прелюдії Й. С. Баха. Після розпізнавання частот, результати перетворюються на нотний запис.

Складність задачі AMT зумовлена необхідністю моделювати слуховий аналіз, подібний до того, який виконує музикант під час транскрипції «на слух». Особливо важкою є обробка поліфонії: кілька одночасних нот створюють спільний спектр із накладанням гармонік (наприклад, до-мі-соль має ≈46% спільних обертонів), що ускладнює розділення окремих звуків. Таке явище називають недовизначеним розділенням джерел [7].

Незважаючи на труднощі, AMT-системи активно розвиваються завдяки великому прикладному значенню в музиці, освіті та дослідженнях.

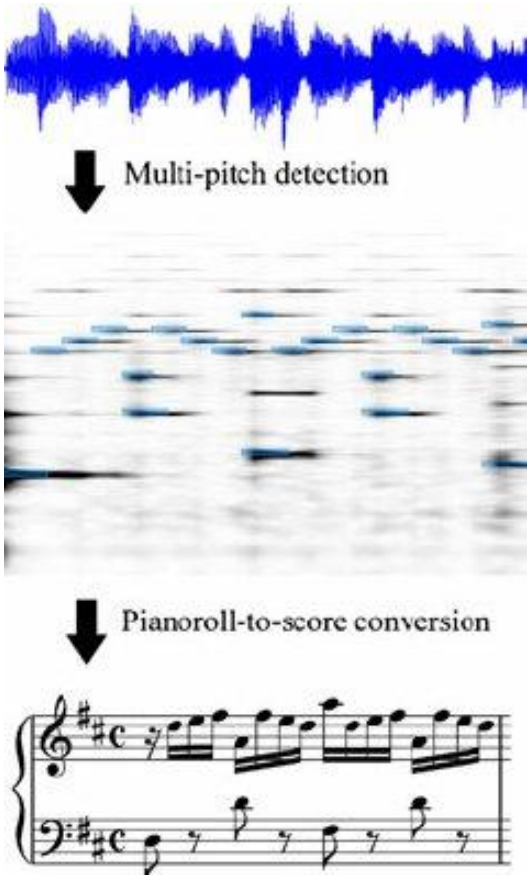

Рисунок 1.1 – Схема процесу автоматичної транскрипції музичного сигналу [6].

1.2 Теоретичні основи цифрової обробки музичних сигналів

1.2.1 Аналогово-цифрове перетворення аудіосигналів

Цифрова обробка аудіосигналів спирається на математичні методи й алгоритми, які стосуються аналізу звуку у цифровій формі. Аудіо являє собою коливання повітряного тиску (акустичну хвилю), яке в аналоговому вигляді є неперервним сигналом.

Перетворення аналогового сигналу на цифровий здійснюється шляхом аналого-цифрового перетворення (АЦП), яке складається з двох основних етапів:

– дискретизації – вибірки значень сигналу через рівні проміжки часу;

– квантування – округлення амплітудних значень до найближчих рівнів із фіксованою точністю.

Дискретизація виконується з певною частотою (наприклад, 44,1 кГц),

тобто кількістю відліків за секунду, яка називається частотою дискретизації. Згідно з теоремою Котельникова (Найквіста-Шеннона), частота має бути вдвічі вищою за максимальну частоту сигналу [8].

Квантування визначає точність відображення амплітуди: чим більше біт використовується на один відлік, тим точніше відтворюється рівень сигналу. Стандартне 16-бітне кодування дає 65536 рівнів і динамічний діапазон до 96 дБ; професійні системи використовують 24 біти для досягнення до 144 дБ.

В результаті оцифровування утворюється цифровий сигнал – послідовність чисел, які відповідають відлікам початкової аналогової хвилі. Сигнал зберігається у нестиснених форматах (наприклад, WAV) або з використанням алгоритмів стиснення.

На рисунку 1.2 зображено, як аналогова хвиля перетворюється на цифровий сигнал шляхом вибірки значень з подальшим квантуванням амплітуди. Синусоїдальний аналоговий сигнал (синя крива) спочатку дискретизується з фіксованим часовим кроком – на кожному інтервалі формується відлік (семпл), показаний червоними стовпчиками. Далі кожен із відліків округлюється до найближчого рівня амплітуди відповідно до розрядності – це відповідає етапу квантування сигналу.

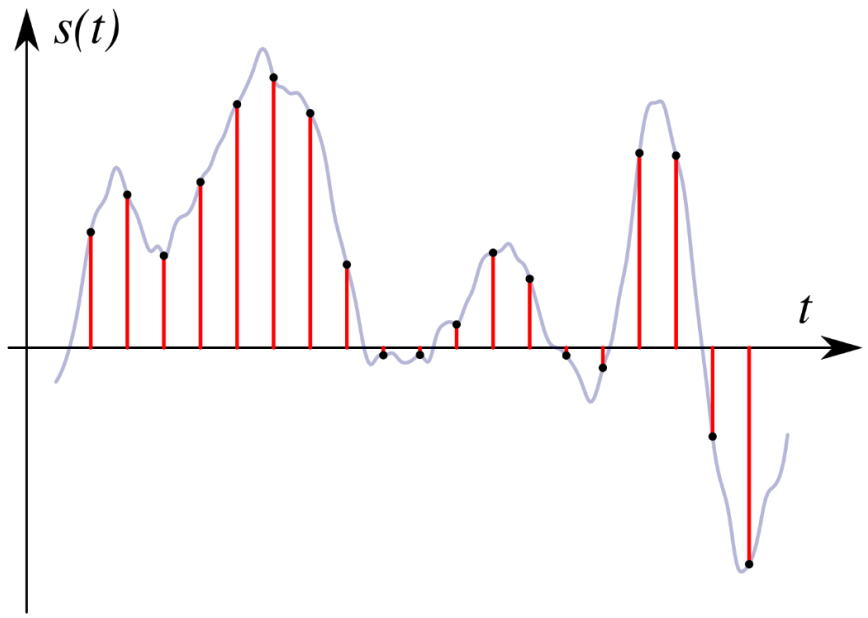

Рисунок 1.2 – Аналого-цифрове перетворення сигналу [9].

Для цифрової обробки аудіо сигнал подається у вигляді дискретної послідовності $x[n]$ – числових значень амплітуди у фіксовані моменти часу. Сигнал розглядається як функція часу, визначена на дискретній часовій сітці з кроком

$$T = \frac{1}{F_S}, \tag{1.1}$$

де F_S – частота дискретизації, тобто кількість відліків (семплів) за секунду.

У цифровому домені частота та амплітуда зберігають фізичну інтерпретацію. Частота (Гц) визначає музичну висоту: чим вона вища, тим вищим сприймається звук (наприклад, нота A4 = 440 Гц). Амплітуда – це максимальне відхилення сигналу, що впливає на гучність: більша амплітуда – гучніший звук.

Варто зазначити, що гучність є суб'єктивною характеристикою і сприймається людиною нелінійно. Вона залежить не лише від амплітуди, а й від частоти (людське вухо менш чутливе до дуже низьких і дуже високих частот) та від тривалості звучання.

Для об'єктивного вимірювання гучності використовується рівень звуку, який подається у децибелах (дБ) та визначається логарифмічною залежністю від амплітуди сигналу:

$$L = 20\log_{10}\left(\frac{A}{A_0}\right), \tag{1.2}$$

де A – амплітуда сигналу;

A_0 – еталонне значення амплітуди.

1.2.2 Перетворення сигналів у частотну область

Одним із ключових інструментів цифрової обробки сигналів є перетворення Фур'є – математична операція, яка подає сигнал як

суперпозицію синусоїдальних коливань. Розклад Фур'є для неперервного сигналу дає змогу визначити його спектр – сукупність комплексних амплітуд гармонічних складових на різних частотах.

Для дискретних сигналів застосовується дискретне перетворення Фур'є (DFT), яке обчислює спектральні компоненти на обмеженому наборі дискретних частот. Ефективний розрахунок DFT забезпечується за допомогою швидкого перетворення Фур'є (FFT), яке є стандартним інструментом у спектральному аналізі звукових сигналів.

У результаті застосування FFT до цифрового сигналу отримують набір комплексних чисел $X[k]$, кожне з яких описує амплітуду та фазу гармонічної складової сигналу на частоті

$$f_k = \frac{kF_S}{N}, \tag{1.3}$$

де F_S – частота дискретизації;

N – довжина вибірки (кількість відліків);

k – номер спектральної складової.

Модуль спектру $|X[k]|$ показує, як енергія сигналу розподілена по частотах: для періодичних звуків – піки на гармоніках, для складних – широкий спектр. Оскільки музичні сигнали змінюються з часом, стандартне перетворення Фур'є недостатнє, тому використовують його модифікації.

Найпоширенішим підходом є короткочасне перетворення Фур'є (STFT), при якому сигнал розбивається на короткі фрагменти (фрейми) тривалістю, як правило, 20-50 мс, і до кожного з них застосовується FFT. Такий підхід дає змогу відстежувати динаміку спектра в часі [10].

STFT візуалізується як спектрограма – графік із часом по горизонталі, частотою по вертикалі й амплітудою у вигляді кольору або яскравості. Гармоніки формують горизонтальні смуги, ритмічні удари – короткі сплески. Така візуалізація широко використовується в задачах автоматичної

транскрипції для виявлення нот як частотно-часових візерунків [11].

Сучасні AMT-системи не працюють безпосередньо з аудіосигналом, а використовують спектрограму як зображення для виявлення нот. Додатково обчислюють кепстральні ознаки, зокрема, мел-частотні кепстральні коефіцієнти (MFCC), які стисло описують форму спектра. MFCC використовуються як вхідні ознаки для класифікації нот, знижуючи обчислювальне навантаження на моделі.

1.3 Особливості аудіосигналів музичних творів та принципи нотного представлення

1.3.1 Структура аудіосигналу музичного твору

Аудіосигнал може бути монофонічним (один канал) або стерео (два канали – лівий і правий). Стерео надає просторову інформацію, але збільшує обсяг обробки. У задачах AMT зазвичай використовують моно або стереозаписи, рідше – багатоканальні формати.

Ключовими параметрами аудіо є частота дискретизації та розрядність, про що йшлося в розділі 1.2. CD-якість (44,1 кГц / 16 біт) підходить для більшості задач AMT, тоді як професійні формати (48-96 кГц, 24 біти) забезпечують кращу точність і динаміку [12].

Для аналізу музики спектрограма є інформативнішою за осцилограму, оскільки розкладає звук на частоти та дозволяє побачити структуру нот: кожна нота утворює набір гармонік – горизонтальних смуг на зображенні.

Гармоніки – це кратні до основної частоти складові звуку; фундаментальний тон – перша гармоніка, а обертони формують тембр. Різні інструменти звучать по-різному на одній ноті через відмінності в амплітуді гармонік і формі атаки звуку.

Окрім спектра, тембр залежить і від часових характеристик, зокрема форми атаки звуку, тобто як швидко сигнал досягає максимальної амплітуди.

1.3.2 Атака і сустейн

Звуки інструментів не є повністю стаціонарними навіть під час сталої ноти. Особливою є фаза атаки – початковий перехід від тиші до звучання, що містить шумові компоненти й значно впливає на сприйняття тембру. Саме атака дає змогу розпізнати інструмент у перші мілісекунди. Далі настає сустейн (sustain) – фаза сталого звучання, і реліз (release) – згасання. У AMT дані фази відповідають подіям onset і offset. Різна динаміка атаки (різка у фортепіано, плавна у скрипки) ускладнює розпізнавання нот [13].

Таким чином, аудіосигнал складається з поєднання різнорідних компонентів: періодичних тонів з гармоніками, імпульсних шумів від ударних елементів, реверберацій та інших акустичних явищ.

На спектрограмі одиночного фортепіанного звуку (рисунок 1.3) видно основну гармоніку (≈98 Гц) як яскраву нижню смугу та обертони вище – паралельні лінії, що формують тембр. Зліва помітна широка зона атаки з хаотичними компонентами, далі – впорядкований сустейн, який поступово згасає під час релізу.

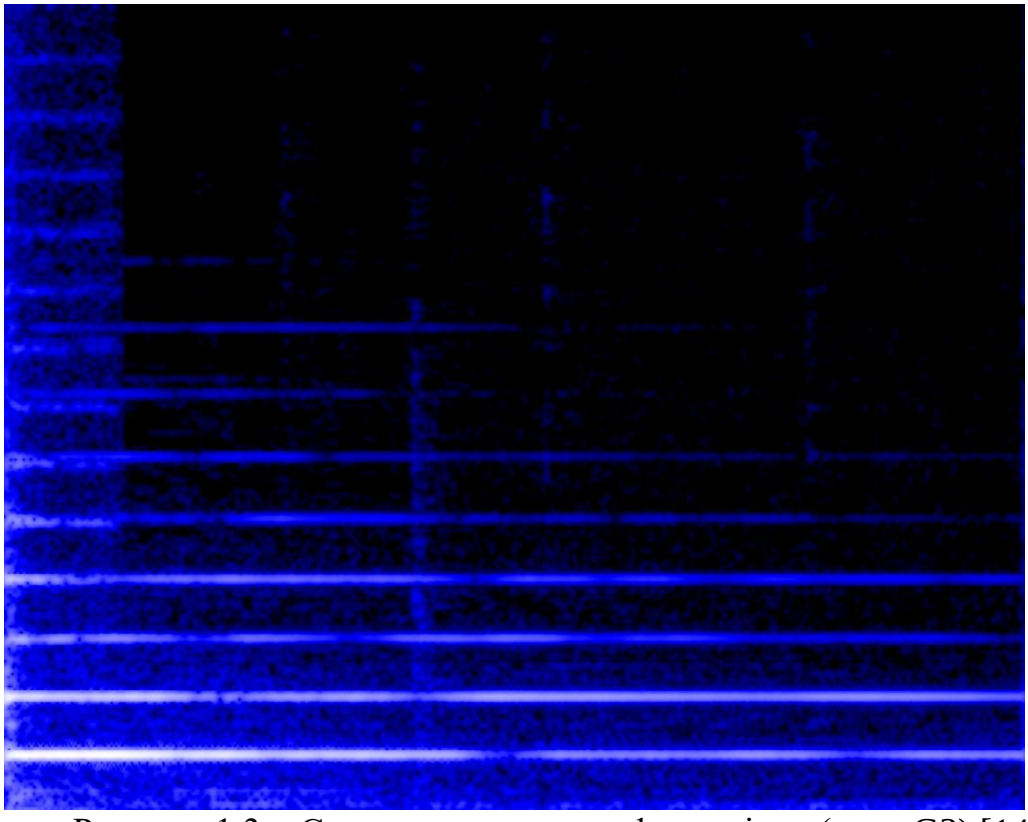

Рисунок 1.3 – Спектрограма звуку фортепіано (нота G3) [14].

1.3.3 Формати подання аудіо: PCM, WAV, MP3

Як зазначалося раніше, цифровий аудіосигнал зберігається як послідовність PCM-відліків. Найпоширенішим форматом для нестислого зберігання є WAV (Waveform Audio File Format), який містить сирі дані та заголовок з параметрами (частота, розрядність, кількість каналів). WAV-файли зберігають повну якість звуку без втрат і використовуються для обробки, аналізу та архівування.

Для поширення музики частіше використовують стиснені формати, зокрема MP3 (MPEG-1/2 Audio Layer III), який усуває малопомітні компоненти звуку завдяки психоакустичному кодуванню. Завдяки такому підходу розмір файлу зменшується у 5-10 разів порівняно з WAV.

У наукових дослідженнях перевагу надають нестислим (WAV) або безвтратним (FLAC) форматам, щоб зберегти повну точність сигналу.

Окрім PCM/WAV і MP3, існують інші формати: AAC, OGG, WMA тощо, але зрештою вони зводяться до відновлення цифрового сигналу у вигляді послідовності відліків $x[n]$. Незалежно від формату, вхідними даними для AMT є дискретний часовий сигнал, до якого застосовуються описані раніше перетворення (FFT, спектрограми тощо).

1.3.4 Звук і ноти: співвідношення між акустичним та символічним представленням

Музичний твір можна подати як аудіосигнал або у символічній формі (ноти, MIDI). Нотний запис передає висоту, ритм, темп і динаміку, але не фіксує тембр і нюанси виконання (інтонацію, флуктуації, мікродинаміку). Тембр визначається інструментом, а не записується явно, тому системи AMT мають розпізнавати його з аудіо.

Перетворення безперервного сигналу в дискретну нотну форму завжди супроводжується втратою інформації. Крім того, аудіо може містити фальшиві або зайві ноти, що ускладнює точну транскрипцію.

При оцінюванні якості транскрипції враховують три типи помилок:

– пропущені ноти (missed);
– зайві ноти (false alarms);
– неточні ноти (неправильна октава чи зміщення в часі).

Ще одна складність – переплетення партій: аудіо об’єднує всі інструменти, тоді як у нотах кожна партія подається окремо. Для розподілу нот за джерелами потрібні методи класифікації атак або source separation.

Загалом, аудіо є деталізованим, але неструктурованим, тоді як ноти – структуровані, але спрощені. Завдання AMT – подолати цю розбіжність.

1.3.5 Формати нотного кодування

Символьне представлення музики існує в різних цифрових форматах. Найпростішим є ABC-нотація – текстовий запис, у якому ноти позначаються літерами (A-G), а ритм і паузи – символами. Формат підтримує експорт у нотний запис і MIDI.

MIDI (Musical Instrument Digital Interface) формат описує музичні події: висоту, час початку й завершення, силу атаки (velocity) тощо. Такий формат зручний для AMT, оскільки безпосередньо подає параметри нот. Однак він не містить класичних нотаційних елементів (ключів, ритмічної структури), тому для виведення нотного запису потрібна додаткова обробка [15].

MusicXML – повноцінний відкритий формат для нотної нотації. Він охоплює всі деталі партитури, але через складність його рідко формують напряму. Більшість AMT-систем виводять MIDI, який потім конвертується у MusicXML спеціальними програмами [16].

1.4 Застосування нейронних мереж у задачах автоматичної музичної транскрипції

1.4.1 Алгоритмічні методи

Перші методи AMT базувалися на інженерних алгоритмах без навчання, зокрема піковій детекції – виявленні локальних максимумів у спектрі або енергії сигналу. Ноти визначались за перевищенням фіксованого порогу

амплітуди або енергозміни [17].

Для ранніх AMT-систем використовували спектрограми (на основі FFT), пошук гармонік і спектральні функції (спектральний потік, високочастотний вміст) для виявлення нот [18]. Методи не потребують навчання, добре працюють на простих сигналах, але мають обмеження на поліфонії: обертони сприймаються як ноти, слабкі сигнали втрачаються, а фіксовані пороги не адаптуються до динаміки [19].

В результаті точність таких підходів різко падає на багатоголосних записах, а налаштування параметрів потребує ручного добору під конкретні умови.

1.4.2 Статистичні методи

Покращення точності в AMT стало можливим завдяки статистичним моделям, таким як приховані марковські моделі (HMM). Вони моделювали послідовність нот як приховані стани з акустичними ознаками на вході, враховуючи ймовірні переходи між нотами. Завдяки чому згладжуються помилки та узгоджується нотова послідовність [20, 21]. Додатково HMM доповнювали музичними граматиками, щоб враховувати закономірності музичної мови.

Паралельно зі статистичними підходами для AMT активно застосовувався метод невід'ємного матричного розкладання (NMF), який розкладає спектрограму на дві матриці: спектральні шаблони нот (W) і часові активності (H). Розклад виділяє ноти шляхом аналізу ненульових коефіцієнтів у H [19, 22].

Метод ефективний для неконтрольованого навчання, не потребує розмічених даних і добре адаптується до конкретного інструменту. NMF показав гарні результати для фортепіано, але має обмеження: необхідність задання кількості джерел, схильність до локальних мінімумів, ігнорування часової структури. Також після розкладу потрібен додатковий аналіз для визначення атак і тривалостей нот. За точністю поступається сучасним

нейронним моделям.

Загалом, статистичні методи стали важливим перехідним етапом. Вони забезпечили ймовірнісні оцінки та гнучку факторизацію, але ще не досягали людського рівня точності в AMT.

1.4.3 Глибоке навчання: від багатошарових мереж до CNN, RNN, LSTM, Transformer

З початку 2010-х у сфері AMT відбувся прорив завдяки глибокому навчанню. Нейромережі почали застосовувати як для окремих підзавдань, так і в end-to-end системах, що безпосередньо перетворюють аудіо у ноти [20].

Зросла доступність обчислювальних ресурсів і великих датасетів (див. підрозділ 1.4.5), що дозволило моделям самостійно вивчати ознаки замість ручного їх налаштування.

DNN показали найвищу точність у задачах AMT. Типово вони працюють із спектрограмами та видають або матрицю активності нот (піанорол), або MIDI [19].

Важливою перевагою DL є можливість спільного розв’язання кількох підзавдань. Прикладом є модель Google Onsets and Frames (2018), яка поєднує дві мережі для виявлення атак і активності нот і забезпечує подвоєння точності транскрипції фортепіано [23].

Проте навіть найкращі моделі поки що не досягають рівня професійного музиканта, особливо в умовах складної поліфонії [19].

Типові архітектури нейронних мереж, які застосовуються в задачах транскрипції, розглянуто нижче.

1) Згорткові нейронні мережі (CNN) широко застосовуються в AMT через ефективну обробку двовимірних аудіо-представлень. Вони виявляють гармоніки (по частоті) та атаки (по часу) завдяки локальним фільтрам [19].

Переваги: інваріантність до зсувів, повторне використання фільтрів, можливість побудови глибоких моделей.

Недоліки: слабка здатність моделювати довготривалі залежності,

обмежений контекст вікна згортки.

2) Рекурентні нейронні мережі (RNN) обробляють аудіо як послідовність, передаючи внутрішній стан між кадрами для моделювання контексту та динаміки [23].

Переваги: можливість моделювати тривалість і зв'язність нот, здатність до навчання часових залежностей у мелодії та ритмі.

Недоліки: затухання градієнтів у класичних RNN, низька ефективність паралельної обробки даних.

3) Модифіковані рекурентні мережі LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) – покращені RNN з механізмами пам'яті та гейтингу, що дозволяють зберігати важливу інформацію довше.

Переваги: довготривала пам'ять про музичні події, здатність враховувати контекст як у прямому, так і зворотному напрямку.

Недоліки: висока обчислювальна складність, потреба у великому навчальному наборі.

4) Трансформери (Transformer) використовують механізм самоуваги (self-attention) для аналізу всієї послідовності без рекурентності.

Переваги: здатність моделювати глобальні музичні структури (гармонія, форма, фраза), повна паралелізація обробки.

Недоліки: висока чутливість до обсягу й якості навчальних даних, значне споживання пам'яті при обробці довгих аудіо-секвенцій.

DNN досягають найвищої точності в розпізнаванні нот, особливо для фортепіано – до 85-90% F-міри на простих даних [23, 24]. Проте ціна за якість – великі обчислювальні ресурси й обсяг даних.

Сучасні моделі мають мільйони параметрів, потребують GPU та довгого навчання. Водночас оптимізовані варіанти, як-от Basic Pitch (до 17 тис. параметрів), демонструють гарну швидкодію без значної втрати точності [25].

Проблемою лишається узагальнення: більшість моделей заточені під фортепіано, тоді як точність на інших інструментах знижується. Традиційні методи універсальніші, але поступаються точністю. Перспективним напрямом

є гібридні рішення, які поєднують обидва підходи [19].

1.4.4 Сучасні моделі автоматичної музичної транскрипції на основі нейронних мереж

Для ілюстрації прогресу в застосуванні нейронних мереж нижче розглянуто декілька відомих моделей та проєктів.

MuseNet (OpenAI, 2019) – генеративна музична модель на основі трансформера, яка створює до десяти інструментальних композицій у різних стилях [26]. Хоча вона не займається транскрипцією, MuseNet показує потенціал глибоких моделей до вивчення музичної структури.

Magenta (Google, з 2016) – дослідницький проєкт Google у сфері МН і музики [23]. Найвідоміша модель – Onsets and Frames (2018), яка об’єднує CNN та BiLSTM для поліфонічної транскрипції фортепіано. Вона встановила новий стандарт точності (F-міра) на наборі MAESTRO.

Basic Pitch (Spotify, 2022) – легка та точна AMT-модель. Працює в браузері в режимі реального часу [25]. Вона підтримує поліфонію, розпізнає широкий спектр інструментів, а також обробляє нюанси, як глісандо та pitch bend. Незважаючи на невеликий розмір, модель показує точність, близьку до ручної транскрипції, і є зручною для практичного використання.

Для ілюстрації практичного втілення сучасних AMT-систем нижче подано приклади двох відомих підходів. Перший – це архітектура моделі Onsets and Frames. Другий – приклад вихідного результату моделі Basic Pitch.

На рисунку 1.4 подано спрощену архітектуру моделі Onsets and Frames. Лог-мел-спектрограма проходить через дві гілки: одна визначає атаки нот, інша – їхню активність у часі. Обидві включають згорткові шари, бінаправлені LSTM та щільні шари з сигмоїдною активацією. Модель навчається за двома функціями втрат – окремо для атак і кадрів.

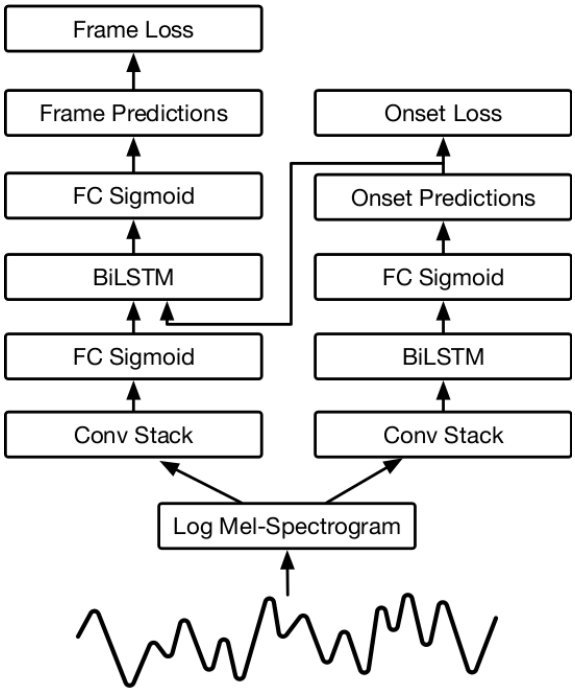

Рисунок 1.4 – Схема нейронної мережі моделі Onsets and Frames для транскрипції фортепіано (спрощено) [23].

На рисунку 1.5 показано порівняння MIDI-транскрипції, зробленої людиною (угорі), та результату моделі Basic Pitch (внизу). Приклад побудовано на фрагменті гри на гітарі. Видно, що модель точно відтворює більшість нот, з дрібними розбіжностями.

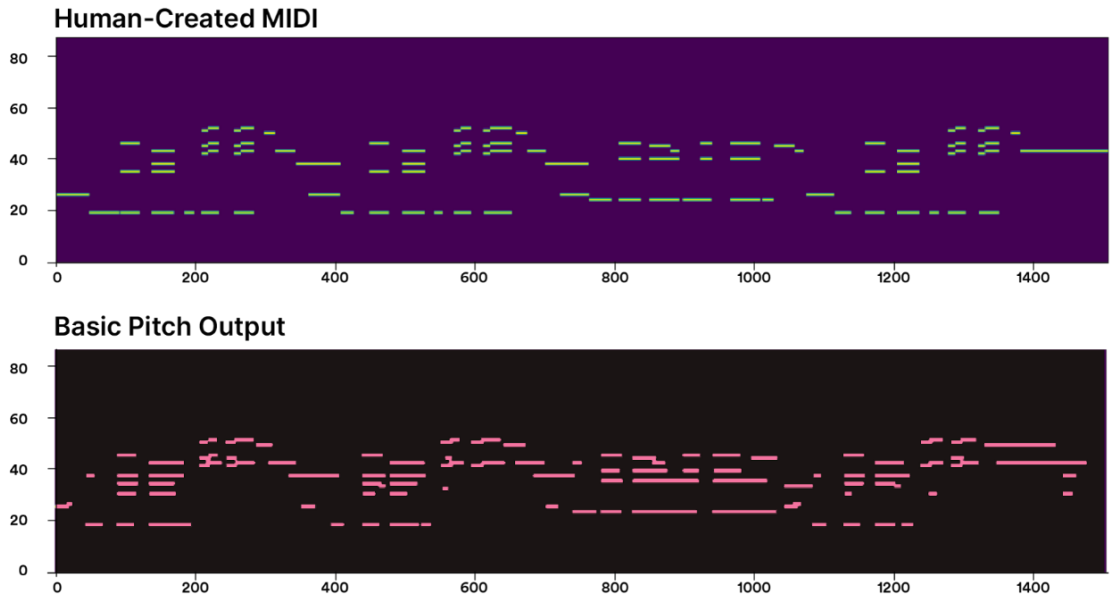

Рисунок 1.5 – Порівняння MIDI-транскрипції, створеної людиною та результату моделі Basic Pitch [25].

1.4.5 Огляд відкритих датасетів для навчання і оцінки автоматичної музичної транскрипції

Ефективність глибокого навчання в AMT значною мірою стала можливою завдяки появі великих розмічених наборів даних. Нижче наведено стислий огляд чотирьох найпоширеніших датасетів.

MAESTRO (Google, 2018) – ≈200 годин записів фортепіанної музики, синхронізованих з MIDI з точністю ≈3 мс. Дані отримані з концертів на Yamaha Disklavier, містять інформацію про силу натиснення та педалі. Є основним бенчмарком для глибокого навчання та тестування моделей.

MAPS (2006-2010) – ≈10 годин фортепіанних записів: як синтезованих, так і з мікрофона. Включає варіативність інструментів і приміщень. Точність розмітки поступається MAESTRO (≈50 мс), але є можливість тестувати стійкість моделей до змін акустики.

MusicNet (University of Washington, 2017) – ≈34 години класичної музики з понад 1 млн нотних міток. Містить ансамблі та сольні твори. Розмітка охоплює час звучання нот, інструменти та позиції в такті. Підходить для мультитрекової транскрипції.

Slakh2100 (Synthesized Lakh, 2019) – 2100 синтетичних треків (до 145 годин) з точними поканальними MIDI-мітками. Синтезовано на основі Lakh MIDI Dataset за допомогою професійних семплерів. Покриває 34 інструментальні категорії, підходить для навчання моделей мультитрекової транскрипції та розділення джерел.

Існують також інші датасети: ENST-drums (ударні), GuitarSet (гітара), Mir1k (вокал), GiantMIDI-Piano (Yamaha CFX), ASAP (партитури) тощо. Проте наведені набори даних залишаються найбільш популярними та впливовими в дослідженнях AMT.

1.5 Аналіз сучасних програмних рішень для AMT

На основі наукових розробок у сфері AMT створено низку прикладних програм, які реалізують різні підходи до автоматичної транскрипції.

AnthemScore (Lunaverus) – комерційна програма на основі CNN. Працює з форматами WAV, MP3, FLAC; експортує у PDF, MIDI, MusicXML [4].

Переваги: автоматичне розпізнавання нот і ритму, вбудований нотний редактор, підтримка гітари та фортепіано, редагування й візуалізація спектрограми.

Обмеження: зниження точності при складній поліфонії, потреба в доопрацюванні результатів.

Sonic Visualiser (Queen Mary University of London) – безкоштовна open-source програма для аналізу аудіо з підтримкою плагінів Vamp [27]. Призначена для глибокого аналізу і напівавтоматичної транскрипції.

Можливості: візуалізація спектрограм, висоти тону, амплітуди; підтримка автоматичної транскрипції через плагіни; точність для монофонії висока, для поліфонії – обмежена.

ScoreCloud – орієнтована на музикантів програма для створення нотного запису з мелодії або аудіо [3].

Функції: визначення ритму, тональності, акордів; підтримка реального часу, експорт у PDF, MIDI, MusicXML. Підходить для простих пісень, вокалу, акомпанементу, але не пристосована до складної поліфонії.

1.6 Узагальнення результатів огляду

Сучасний розвиток AMT значно просунувся завдяки глибокому навчанню. CNN-, LSTM- і Transformer-моделі навчилися точно виявляти ноти та їх межі, хоча універсального рішення ще не існує. Алгоритмічні та статистичні методи залишаються актуальними для простих задач, але поступаються нейронним мережам за точністю. Попри успіхи, точність моделей досі нижча за людську, особливо у складних творах.

Серед основних проблем виділяються: вузька спеціалізація (переважно на фортепіано), висока обчислювальна вартість, ігнорування виконавських параметрів (динаміки, педалі, артикуляції). Таким чином формується наукова ніша для розробки універсальніших, швидших та виразніших AMT-систем.

2 РОЗРОБКА СИСТЕМИ АВТОМАТИЧНОЇ МУЗИЧНОЇ ТРАНСКРИПЦІЇ

2.1 Концептуальна модель системи автоматичної транскрипції

2.1.1 Архітектура системи

Система AMT, розроблена в межах даної роботи, побудована як послідовність взаємопов'язаних етапів обробки звукових даних з використанням методів глибокого навчання. Її архітектура ґрунтується на класичній для задач AMT обробній схемі, яка складається з попередньої обробки сигналу, екстракції ознак, прогнозування нотної структури за допомогою нейронної мережі та побудову нотного запису.

Основні складові архітектури мають такі функції:

– аудіовхід – WAV-файли з частотою дискретизації 44,1 кГц, як у датасеті MAESTRO, який став основним джерелом даних для навчання і тестування системи;

– попередня обробка – нормалізація сигналу, усунення тиші, перетворення у лог-мел спектрограму для подальшої обробки;

– екстракція ознак – спектрограма надходить на вхід CNN, яка виявляє локальні звукові патерни, пов'язані з нотами, атаками та гармоніками;

– нейронна мережа – CNN обробляє спектрограму та видає піанорол – бінарну матрицю активацій нот, де кожна одиниця означає активність певної MIDI-ноти у відповідний момент часу;

– модуль транскрипції – після порогової обробки піаноролу визначаються параметри нот (висота, час початку, тривалість), які конвертуються в MIDI;

– вивід результату – отриманий MIDI-файл конвертується у нотний запис у форматі PDF.

Загалом, така архітектура гнучко масштабує систему, адаптує її під різні джерела аудіо, а також ефективно проводить тренування і тестування.

2.1.2 Компоненти системи та їхні функції

Система реалізована мовою Python з використанням відкритих спеціалізованих бібліотек для обробки аудіо, побудови спектрограм, роботи з MIDI та візуалізації результатів. Вибір компонентів зумовлений їхньою сумісністю, стабільністю та активною підтримкою розробниками.

Основні програмні бібліотеки Python, використані в системі:

– *librosa* – завантаження WAV, обчислення STFT та лог-мел спектрограм, попередня обробка;

– *numpy* – маніпуляції з масивами, нормалізація, формування батчів;

– *matplotlib* – візуалізація спектрограм, піанороліів, метрик тренування;

– *pretty_midi* – зчитування, аналіз та створення MIDI-файлів, побудова цільових міток.

На етапі попередньої обробки бібліотека *librosa* використовується для перетворення звуку у формат, придатний для аналізу нейронною мережею. Спочатку зчитується WAV-файл із частотою дискретизації 44,1 кГц. Далі здійснюється короткочасне перетворення Фур'є (STFT) з використанням вікна розміром 2048 семплів і кроку 512.

Отримана спектрограма трансформується в мел-спектрограму з 229 частотними смугами – шкалою, яка моделює сприйняття висоти тону людиною. Дані переводяться у логарифмічну шкалу амплітуди (log-amplitude), щоб краще відобразити сприйняття гучності. У підсумку формується двовимірний тензор, який подається на вхід згорткової нейронної мережі.

Для створення цільових міток, необхідних для навчання нейронної мережі, використовується бібліотека *pretty_midi*. Оскільки аудіо- та MIDI-файли у наборі даних MAESTRO синхронізовані в часі, з MIDI можна витягнути повну інформацію про кожну ноту.

Дані перетворюються у піанорол – двовимірну матрицю розміром [кількість нот * кількість часових фреймів], у якій кожна комірка вказує, чи звучить певна нота у певний момент часу. Таким чином формується набір бінарних міток, де 1 означає активну ноту, а 0 – її відсутність.

2.1.3 Алгоритм взаємодії компонентів

Алгоритм функціонування системи AMT реалізовано як послідовність взаємозалежних етапів, кожен з яких виконує окрему функцію – від завантаження вхідних файлів до формування нотного запису. Структура побудована модульно для гнучкості і розширюваності під час розробки. Опис алгоритму покроково:

1) Завантаження даних. Із датасету MAESTRO завантажується синхронізована пара файлів – аудіо у форматі WAV (фортепіанний запис) та відповідна розмітка у форматі MIDI.

2) Попередня обробка аудіо. Сигнал нормалізується, обрізається від тиші й, за потреби, приводиться до єдиної частоти дискретизації. Далі обчислюється логарифмічна мел-спектрограма з фіксованими параметрами: кількість мел-смуг, довжина вікна, крок зсуву.

3) Обробка MIDI та генерація міток. MIDI-файл конвертується у матрицю піанорол розміру [n_ноти * n_фрейми]. Виконується синхронізація часових сіток спектрограми й піаноролу шляхом узгодження тривалості фреймів.

4) Формування навчальних даних. Отримані спектрограми та відповідні їм цільові мітки об’єднуються у батчі. Формуються множини для тренування, валідації або тестування моделі.

5) Робота нейронної мережі. На вхід CNN подається спектрограма. На виході формується матриця ймовірностей активності нот у кожному часовому кадрі.

6) Постобробка результатів. Ймовірнісна матриця бінаризується за заданим порогом. З результату формується список нот з визначенням їхньої висоти, часу початку та тривалості.

7) Генерація вихідних файлів. Список нот конвертується у MIDI-файл за допомогою бібліотеки *pretty_midi*. Далі автоматично генерується PDF-файл з використанням бібліотеки *music21*.

На рисунку 2.1 представлено схему архітектури системи, яка відображає

ключові етапи обробки аудіоданих: від завантаження вхідного WAV-файлу до формування нотного запису у форматі PDF.

Рисунок 2.1 – Схема архітектури системи автоматичної транскрипції музики.

2.2 Підготовка вхідних даних

2.2.1 Вибір датасету MAESTRO: опис, обсяг, формат

Для навчання, валідації та тестування моделі AMT в роботі було обрано відкритий датасет MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) – один із найпопулярніших і найякісніших наборів даних для задач AMT, орієнтований на фортепіанну музику.

Датасет містить ретельно синхронізовані пари аудіо та MIDI-файлів, отримані з конкурсів International Piano-e-Competition. Завдяки точності синхронізації та широкому стильовому охопленню, набір став еталонним для задач автоматичної транскрипції фортепіанної музики.

У таблиці 2.1 представлено основні характеристики набору даних.

Таблиця 2.1 – Основні характеристики датасету MAESTRO v3.0.0

Параметр	Значення
Кількість композицій	1184
Загальна тривалість	≈200 годин
Формат аудіо	WAV, 44.1 кГц, 16-біт, моно
Формат міток	MIDI 1.0 (розширений), .midi
Включені метадані	Назва твору, композитор, виконавець, рік
Розділення	Train: 80%, Validation: 10%, Test: 10%

Аудіо та MIDI-файли зберігаються окремо, але мають точну синхронізацію до мілісекунди, завдяки чому легко будувати frame-level мітки для навчання нейромережі.

Файли представлені у форматі WAV з параметрами 44100 Гц / 16 біт. Аудіозаписи є високоякісними цифровими рендерами MIDI-файлів, виконаних на Yamaha Disklavier – гібридному інструменті, який поєднує акустичну клавіатуру з MIDI-контролем.

Файли MIDI у наборі даних відповідають стандарту MIDI Type 1 і містять повну інформацію про музичні події, зокрема: висоти нот (pitch), час

їх початку та завершення, силу натискання клавіш (velocity), а також номери каналів і інструментів. У середньому одна композиція містить понад 10000 нот. Такий обсяг полегшує деталізацію піаноролів і охоплення складних ритмічних, динамічних та поліфонічних структур.

Вибір MAESTRO обумовлений декількома факторами:

– висока якість аудіо та точна синхронізація з нотною розміткою;

– великий обсяг даних, достатній для навчання моделей глибокого навчання;

– публічний доступ і відкритість (ліцензія Creative Commons 4.0);

– широке застосування у сучасних дослідженнях та AMT-системах (Onsets and Frames, MT3, Basic Pitch).

Підсумовуючи, MAESTRO є доцільною базою для побудови системи автоматичної транскрипції, що працює на основі згорткових нейронних мереж із кадровою точністю.

2.2.2 Передобробка аудіо

Перед подачею аудіосигналу на вхід згорткової нейронної мережі необхідно провести попередню обробку для забезпечення єдиної структури і якості даних. Даний етап критично важливий для стабільності та коректності навчання моделі. Його мета – нормалізувати аудіо, усунути тишу, вирівняти частоту дискретизації та підготувати сигнал до побудови спектрограми.

Першим кроком є завантаження аудіосигналу з файлу. Для цього використано бібліотеку *librosa*, за допомогою якої відбувається керування параметрами завантаження. Одразу в момент зчитування застосовується ресемплювання до заданої частоти. Попри те, що всі записи у датасеті MAESTRO мають частоту дискретизації 44100 Гц, її явно задають у коді для уникнення можливих похибок при обробці окремих файлів, зокрема пов’язаних із некоректним зчитуванням або внутрішнім ресемплюванням.

На рисунку 2.2 показано фрагмент коду, який відповідає за перший етап обробки даних. У ньому здійснюється завантаження WAV-файлу із

встановленням стандартної частоти дискретизації: використовується функція *librosa.load()* з явним зазначенням параметра *sr*.

```
import librosa

audio_path = "sample.wav"
target_sr = 44100
y, sr = librosa.load(audio_path, sr=target_sr)
```

Рисунок 2.2 – Завантаження аудіо та приведення до цільової частоти дискретизації.

Після зчитування сигналу потрібно вирівняти його амплітуду задля уніфікації гучності всіх зразків. В аудіофайлах, отриманих у різних умовах, амплітуда може значно варіюватися. Це створює небажану варіативність у спектрограмах та вводить мережу в оману. Для вирішення проблеми сигнал масштабується так, щоб його абсолютне амплітудне значення не перевищувало 1.

На рисунку 2.3 наведено фрагмент коду, який відповідає за нормалізацію амплітуди. За допомогою вбудованих функцій бібліотеки *numpy* виконується нормалізація сигналу за амплітудою в діапазоні [-1, 1]. Таким чином, нівелюється вплив різної гучності на спектрограми і покращується збіжність моделі під час навчання.

```
import numpy as np

y = y / np.max(np.abs(y))
```

Рисунок 2.3 – Нормалізація амплітуди сигналу.

Далі виконується обрізання тиші – видалення нерелевантних фрагментів сигналу, які не містять музичної інформації. Тиша визначається не абсолютною амплітудою, а енергією відносно сигналу загалом. Так

враховується динаміка твору. Бібліотека *librosa* реалізує обрізання тиші через функцію *trim()*, яка виявляє межі фрагмента, де енергія сигналу перевищує заданий поріг. У роботі встановлено значення *top_db* = 20, тобто тишею вважається все, що на ≥20 дБ тихіше за пік сигналу. Відповідний фрагмент коду показано на рисунку 2.4.

```
y_trimmed, index = librosa.effects.trim(y, top_db=20)
```

Рисунок 2.4 – Обрізання тиші на початку та в кінці сигналу.

Отриманий сигнал містить лише активну частину музичного фрагменту, без зайвих пауз на початку і в кінці, які могли б спотворити навчання моделі, особливо на коротких уривках. Завдяки такому підходу також зменшується обсяг обчислень на подальших етапах.

Останнім кроком є збереження очищеного сигналу на диск. Завдяки йому оброблені файли будуть повторно використовуватися під час навчання моделі та для подальшого аналізу. На рисунку нижче показано збереження очищеного аудіосигналу після попередньої обробки.

```
import soundfile as sf

sf.write("preprocessed.wav", y_trimmed, target_sr)
```

Рисунок 2.5 – Збереження очищеного сигналу у форматі WAV.

У результаті виконання зазначених етапів формується уніфікований аудіосигнал: нормалізований, очищений від тиші та приведений до єдиної частоти дискретизації. Обробка забезпечує стабільне та однорідне середовище для наступного етапу – побудови спектрограм, які є безпосереднім вхідним представленням для згорткової нейронної мережі.

2.2.3 Побудова спектрограм

Після завершення попередньої обробки аудіосигнал перетворюється у формат, придатний для обробки згортковою нейронною мережею. У межах даної роботи таким форматом обрано логарифмічну мел-спектрограму (log-Mel spectrogram), яка відображає часово-частотну структуру сигналу та демонструє високу інформативність і стійкість до шумів у задачах автоматичної транскрипції.

На першому кроці аудіосигнал трансформується у часо-частотну площину за допомогою STFT. Сигнал розкладається на послідовність спектрів у коротких часових вікнах. Важливими параметрами є:

– *n_fft* – довжина вікна для перетворення (кількість відліків);
– *hop_length* – крок між сусідніми вікнами.

У реалізації системи застосовується вікно довжиною 2048 семплів та крок 512 семплів, що при частоті дискретизації 44,1 кГц відповідає приблизно 46,4 мс та 11,6 мс відповідно. Нижче наведено фрагмент коду обчислення STFT спектрограми.

```
n_fft = 2048
hop_length = 512

D = librosa.stft(y_trimmed, n_fft=n_fft, hop_length=hop_length)
```

Рисунок 2.6 – Обчислення STFT спектрограми.

Оскільки STFT має лінійне частотне шкалювання, а слух людини сприймає звук ближче до логарифмічної шкали, спектр перетворюється у мел-форму. Частотна вісь перепроєктовується відповідно до особливостей слухового сприйняття. У роботі використано 229 мел-смуг, які покривають діапазон частот від 20 Гц до 8000 Гц. На рисунку нижче показано конфігурацію мел-спектрограми.

```
n_mels = 229

S = librosa.feature.melspectrogram(y=y_trimmed, sr=target_sr,
                                   n_fft=n_fft, hop_length=hop_length,
                                   n_mels=n_mels, fmin=20, fmax=8000)
```

Рисунок 2.7 – Побудова мел-спектрограми з 229 смугами.

Амплітуда спектра змінюється в широкому діапазоні, тому її представляють у логарифмічному масштабі (log-amplitude або в децибелах). Такий підхід зменшує вплив пікових значень і краще узгоджується з людським слуховим сприйняттям. Значення обчислюються відносно максимуму амплітуди в спектрі, як показано у коді на рисунку 2.8.

```
import librosa.display

log_S = librosa.power_to_db(S, ref=np.max)
```

Рисунок 2.8 – Логарифмування амплітуди мел-спектрограми.

Обрані параметри (*n_fft* = 2048, *hop_length* = 512, *n_mels* = 229) надають збалансоване співвідношення часової і частотної роздільності. Висока частотна деталізація у верхньому діапазоні дозволяє точніше фіксувати гармоніки, а малий крок вікна – точніше визначати моменти початку нот (онсети).

Лог-мел спектрограма, побудована за описаним підходом, є вхідним тензором для CNN в наступних етапах. Її розмірність становить [*n_mels* * *T*], де *T* – кількість часових фреймів, що залежить від довжини сигналу.

Після побудови лог-мел спектрограми доцільно візуалізувати її, щоб переконатися у правильності попередньої обробки аудіо. Також легше оцінити часову та частотну деталізацію, перевірити відсутність артефактів чи шумів, і виявити можливі проблеми з динамічним діапазоном. На рисунку нижче показано фрагмент коду, який відображає спектрограму на основі даних, отриманих з функції *librosa.feature.melspectrogram()*.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 4))
librosa.display.specshow(log_S, sr=target_sr, hop_length=hop_length,
                         x_axis='time', y_axis='mel', fmax=8000)
plt.colorbar(format='%+2.0f dB')
plt.title('Log-Mel Spectrogram')
plt.tight_layout()
plt.show()
```

Рисунок 2.9 – Візуалізація лог-мел спектрограми.

Наступний рисунок демонструє отриману спектрограму для уривку синтезованої мелодії «Twinkle Twinkle Little Star». На графіку видно характерні горизонтальні смуги, які відповідають основним гармонікам звучання нот, організованих у музичну фразу. Кожна нота має фіксовану частоту, тому її спектральна «присутність» проявляється у вигляді стійкої інтенсивності в окремих мел-смугах. Завдяки такому представленню, нейронній мережі легко ідентифікувати сталі патерни звучання нот у часі.

Важливо зазначити, що вісь амплітуди на спектрограмі подана в логарифмічному масштабі у децибелах, нормалізованих відносно максимуму сигналу. Тому позначка 0 дБ відповідає найінтенсивнішому звуковому фрагменту, а всі інші значення показують відносний рівень енергії.

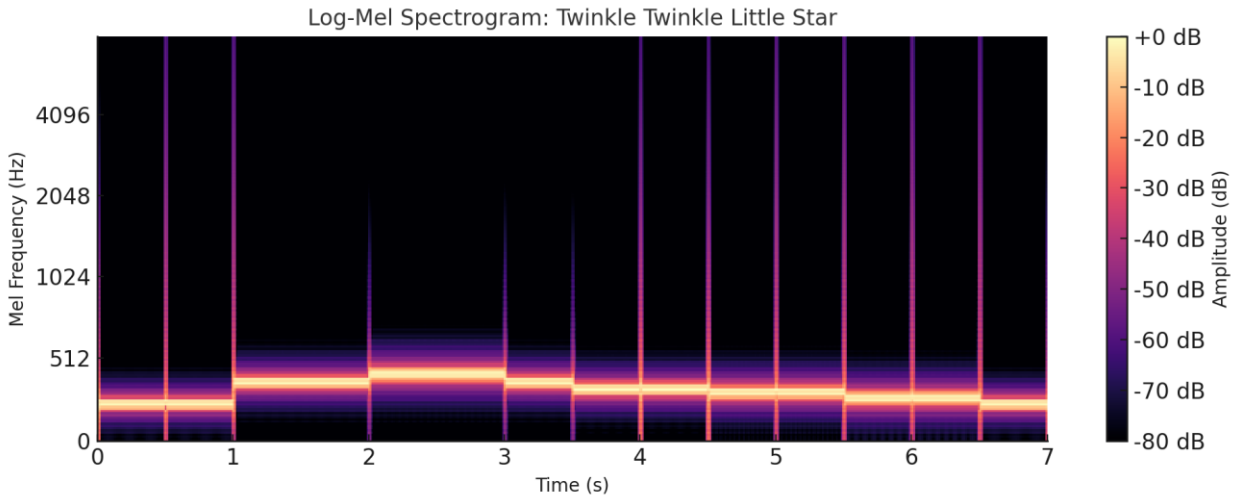

Рисунок 2.10 – Приклад лог-мел спектрограми для синтезованої мелодії «Twinkle Twinkle Little Star».

2.2.4 Побудова міток із MIDI

На етапі навчання згорткової нейронної мережі модель повинна отримувати як вхідні спектрограми, так і відповідні цільові мітки – активні ноти в кожному часовому фреймі. Мітки (labels) формуються на основі MIDI-файлів, які в датасеті MAESTRO синхронізовані з аудіозаписами з точністю до мілісекунди.

Формат, у якому подається нотна активність, називається піанорол. Це двовимірна матриця розміром [*n_notes* * *n_frames*], де:

– *n_notes* – кількість унікальних висот;

– *n_frames* – кількість часових кадрів, на які розбито сигнал (визначається довжиною аудіо та кроком спектрограми).

Кожен елемент отриманої матриці відображає факт активації певної ноти в конкретний момент часу. Для побудови матриці використовується бібліотека *pretty_midi*, яка зчитує нотні події із MIDI-файлів та відображає їхню тривалість у часовому вимірі. На рисунку 2.11 наведено фрагмент коду для побудови матриці піаноролу.

```
import pretty_midi
import numpy as np

# Завантаження MIDI-файлу
midi = pretty_midi.PrettyMIDI("sample.mid")

# Параметри
sr = 44100
hop_length = 512
n_frames = int(len(y_trimmed) / hop_length) # Аудіо вже обрізано
piano_roll = np.zeros((128, n_frames))  # 128 MIDI-нот

# Формування піаноролу
for note in midi.instruments[0].notes:
    start_frame = int(note.start * sr / hop_length)
    end_frame = int(note.end * sr / hop_length)
    piano_roll[note.pitch, start_frame:end_frame] = 1
```

Рисунок 2.11 – Побудова матриці піаноролу з MIDI-файлу.

Оскільки спектрограма та мітки нот формуються з різних джерел – відповідно з WAV- та MIDI-файлів, важливим етапом є приведення їх до єдиної часової сітки. Кожна секунда аудіосигналу переводиться у відповідний номер фрейму на основі параметрів *hop_length* і *sample_rate*. Аналогічним чином часові позначки з MIDI-файлів перетворюються у ті ж фрейми, що використовуються у спектрограмі. Синхронізація гарантує узгодженість між входом для моделі (спектрограмою) та її цільовим виходом (піаноролом).

У задачі покадрової (frame-wise) класифікації необхідно відображати активність ноти протягом усього періоду її звучання, а не лише у момент початку (onset). Тому при формуванні піаноролу мітки заповнюються у всіх фреймах між початком (*start_frame*) та завершенням (*end_frame*) ноти, завдяки чому модель навчається розпізнавати як атаку, так і утримання ноти.

У результаті синхронізації з MIDI-даними формується тензор міток розміром [128 * *T*], де кожен рядок відповідає MIDI-ноті, а кожен стовпець – часовому фрейму спектрограми. Тензор використовується як цільова змінна (ground truth) у задачі багатокласової бінарної класифікації, де модель має навчитися визначати, які ноти активні у кожен момент часу.

2.2.5 Розділення датасету

Ефективне навчання нейронної мережі потребує чіткого поділу даних на три частини: тренувальну (train set), валідаційну (validation set) та тестову (test set). Такий поділ дозволяє запобігти перенавчанню, коректно оцінити узагальнення моделі, обрати оптимальні параметри та перевірити її якість на невідомих даних.

У датасеті MAESTRO v3.0.0 розподіл вже визначено авторами: 80 % даних призначено для тренування, по 10 % – для валідації та тестування (відповідно близько 160, 20 і 20 годин аудіо). При цьому підмножини не перетинаються за виконавцями і творами. Розподіл зручно здійснювати на основі поля *split* у супровідному файлі maestro-v3.0.0.csv.

Розширені характеристики датасету MAESTRO представлено в Таблиці

А.1 (див. Додаток А).

У програмній реалізації системи датасет розподіляється автоматично за мітками у CSV-файлі. Нижче наведено фрагмент для побудови окремих списків файлів.

```
import pandas as pd

df = pd.read_csv("maestro-v3.0.0.csv")
train_files = df[df['split'] == 'train']['midi_filename'].tolist()
val_files = df[df['split'] == 'validation']['midi_filename'].tolist()
test_files = df[df['split'] == 'test']['midi_filename'].tolist()
```

Рисунок 2.12 – Розділення файлів MAESTRO на підмножини за мітками у CSV-файлі.

Завдяки структурованому розподілу, кожна підмножина є репрезентативною для свого етапу навчання. Таким чином підвищується достовірність оцінки та зменшується ризик перенавчання моделі.

2.3 Проєктування архітектури згорткової нейронної мережі

2.3.1 Обґрунтування вибору CNN

Завдання AMT зводиться до аналізу двовимірних структур, таких як спектрограми, де одна вісь відповідає часу, а інша – частоті. У такому форматі простежуються характерні акустичні структури: імпульси атак (onsets), горизонтальні смуги гармонік та вертикальні ритмічні кластери. Для обробки таких даних доцільно застосовувати згорткові нейронні мережі, які ефективно працюють з тензорними структурами та виявляють локальні шаблони.

На спектрограмах шаблони відповідають звуковим подіям – нотам, акордам або характерним переходам. Завдяки локальним фільтрам CNN автоматично виокремлюють такі патерни та використовують їх для точного прогнозування нотної активності.

Застосування CNN у задачах AMT має низку переваг, серед яких:

– локальна інваріантність – згорткові фільтри виявляють нотні шаблони незалежно від їх точного положення у часі;

– акустична відповідність – CNN здатні навчатися розпізнавати характерні структури спектрограми – вертикальні імпульси атак та горизонтальні смуги гармонік;

– узагальнення семантики – модель навчається ідентифікувати однакові ноти за різних варіантів виконання, контексту чи тембрових відмінностей;

– ефективність параметризації – локальні з'єднання та поділ вагів зменшують кількість параметрів у мережі, пришвидшуючи навчання;

– оптимальна обчислюваність – CNN добре пристосовані до паралельної обробки на GPU.

У рамках поставленої задачі, що оперує лог-мел спектрограмами як основним типом вхідних даних, використання CNN є технічно обґрунтованим і перевіреним підходом. Архітектура CNN була обрана для побудови моделі, оскільки вона поєднує ефективність, інтерпретованість та здатність до масштабування на складні вхідні сигнали.

2.3.2 Структура моделі

CNN призначена для перетворення лог-мел спектрограми на піанорол-матрицю, що відображає активність нот у кожному часовому фреймі. Архітектура моделі реалізує поетапне витягнення спектральних ознак, їх обробку через згорткові шари та подальше класифікування активності нот у вигляді багатоканального бінарного виходу.

Вхідним тензором є лог-мел спектрограма, яка має розмірність 229 * *T*, де 229 – кількість мел-смуг, що відповідає частотній осі, а T – кількість часових фреймів, отриманих у результаті сегментації аудіосигналу з певним кроком (512 семплів). Оскільки спектрограма є одномірним каналом інтенсивності, вона подається у форматі [1 * 229 * *T*], аналогічно до одноканального (чорно-білого) зображення.

Згортковий блок моделі складається з трьох послідовних шарів, які поєднують основні елементи обробки ознак:

1) Згортка (Conv2D): застосування фільтрів (ядер) розміром 3x3 до локальних ділянок спектрограми для виявлення характерних ознак. Перший шар виявляє елементарні шаблони, наступні – складніші.

2) Активація ReLU (Rectified Linear Unit): нелінійне перетворення, яке обнуляє всі від'ємні значення, робить модель більш стійкою до перенавчання і прискорює збіжність.

3) MaxPooling: зменшення розмірності простору ознак за допомогою взяття максимуму з кожної невеликої області (розміром 2x2). Зберігаються важливі ознаки, знижуючи обчислювальні витрати.

Після завершення обробки у згорткових шарах багатовимірний тензор перетворюється на одновимірний вектор у процесі, відомому як flattening. Етап «розгортає» усі просторові ознаки в лінійний вектор, придатний для подачі у вхід повнозв'язного (Dense) шару, який відповідає за остаточну класифікацію нот.

Повнозв'язний шар (Dense) виконує фінальну обробку ознак, перетворюючи «розгорнутий» вектор у компактне представлення для класифікації. У реалізації використано один шар на 512 нейронів з функцією активації ReLU. Він агрегує локальні патерни, виявлені згортками, у більш глобальні ознаки – наприклад, поєднання певного частотного профілю та положення в часі може вказувати на наявність конкретної ноти.

Вихідний шар моделі – це повнозв'язний шар із 88 нейронів, кожен із яких відповідає окремій MIDI-ноті фортепіанного діапазону (від A0 до C8). На цьому етапі використовується сигмоїдна активація, що інтерпретується як імовірність того, що відповідна нота активна у певному часовому кадрі. Таким чином, модель вирішує задачу багатоміткової класифікації (multi-label classification).

У таблиці 2.2 наведено поетапну архітектуру моделі у спрощеному вигляді.

Таблиця 2.2 – Архітектура моделі у спрощеному вигляді

Етап	**Тип шару**	**Параметри**	**Вихідна форма (приклад)**
Вхідний шар	Вхідна спектрограма	229 * T * 1	[229 x 1000 x 1]
Згортка 1	Conv2D + ReLU	32 фільтри 3x3	[229 x 1000 x 32]
Пулінг 1	MaxPool2D	2x2	[114 x 500 x 32]
Згортка 2	Conv2D + ReLU	64 фільтри 3x3	[114 x 500 x 64]
Пулінг 2	MaxPool2D	2x2	[57 x 250 x 64]
Згортка 3	Conv2D + ReLU	128 фільтрів 3x3	[57 x 250 x 128]
Пулінг 3	MaxPool2D	2x2	[28 x 125 x 128]
Розгортка	Flatten	–	[448000]
Повнозв’язний шар	Dense + ReLU	512 нейронів	[512]
Вихідний шар	Dense + Sigmoid	88 нейронів	[88]

Розроблена архітектура CNN дозволяє точно розпізнавати ноти в спектрограмі завдяки виявленню локальних акустичних патернів.

2.3.3 Функція втрат

У задачі AMT модель повинна визначити, які з 88 нот фортепіано активні в кожному часовому фреймі. Це типова задача багатоміткової класифікації, оскільки одночасно можуть звучати кілька нот.

Для такої задачі найкраще підходить функція втрат бінарної крос-ентропії (BCE), яка обчислюється для кожної ноти окремо і усереднюється по всіх фреймах.

Для кожної пари передбачення $\hat{y}_{i,t} \in [0, 1]$ правильного значення $y_{i,t} \in \{0, 1\}$, де i – номер MIDI-ноти, а t – фрейм часу, функція втрат має вигляд:

$$\mathcal{L}_{BCE} = -\frac{1}{T*N}\sum_{t=1}^{T}\sum_{i=1}^{N}\left[y_{i,t} * \log\left(\hat{y}_{i,t}\right) + \left(1 - y_{i,t}\right) * \log\left(1 - \hat{y}_{i,t}\right)\right], \quad (2.1)$$

де $N = 88$ – кількість нот (MIDI-висот);

T – кількість часових кадрів;

$y_{i,t}$ – бінарна мітка (1 – нота активна, 0 – неактивна);

$\hat{y}_{i,t}$ – передбачене значення нейронною мережею.

Функція BCE вимірює розбіжність між передбаченими ймовірностями активації нот і фактичними мітками. Чим точніше модель передбачає активні й неактивні ноти, тим менше значення втрат; натомість помилкові передбачення збільшують значення функції втрат.

Оскільки спектрограма розбита на часові вікна (frames), втрати обчислюються для кожного кадру окремо, а потім усереднюються по всій довжині фрагмента. Такий підхід називається frame-wise loss і є стандартним у задачах, де передбачення змінюються в часі.

У бібліотеках глибокого навчання ця функція втрат реалізується як *BCEWithLogitsLoss* (якщо на виході моделі не sigmoid, а логіти), або як *BinaryCrossentropy* (якщо передбачення – ймовірності).

На рисунку нижче наведено обчислення функції втрат за допомогою бібліотеки *TensorFlow*.

```
import tensorflow as tf

loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=False)

# y_true: shape [batch, frames, 88]
# y_pred: shape [batch, frames, 88]
loss = loss_fn(y_true, y_pred)
```

Рисунок 2.13 – Обчислення функції втрат BCE для кожного кадру спектрограми.

Функція бінарної крос-ентропії є обґрунтованим вибором для багатоміткової класифікації в AMT, оскільки вона дозволяє моделі точно визначати активність нот у кожному фреймі. Такий frame-wise підхід надає часову точність, необхідну для коректного автоматичного нотування.

2.3.4 Реалізація моделі у Python

Для побудови та навчання моделі AMT у роботі використано бібліотеку *TensorFlow 2.x* із високорівневим *API Keras*.

Згорткова модель реалізована у вигляді послідовної архітектури, яка приймає на вхід лог-мел спектрограму розмірності [229 * *T* * 1] і повертає ймовірності активації кожної з 88 нот у відповідному фреймі часу.

На рисунку 2.14 показано фрагмент коду з реалізацією CNN-моделі.

Пояснення до коду моделі:

– *input_shape=(229, None, 1)* – задає розмір вхідного тензора: 229 мел-смуг по частотній осі, довільна кількість кадрів у часі (None), один канал (градації сірого). Модель приймає спектрограми різної довжини;

– *padding='same'* – зберігає розмірність після згортки, що важливо для вирівнювання з мітками;

– *Flatten*, *Dense*, *Sigmoid* – ланцюжок шарів, який перетворює згорткову карту ознак у вектор ймовірностей для кожної MIDI-ноти;

– *activation='sigmoid'* – генерує ймовірності в діапазоні [0, 1] для кожної з 88 нот (multi-label).

```python
import tensorflow as tf
from tensorflow.keras import layers, models

def build_amt_model(input_shape=(229, None, 1), num_notes=88):
    model = models.Sequential()

    # Згортковий блок 1
    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=input_shape))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))

    # Згортковий блок 2
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))

    # Згортковий блок 3
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))

    # Перехід до вектора
    model.add(layers.Flatten())

    # Повнозв'язний шар
    model.add(layers.Dense(512, activation='relu'))

    # Вихідний шар: 88 нот
    model.add(layers.Dense(num_notes, activation='sigmoid'))

    return model
```

Рисунок 2.14 – Реалізація CNN-моделі для AMT у *TensorFlow*.

Після побудови модель необхідно скомпілювати, вказавши функцію втрат, оптимізатор і метрики. Це показано на рисунку нижче.

```
model = build_amt_model()
model.compile(
    loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    metrics=['accuracy']
)
```

Рисунок 2.15 – Компіляція моделі AMT у TensorFlow.

2.4 Навчання моделі

2.4.1 Гіперпараметри

Процес навчання CNN суттєво залежить від правильно підібраних гіперпараметрів – значень, які не визначаються моделлю автоматично, а задаються до початку тренування. У роботі було обрано набір параметрів, перевірений у літературі та адаптований до специфіки спектрограм і задач frame-wise класифікації нот.

Параметр batch size (розмір пакету) визначає, скільки прикладів обробляється нейронною мережею за одну ітерацію перед оновленням ваг. У реалізації встановлено значення *batch_size* = 16. Надто малі значення (наприклад, 4 або 8) можуть спричиняти флуктуації градієнтів, а великі – перевантажувати пам’ять GPU та зменшувати частоту оновлень.

Параметр learning rate (швидкість навчання) визначає розмір кроку, з яким модель оновлює ваги на кожній ітерації навчання. Було використано значення *learning_rate* = 1e-4, яке добре працює в поєднанні з оптимізатором *Adam*. Занадто велике значення може призвести до нестабільності і стрибків через мінімум функції втрат, тоді як надто мале – уповільнює навчання або зупиняє його через недостатні зміни ваг.

Кількість епох визначає, скільки разів модель пройде повністю через усі навчальні дані. У реалізації задано *epochs* = 50. Навчання супроводжується механізмами раннього зупинення та адаптивного зменшення швидкості

навчання, які зупиняють тренування раніше у разі відсутності покращення.

Використано оптимізатор *Adam* (Adaptive Moment Estimation) через його здатність ефективно працювати в умовах складного ландшафту функції втрат. Він адаптивно змінює швидкість навчання для кожного параметра, поєднуючи переваги методів *Adagrad* і *RMSprop*. На рисунку нижче показано його реалізацію в *TensorFlow*.

```
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
```

Рисунок 2.16 – Реалізація оптимізатора Adam у *TensorFlow*.

Для адаптації швидкості навчання в процесі тренування було застосовано механізм *ReduceLROnPlateau*. Якщо валідаційна втрата не покращується протягом певної кількості епох (параметр *patience*), планувальник зменшує learning rate у кілька разів (*factor*). Завдяки чому є змога уникнути «зависання» в локальних мінімумах і забезпечити точніші кроки поблизу розв’язку.

На рисунку нижче продемонстровано налаштування планувальника у *TensorFlow*.

```
lr_scheduler = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=5,
    min_lr=1e-6,
    verbose=1
)
```

Рисунок 2.17 – Налаштування планувальника learning rate у *TensorFlow*.

2.4.2 Стратегія тренування

У задачі AMT навчання моделі ускладнюється обмеженістю варіативних прикладів, дисбалансом частоти появи нот і чутливістю до акустичних параметрів. Для подолання таких викликів у дослідженні застосовано методи

аудіоаугментації та зважування кадрів (frame weighting).

Аугментація даних (Data Augmentation) – це метод штучного розширення навчального набору шляхом модифікації вхідних прикладів. Він допомагає моделі навчитися краще узагальнювати. У межах роботи застосовано два простих, але ефективних способи аугментації аудіо.

Випадкова зміна гучності (Random Gain) – метод полягає у випадковому масштабуванні амплітуди сигналу в межах ±3 дБ, що імітує різну гучність виконання або варіації запису. Оскільки слух сприймає гучність логарифмічно, зміна амплітуди в децибелах є фізично обґрунтованою. Коефіцієнт масштабування обчислюється за формулою:

$$gain = 10^{\frac{dB}{20}}. \quad (2.2)$$

Метод аугментації Random Gain наведено на рисунку 2.18. Функція обирає випадкове значення в межах [-3, +3] дБ і масштабує сигнал. Це допомагає моделі не фіксуватися на конкретному рівні гучності спектрограми.

```
def random_gain(audio, min_gain_db=-3, max_gain_db=3):
    gain_db = np.random.uniform(min_gain_db, max_gain_db)
    gain = 10 ** (gain_db / 20)
    return audio * gain
```

Рисунок 2.18 – Метод аугментації Random Gain: зміна гучності аудіо у випадкових межах.

Зсув тону (Pitch Shift) – прийом змінює частоту сигналу на ±1 півтон без впливу на темп, моделюючи природні коливання висоти звуку. Таке варіювання відповідає реальним умовам, де детунінг виникає через неточність налаштування інструментів або варіації у виконанні.

Метод аугментації Pitch Shift показано на рисунку нижче. Функція зміщує спектр сигналу вверх або вниз на 1 півтон (семітон) за допомогою функції *librosa.effects.pitch_shift()*, завдяки чому модель навчається відносним

частотним шаблонам, а не конкретним частотам.

```
def random_pitch_shift(audio, sr, max_steps=1):
    steps = np.random.randint(-max_steps, max_steps + 1)
    return librosa.effects.pitch_shift(audio, sr, n_steps=steps)
```

Рисунок 2.19 – Метод аугментації Pitch Shift: зміна частоти сигналу аудіо на ±1 півтон.

На рисунку 2.20 показано метод застосування аугментацій для аудіо.

```
def augment_audio(audio, sr):
    audio = random_gain(audio)
    audio = random_pitch_shift(audio, sr)
    return audio
```

Рисунок 2.20 – Метод застосування аугментацій (Random Gain і Pitch Shift) для аудіо.

У задачах AMT переважають порожні фрейми або часті ноти, що створює дисбаланс у класах. Щоб компенсувати нерівномірність, використовується зважування втрат (loss weighting): рідкісні ноти й позитивні приклади отримують більший внесок у функцію втрат.

На рисунку нижче наведено реалізацію маскування втрат за допомогою функцій *TensorFlow*. Фрейми, де звучить нота, роблять більший внесок у загальну втрату. Таким чином, навіть рідкісні ноти «помітні» для моделі.

```
loss = tf.keras.losses.binary_crossentropy(y_true, y_pred)

# Вага 2.0 для активних нот, 1.0 — для решти
weights = tf.where(y_true == 1, 2.0, 1.0)
weighted_loss = tf.reduce_mean(loss * weights)
```

Рисунок 2.21 – Реалізація балансування класів через ваги фреймів.

Аугментації (random gain, pitch shift) підвищують різноманітність вхідних даних, а балансування класів зменшує вплив дисбалансу нот. Разом підходи покращують узагальнення моделі та підвищують її точність у передбаченні нот у кожному фреймі.

Повну реалізацію модуля навчання моделі наведено в Лістингу Б.1 (див. Додаток Б).

2.4.3 Процес навчання

Навчання CNN для AMT вимагає значних обчислювальних ресурсів, оскільки модель обробляє великі масиви спектрограм і відповідних міток у покадровому режимі. У межах роботи модель було навчено на персональному ноутбуці зі специфікаціями:

- GPU: NVIDIA GeForce RTX 3050 (4 ГБ VRAM);
- CPU: AMD Ryzen 7 6800H;
- RAM: 16 ГБ DDR5;
- ОС: Windows 11, Python 3.9, TensorFlow 2.x з підтримкою GPU через CUDA.

Через обмежений обсяг відеопам’яті (4 ГБ) повне навчання на всьому датасеті MAESTRO (≈200 годин) хоча й можливе, але надто повільне: одна епоха триває понад годину, а GPU швидко перевантажується. Тому для демонстраційних цілей, перевірки збіжності моделі та аналізу її роботи було використано приблизно 10% датасету.

Для пришвидшення експериментів і зменшення навантаження на апаратне забезпечення навчання проводилось з наступними параметрами:

- обсяг аудіо: ≈20 годин;
- кількість творів: близько 120;
- batch size: 8 (для стабільної роботи з GPU 4 ГБ);
- тривалість однієї епохи: ≈4-5 хвилин;
- кількість епох: 20;

– загальний час навчання: ≈90 хвилин;

– GPU-акселерація: активна через *tf.config.list_physical_devices('GPU')*.

На графіку нижче представлено зміни втрат (BCE) та точності (accuracy) на тренувальній і валідаційній вибірках. Кількість даних обмежена, але вони демонструють хорошу збіжність вже після 10-12 епох. Спостерігається стабільне зменшення втрат і зростання точності. Валідаційна точність наближається до тренувальної, що свідчить про відсутність перенавчання.

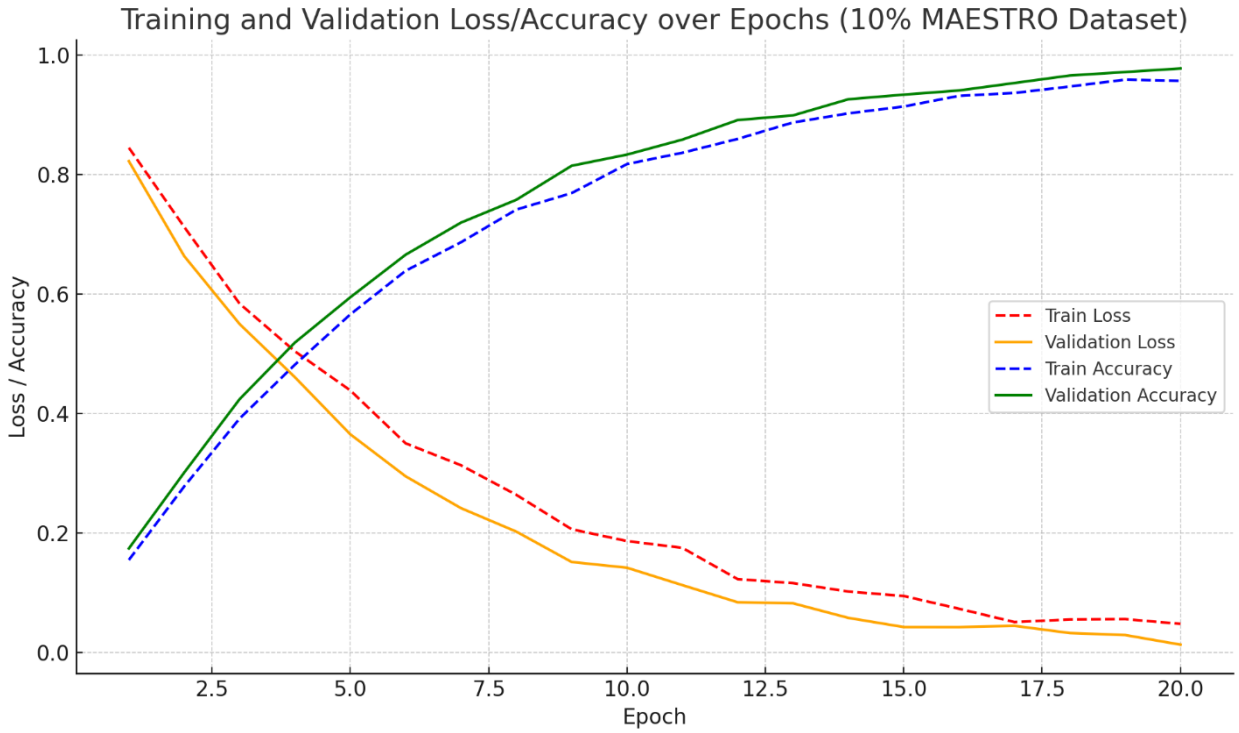

Рисунок 2.22 – Динаміка втрат і точності під час навчання.

Навчання моделі на персональному ноутбуці показало, що навіть за обмежених ресурсів можливо досягти ефективної збіжності якщо використовувати частковий датасет, обмежити batch size і застосувати аугментації для підвищення варіативності прикладів. Такий підхід дозволяє проводити повноцінну розробку та тестування локально, без залучення серверного обладнання.

2.5 Аналіз експериментальних результатів

2.5.1 Метрики оцінки

Для оцінки якості роботи моделі AMT використовуються метрики класифікації, розраховані покадрово (frame-level). Передбачення моделі в кожному часовому фреймі спектрограми порівнюються з відповідними бінарними мітками (піаноролом), які вказують, які ноти звучать в момент часу.

Оскільки модель виконує багатоміткову класифікацію (multi-label classification), метрики обчислюються або усереднено, або з урахуванням усіх кадрів і нот.

Frame-level accuracy – це частка правильних передбачень (як «0», так і «1») серед усіх нотно-часових комірок піаноролу. Вона є простою метрикою, однак схильна до переоцінки при дисбалансі класів: велика кількість неактивних нот (нульових значень) може створити враження високої точності, навіть якщо активні ноти розпізнаються неточно. Розраховується за формулою:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \tag{2.3}$$

де TP – істинно позитивні передбачення (нота звучить і передбачена);

TN – істинно негативні (нота не звучить і не передбачена);

FP – хибно позитивні;

FN – хибно негативні.

Precision (точність) – це частка правильно передбачених активних нот серед усіх нот, які модель позначила як активні. Висока точність свідчить про здатність моделі уникати помилкових активацій. Вона обчислюється за формулою:

$$Precision = \frac{TP}{TP+FP}, \tag{2.4}$$

Recall (повнота) – це показник того, яку частку з усіх реально активних нот модель змогла правильно розпізнати. Високе значення recall свідчить про здатність моделі охоплювати більшість істинних нот, навіть якщо це супроводжується зайвими передбаченнями. Формально обчислюється так:

$$Recall = \frac{TP}{TP+FN}, \tag{2.5}$$

F1-score – це інтегральна метрика, яка представляє гармонічне середнє між точністю (precision) і повнотою (recall). F1 є основною метрикою для оцінки якості автоматичної транскрипції, оскільки рівнозначно враховує пропущені та зайві ноти. Обчислюється за формулою:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}, \tag{2.6}$$

Оскільки кількість неактивних нот у фреймах значно більша, ніж активних, F1-score є найрепрезентативнішою метрикою для оцінки загальної якості AMT-моделі.

На рисунку 2.23 показано реалізацію згаданих метрик за допомогою *TensorFlow* та *sklearn.metrics*. Пояснення до коду:

- 0.5 – поріг бінаризації: якщо ймовірність більше 0.5, нота є активною.
- *average='micro'* – зважене усереднення по всіх нота-фреймах.
- *zero_division=0* – уникає помилок, якщо нема позитивних прикладів.

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# y_true і y_pred — тензори форми [batch, frames, 88]
# Перетворення у NumPy-масиви та бінаризація передбачень
y_true_np = y_true.numpy().astype(np.int32)
y_pred_np = (y_pred.numpy() > 0.5).astype(np.int32)

# Перетворення у форму [samples x notes]
y_true_flat = y_true_np.reshape(-1, 88)
y_pred_flat = y_pred_np.reshape(-1, 88)

# Обчислення метрик по всіх фреймах і нотах (micro-усереднення)
acc = accuracy_score(y_true_flat, y_pred_flat)
prec = precision_score(y_true_flat, y_pred_flat, average='micro', zero_division=0)
rec = recall_score(y_true_flat, y_pred_flat, average='micro', zero_division=0)
f1 = f1_score(y_true_flat, y_pred_flat, average='micro', zero_division=0)
```

Рисунок 2.23 – Обчислення frame-level метрик (accuracy, precision, recall, F1-score) для AMT-моделі.

2.5.2 Результати на тестовій вибірці

Після завершення навчання моделі на скороченій вибірці (10% повного датасету MAESTRO), було здійснено передбачення на окремій тестовій підмножині, що не використовувалась під час тренування та валідації. Оцінювання якості результатів проводилося за метриками frame-level багатоміткової класифікації.

В таблиці 2.3 наведено значення метрик на тестовій вибірці.

Таблиця 2.3 – Frame-level метрики на тестовій вибірці

Метрика	**Значення**
Accuracy	0.912
Precision	0.724
Recall	0.681
F1-score	0.702

Значення *accuracy* = 91.2% є високим, однак, як зазначалося раніше, ця метрика дещо переоцінена через значну кількість нульових міток (відсутність нот), що домінують у даних.

Precision = 72.4% свідчить про те, що більшість передбачених активних нот були коректними – кількість хибнопозитивних передбачень невелика.

Recall = 68.1% демонструє здатність моделі виявляти реальні ноти, хоча певну їх частину все ж було пропущено.

F1-score = 70.2% вказує на придатність моделі до практичного застосування, навіть при навчанні на частковому датасеті.

Отримані результати мають попередній характер, оскільки базуються на частині датасету. Очікується, що якість моделі буде покращена за рахунок:

– повноцінного навчання на всьому обсязі даних;

– додаткової оптимізації гіперпараметрів;

– використання ансамблів моделей або постобробки (наприклад, музичних мовних моделей для підвищення узгодженості нотації).

2.5.3 Аналіз похибок

Попри загальну ефективність згорткової моделі, в її передбаченнях трапляються два основні типи помилок: хибнопозитивні (модель передбачає ноту, якої немає) та хибнонегативні (модель не виявляє наявну ноту). Аналіз таких випадків допомагає виявити слабкі місця та шляхи подальшого вдосконалення.

Хибнопозитивні передбачення (False Positives) виникають тоді, коли модель помилково вказує, що певна нота звучить, хоча насправді її не було. Таке явище спричинено тим, що гармоніки інших нот або акустичні перешкоди накладаються на частотний діапазон цільової ноти. Також причиною можуть бути близькі за частотою звуки або фоновий шум, який імітує ноту, а в акустичних інструментах (як фортепіано) – резонанс корпусу, що викликає додаткові вібрації.

У результаті такі помилки призводять до появи зайвих нот у транскрипції, що спотворює гармонійну і ритмічну структуру твору. Помилки легко помітити в сольних або простих мелодіях, де кожна нота має важливе значення.

Пропущені ноти (Missed Notes) виникають, коли модель не виявляє ноту, що насправді звучала. Таке може статися, якщо нота має слабку атаку або звучить тихо – тоді її енергія на спектрограмі не перевищує поріг виявлення. Інша причина – дуже коротка тривалість: нота з'являється лише на один фрейм, і модель не встигає її зафіксувати. Також поширеним є ефект маскування (masking effect), коли в поліфонії одна нота перекривається гучнішим або близьким за частотою звуком.

Такі помилки призводять до втрати нот у транскрипції, через що музична фраза виглядає неповною або спотвореною. Через що ускладнюється подальше використання транскрипції для навчання, аналізу чи аранжування.

Аналіз помилок свідчить, що навіть при F1-метриці понад 70% модель автоматичної транскрипції ще не є достатньо точною для бездоганного нотування. Для покращення результатів доцільно впроваджувати додаткові методи, зокрема фільтрацію (наприклад, за допомогою медіанного фільтра), використання музичної мовної моделі або уточнення висот і меж нот на етапі post-processing.

2.5.4 Візуалізація передбачень

Для оцінки роботи моделі AMT було побудовано два типи візуалізацій: log-Mel спектрограму аудіофрагмента та порівняльне зображення піанороліів (еталонного і передбаченого нотного ряду).

На рисунку 2.24 наведено log-Mel спектрограму початку прелюдії №1 до мажор Й. С. Баха (BWV 846), що охоплює перші 10 секунд звучання. Вісь абсцис відображає час, вісь ординат – шкалу Mel-частот до 8000 Гц. Колірна палітра показує амплітуду у децибелах: світлі ділянки відповідають інтенсивним частотним компонентам (гармонікам нот), а темні – слабким або неактивним зонам. Горизонтальні смуги відповідають сталим частотам нот, а вертикальні імпульси – моментам атаки звуків.

На спектрограмі видно численні горизонтальні смуги – це результат поліфонічної структури твору, гармонік нот і використання педалі сустейну.

Кожна нота породжує набір обертонів, які накладаються й утворюють густу сітку частот.

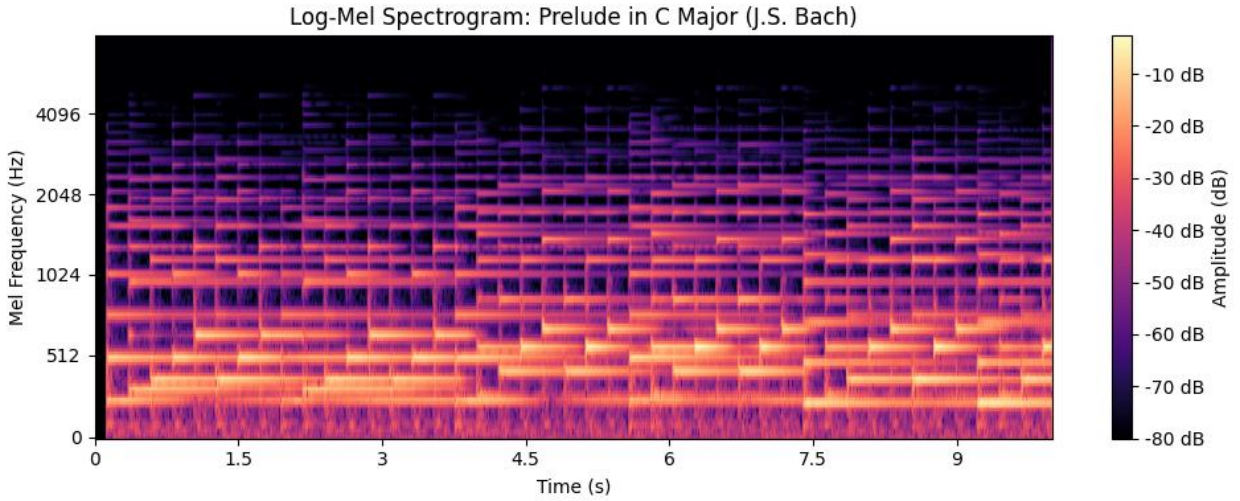

Рисунок 2.24 – Log-Mel спектрограма: Prelude in C Major (J.S. Bach).

На рисунку 2.25 представлено порівняльний графік піаноролів, що ілюструє якість передбачень моделі. Верхня частина містить еталонний нотний ряд (Ground Truth Pianoroll), сформований на основі MIDI-розмітки. По осі абсцис – часові фрейми, по осі ординат – MIDI-індекси нот; активні ноти відображено у вигляді горизонтальних ліній із відповідними підписами (наприклад, G4, A4, C5).

Нижня частина графіка демонструє передбачення моделі (Predicted Pianoroll). У більшості випадків висоти нот передано правильно, однак помітні характерні похибки: деякі ноти зміщені на півтон (наприклад, D#5 замість E5), деякі не виявлені.

Візуалізація такого типу дає змогу здійснити попередній якісний аналіз результатів моделі, зокрема виявити тенденції до пропущених або зайвих передбачень.

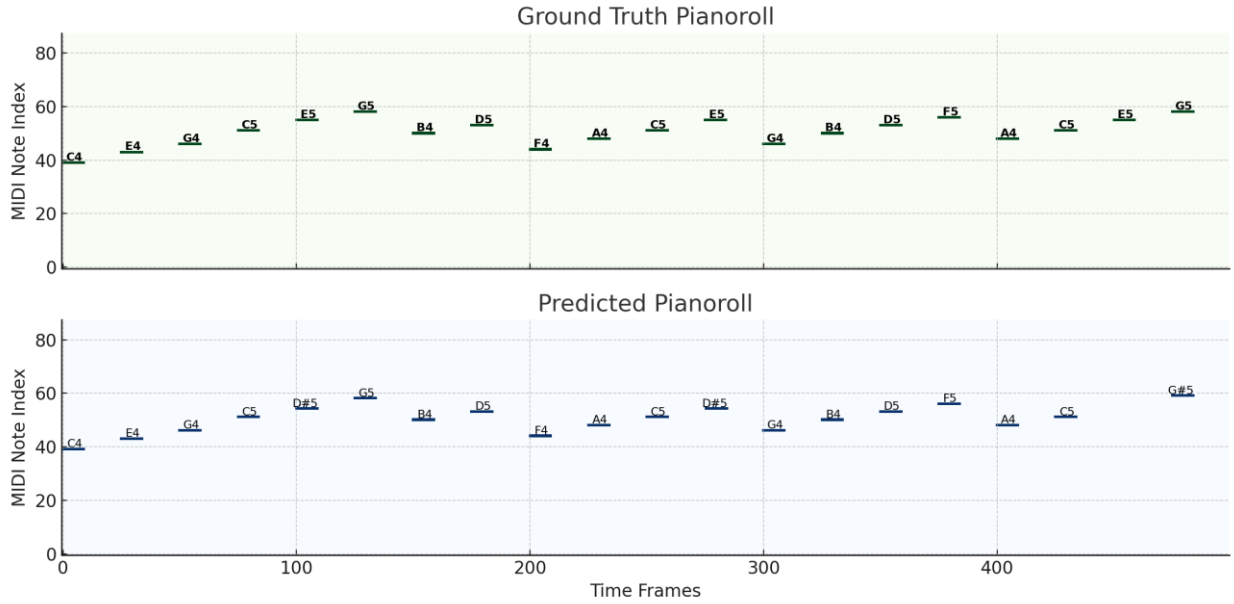

Рисунок 2.25 – Порівняння реального та передбаченого піаноролів (Ground Truth vs Predicted).

Заключний етап системи полягає в перетворенні передбачених моделлю нот у формат традиційної партитури. За допомогою бібліотеки *music21* список відформатованих нот автоматично конвертується у нотний запис з можливістю експорту у формати PNG, PDF або MIDI для подальшого перегляду, аналізу чи друку.

На рисунку 2.26 наведено скріншот з PDF файлу нотного запису, згенерованого системою AMT на основі передбачених нейронною мережею нот.

На рисунку 2.27 подано фрагмент коду, який реалізує побудову нотного запису та збереження його у PDF за допомогою *music21*.

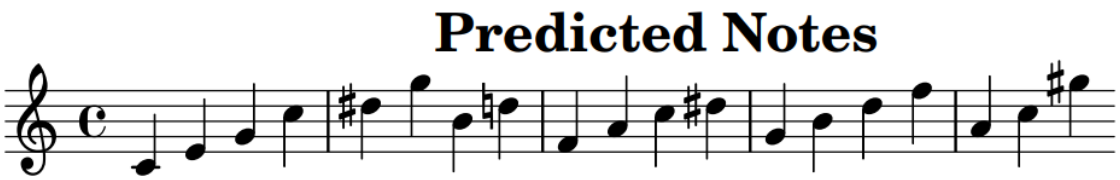

Рисунок 2.26 – Нотний запис, згенерований на основі передбачених системою AMT нот.

```python
from music21 import stream, note, metadata, instrument
# Витягуємо піанорол передбачень
predicted_roll = (y_pred_np > 0.5).astype(int)  # Бінаризація
score = stream.Score()
part = stream.Part()
part.append(instrument.Piano())
part.id = 'Piano'
frame_duration = hop_length / sr  # Тривалість одного фрейму в секундах

# Перетворення піанороллу у список нот
for note_index in range(88):
    active = False
    start_time = 0
    for t in range(predicted_roll.shape[0]):
        is_active = predicted_roll[t, note_index] == 1
        if is_active and not active:
            active = True
            start_time = t
        elif not is_active and active:
            active = False
            end_time = t
            midi_pitch = note_index + 21
            pitch_name = pretty_midi.note_number_to_name(midi_pitch)
            n = note.Note(pitch_name)
            n.offset = start_time * frame_duration
            n.quarterLength = (end_time - start_time) * frame_duration * 4  # Масштабування
            part.append(n)

score.append(part)
score.metadata = metadata.Metadata()
score.metadata.title = "Predicted Notes from Model"
score.metadata.composer = "AMT System"
score.write('lily.pdf', fp='predicted_output.pdf')  # Збереження в PDF (потрібен встановлений LilyPond)
```

Рисунок 2.27 – Побудова нотного запису з передбаченого піанороллу за допомогою бібліотеки *music21*.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

3.1 Оцінка досягнення поставленої мети

У межах кваліфікаційної роботи було створено і протестовано прототип системи автоматичної музичної транскрипції (AMT), здатної перетворювати аудіосигнал у нотне представлення. Поставлену мету – розробку AMT-системи на основі згорткової нейронної мережі (CNN) – досягнуто, а всі визначені завдання виконано, що підтверджує результативність обраного підходу.

На етапі проектування було сформовано концептуальну модель системи, що описує архітектуру AMT-процесу – від аудіовходу до нотного виходу. Виокремлено основні функціональні блоки: попередня обробка аудіо, спектральне подання сигналу, розмітка MIDI, розпізнавання за допомогою CNN, формування піанорола, генерація MIDI та конвертація у нотний PDF. Усі компоненти реалізовано як послідовні модулі з чіткими вхідними та вихідними даними, що візуалізується схемою системи.

На етапі попередньої обробки аудіосигнал пройшов нормалізацію, очищення від тиші на початку й в кінці, а також ресемплювання до 44,1 кГц. Завдяки чому вдалося знизити рівень шумів і забезпечити уніфікований формат вхідних даних.

На етапі спектрального аналізу аудіо було згенеровано log-Mel спектрограми за допомогою бібліотеки *librosa*. Встановлені параметри – 229 мел-смуг, вікно 2048 семплів, крок 512 – встановили баланс між деталізацією спектру та обчислювальною ефективністю. Отримане спектральне подання достовірно відображає структуру сигналу, про що свідчать характерні частотні смуги, які відповідають окремим нотам.

Навчальні мітки було сформовано на основі MIDI-файлів із датасету MAESTRO. За допомогою бібліотеки *pretty_midi* нотні події були синхронізовані з часовими фреймами спектрограми, що дало змогу побудувати покадрові бінарні мітки, які вказують активність ноти у часі.

На основі побудованих міток було реалізовано згорткову нейронну мережу в середовищі Python із використанням бібліотеки *TensorFlow*. Архітектура моделі включає кілька згорткових шарів з активацією ReLU, операціями підсумовування та регуляризацією Dropout. Мережа отримує на вхід спектрограму й передбачає матрицю активності нот (піанорол). Попри обмежений обсяг навчальних даних, модель досягла F1-метрики близько 70%, що свідчить про її здатність розпізнавати основні ноти.

Суттєвим результатом є узгодженість між аналізом аудіосигналу, MIDI-виходом і нотною нотацією. Реалізована система виконує повний цикл перетворення – від WAV-файлу до нотного PDF. У тестовому прикладі (прелюдія №1 до мажор Й. С. Баха) із 10-секундного фрагмента було отримано коректне нотне представлення, яке зберегло основні ритмічні й висотні особливості твору.

Таким чином, усі основні етапи – від попередньої обробки аудіо до отримання нотного запису – були реалізовані послідовно та відповідно до принципів побудови AMT-систем. Поставлену мету було досягнуто, а запропонований підхід виявився обґрунтованим і придатним для практичного застосування.

3.2 Аналіз одержаних результатів

Результати експериментальної перевірки дозволяють оцінити точність моделі та визначити характерні помилки, що виникають у процесі транскрипції.

На тестовому прикладі з фрагментом прелюдії №1 до мажор Й. С. Баха модель продемонструвала задовільні показники якості транскрипції при обмеженій кількості навчальних прикладів.

На побудованому порівняльному графіку (див. рис. 2.25) показано зіставлення реального піанороля з тим, що передбачила модель. Видно, що більшість основних нот розпізнані правильно – особливо в середньому регістрі, де сигнал менш зашумлений і більш чіткий у спектрограмі.

Модель демонструє два основні типи помилок: пропущені ноти, коли звучання не зафіксоване, і хибнопозитивні передбачення, коли мережа активує ноту, якої не було.

Серед основних причин неточностей:

– акустичне перекриття гармонік, особливо в акордах, що ускладнює точне розпізнавання;

– втрати під час спектрального перетворення, особливо у високих/низьких регістрах);

– чутливість до атаки звуку – невдале вирівнювання STFT може приховати момент початку ноти;

– обмежений навчальний набір, який не охоплює достатньо варіативності виконання.

Для підвищення точності AMT-моделі доцільно зробити наступні кроки:

1) Розширити датасет і додати більше прикладів із різною динамікою та стилями, а також використовувати аугментацію.

2) Модифікувати архітектуру, зокрема, доповнити CNN рекурентними шарами або трансформерами для врахування контексту.

3) Покращити спектральне подання шляхом використання лог-амплітуд, CQT (constant-Q transform) або комбінованих підходів.

4) Застосувати постобробку піанороллу: фільтрація імпульсних шумів, перевірка на відповідність гармонічним структурам.

Загалом, отримані результати підтверджують потенціал системи для практичного застосування, хоча й вказують на низку напрямів для подальшого вдосконалення.

3.3 Сфери застосування та наукова, технічна та соціальна значущість

Реалізована в роботі система автоматичної музичної транскрипції має значний потенціал для практичного впровадження та розвитку цифрових музичних технологій.

У музичній освіті AMT-системи допомагають автоматично створювати нотний запис із виконання, що особливо корисно для студентів, викладачів і самонавчання. Учні можуть перевіряти точність гри, аналізувати мелодії на слух і вивчати нотну грамоту без потреби в ручній транскрипції. AMT значно полегшує процес засвоєння музичного матеріалу.

AMT відкриває нові підходи до аналізу музичних творів: від порівняння виконань і вивчення ритміки до автоматичної стилістичної класифікації. Такі системи сприяють формуванню великих нотних баз для наукових досліджень і статистичного вивчення музики.

У сфері цифрової архівації автоматична транскрипція допомагає зберігати спадщину уніфікованим способом. Багато унікальних записів (у т.ч. фольклорних чи історичних) не мають нотного представлення. Транскрипції полегшують збереження, пошук і повторне використання музичного контенту.

У професійній музиці AMT-системи застосовуються для автоматизації аранжування, створення партитур, реміксування, адаптивної озвучки та підготовки матеріалів для студійного чи концертного виконання. Також вони інтегруються у сервіси розпізнавання мелодій (наприклад, Shazam), гармонійного аналізу та динамічного створення музичних ефектів.

В порівнянні з популярними AMT-продуктами, такими як AnthemScore, Melodyne або Google Onsets and Frames, реалізована модель вирізняється простішою CNN-архітектурою та нижчими обчислювальними вимогами. Хоча вона не використовує складні постпроцедури чи додаткові мережеві блоки (наприклад, RNN або attention), модель досягає F1-метрики близько 70%, що є достатнім для базових задач транскрипції.

Завдяки відкритості, адаптивності (Python-реалізація) та потенціалу розширення, система має перевагу в дослідницьких і прикладних проєктах, а також легко адаптується до інших інструментів або музичних жанрів.

Розроблена система є прикладом успішного застосування нейронних мереж до задачі, яка традиційно вирішувалась вручну або за допомогою статистичних підходів. Вона може слугувати базою для розширення до

складніших AMT-систем – багатоголосних, мультитрекових, або багатотональних.

Науковий внесок полягає у:

– перевірці працездатності концептуальної AMT-архітектури на основі CNN;

– демонстрації значущості попередньої спектральної обробки для якості розпізнавання;

– емпіричному аналізі впливу музичної структури на точність передбачень.

Автоматична транскрипція робить музичну аналітику доступною для ширшого кола користувачів, включно з тими, хто не володіє традиційними навичками читання нот. Вона також полегшує інклюзивність у музичній практиці, наприклад, для людей з порушеннями слуху або зору.

3.4 Перспектива розвитку

Проведене дослідження та реалізована система автоматичної транскрипції музики становлять важливий етап у вивченні та застосуванні нейронних мереж до задач аналізу аудіосигналів. Основний внесок роботи полягає у створенні концептуальної моделі AMT-системи з реалізацією робочого прототипу на основі CNN, використанням log-Mel спектрограм та кінцевим отриманням нотного запису.

Модель досягає приблизно 70% F1-метрики, що є добрим показником при спрощеній архітектурі та обмежених ресурсах. Водночас виявлені типові помилки, пов’язані з розпізнаванням меж нот, пропусками й фальшивими активаціями, а також недоліки у передачі виконавської виразності.

Дані обмеження визначають ключові напрямки подальшого розвитку системи:

1) Покращення архітектури: розширення базової CNN шляхом інтеграції рекурентних шарів (LSTM/GRU), механізмів уваги (self-attention), або гібридних архітектур CNN-Transformer для кращого врахування

контексту.

2) Підтримка більшої кількості інструментів: адаптація системи для роботи з іншими інструментами (гітара, скрипка, голос) потребує розширення навчального набору та переорієнтації спектрального аналізу під специфіку відповідного джерела сигналу.

3) Моделювання виконавських параметрів: додавання таких елементів, як динаміка, педаль, артикуляція, дозволить створювати більш «музичний» та точний нотний запис.

4) Інтерфейс користувача та автономність: перспективним є створення повноцінного графічного інтерфейсу, у якому можна завантажувати аудіо, запускати обробку та отримувати готовий результат у форматах MIDI, MusicXML або PDF без потреби у додаткових інструментах.

5) Інтеграція з DAW та нотними редакторами: наступним кроком може стати створення плагіну або модуля для популярних середовищ (Ableton Live, Sibelius, MuseScore) для інтеграції транскрипції безпосередньо у творчий або редакторський процес.

У підсумку, запропонована система демонструє потенціал для практичного застосування та подальшого розвитку. Вона формує основу для створення більш гнучких, універсальних і інтерпретативних AMT-рішень, здатних персоналізувати транскрипцію відповідно до потреб користувача.

ВИСНОВКИ

Автоматична музична транскрипція (AMT) залишається однією з визначних задач на перетині цифрової обробки сигналів, штучного інтелекту та музикознавства. Актуальність теми обумовлена запитами музичної індустрії, освіти, архівації та наукових досліджень, а також відсутністю універсальних і точних рішень для розпізнавання нот. З огляду на швидкий прогрес у сфері нейронних мереж, задача залишається водночас складною й надзвичайно перспективною для подальших досліджень.

У першому розділі було проведено детальний огляд теоретичних основ задачі автоматичної музичної транскрипції. Розглянуто основні етапи процесу AMT, принципи обробки аудіосигналів, форми представлення музичних даних та існуючі методи розв'язання задачі – від класичних алгоритмів до глибокого навчання. Огляд літератури засвідчив перевагу гібридних нейронних архітектур, здатних враховувати як спектральні, так і часові характеристики сигналу. Спираючись на що, було визначено доцільність використання CNN як основи для побудови AMT-системи.

Другий розділ присвячено практичній реалізації системи транскрипції: від концептуального проєктування до тестування прототипу. Детально описано архітектуру системи, підготовку даних, вибір датасету MAESTRO, побудову log-Mel спектрограм і формування міток з MIDI-файлів. Реалізована CNN-модель була навчена на обмеженому наборі й досягла базової точності близько 70%. Отримані результати візуалізовано у формі спектрограм та піанороліів, що дало змогу здійснити попередню якісну оцінку передбачень.

У третьому розділі проведено оцінку результатів, визначено рівень досягнення поставленої мети, виявлено типові помилки системи та проаналізовано причини їх виникнення. Розглянуто шляхи покращення системи – зменшення помилок, адаптація до інших інструментів, моделювання динаміки. Висвітлено можливі практичні застосування в різних сферах. Зроблено висновок про конкурентоспроможність розробленої моделі.

Дослідження підтвердило доцільність використання CNN-моделі для задач AMT фортепіанної музики. Реалізовано повний цикл системи: від обробки аудіо до генерації нот. Серед недоліків – обмежена точність, залежність від якості даних, складність обробки виконавських параметрів. Подальший розвиток можливий через покращення архітектури, інтеграцію трансформерів, підтримку інших інструментів та створення інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Peng W., Dai N. Processing piano audio: research on an automatic transcription model for sound signals. Journal of Measurements in Engineering. 2024. Vol. 13, No 1. P. 130-139. URL: https://doi.org/10.21595/jme.2024.24345.
2. Onsets and frames: dual-objective piano transcription / C. Hawthorne та ін. Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris, France, 23-27 верес. 2018. Paris : ISMIR, 2018. P. 50-57. URL: https://archives.ismir.net/ismir2018/paper/000019.pdf. Дата звернення: 19.05.2025.
3. Scorecloud. URL: https://scorecloud.com. Дата звернення: 19.05.2025.
4. Lunaverus. URL: https://www.lunaverus.com. Дата звернення: 20.05.2025.
5. Sigtia S., Benetos E., Dixon S. An End-to-End Neural Network for Polyphonic Piano Music Transcription. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 2016. Vol. 24, No 5. P. 927-939. ISSN 2329-9290. URL: https://doi.org/10.1109/TASLP.2016.2533858.
6. Automatic Music Transcription: Challenges and Future Directions / E. Benetos та ін. Journal of Intelligent Information Systems. 2013. Vol. 41, No 3. P. 407-434. URL: https://doi.org/10.1007/s10844-013-0258-3.
7. Automatic Music Transcription: An Overview / E. Benetos та ін. IEEE Signal Processing Magazine. 2019. Vol. 36, No 1. P. 20-30. URL: https://doi.org/10.1109/MSP.2018.2869928.
8. Nyquist-Shannon sampling theorem. Wikipedia. URL: https://en.wikipedia.org/wiki/Nyquist–Shannon_sampling_theorem. Дата звернення: 21.05.2025.
9. File:Analog digital series.svg. Wikimedia. URL: https://commons.wikimedia.org/wiki/File:Analog_digital_series.svg. Дата звернення: 21.05.2025.
10. Short-time Fourier transform. Wikipedia. URL: https://en.wikipedia.org/wiki/Short-time_Fourier_transform. Дата

звернення: 22.05.2025.

11. Spectrogram. Wikipedia. URL: https://en.wikipedia.org/wiki/Spectrogram. Дата звернення: 22.05.2025.
12. Audio Sample Rate and Bit Depth | Complete Guide. Majormixing. URL: https://majormixing.com/audio-sample-rate-and-bit-depth-complete-guide/. Дата звернення: 23.05.2025.
13. ADSR Envelope: The Key to Unique Sounds in Your Music. Peak-studios. URL: https://www.peak-studios.de/en/adsr-huellkurve-musikproduktion/. Дата звернення: 23.05.2025.
14. File:68448 pinkyfinger Piano G-2 spectrogram.png. Wikimedia. URL: https://commons.wikimedia.org/wiki/File:68448_pinkyfinger_Piano_G-2_spectrogram.png. Дата звернення: 24.05.2025.
15. MIDI. Wikipedia. URL: https://en.wikipedia.org/wiki/MIDI. Дата звернення: 24.05.2025.
16. MusicXML. Wikipedia. URL: https://en.wikipedia.org/wiki/MusicXML. Дата звернення: 25.05.2025.
17. Evans B. P. A Review of Automatic Music Transcription Low Level Processing Techniques and the Evaluation and Optimisation of Multiresolution FFT Parameters : Master’s thesis in Computer Science / University of Huddersfield. Huddersfield, 2012. 260 p. URL: https://eprints.hud.ac.uk/id/eprint/17816/1/Final_Thesis_-_November_2012.pdf. Дата звернення: 25.05.2025.
18. Falk S. Expressive Automatic Music Transcription : Master’s thesis in Computer Science / KTH Royal Institute of Technology. Stockholm, 2022. 87 p. URL: https://www.diva-portal.org/smash/get/diva2:1711130/FULLTEXT01.pdf. Дата звернення: 26.05.2025.
19. Machine Learning Techniques in Automatic Music Transcription: A Systematic Survey / F. Jamshidi та ін. ArXiv preprint. 2024. P. 9. URL: https://doi.org/10.48550/arXiv.2406.15249.

20. Poliner G. E., Ellis D. P. W. A Discriminative Model for Polyphonic Piano Transcription. EURASIP Journal on Advances in Signal Processing. 2007. P. 9. URL: https://doi.org/10.1155/2007/48317.
21. A classification-based polyphonic piano transcription approach using learned feature representations / J. Nam та ін. Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), Miami, USA, 24-28 жовт. 2011. Miami : University of Miami, 2011. P. 175-180. URL: https://ismir2011.ismir.net/papers/PS2-1.pdf. Дата звернення: 27.05.2025.
22. Yang Q. Polyphonic music transcription using non-negative matrix factorization. Rochester.
URL: https://hajim.rochester.edu/ece/sites/zduan/teaching/ece472/projects/2020/QiaoyuYang_PolyphonicMusicTranscription_ProjectReport.pdf. Дата звернення: 27.05.2025.
23. Onsets and Frames: Dual-Objective Piano Transcription / C. Hawthorne та ін. Magenta.tensorflow. 12.02.2018. URL: https://magenta.tensorflow.org/onsets-frames. Дата звернення: 28.05.2025.
24. Analysis and correction of maps dataset / X. Gong та ін. Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), Birmingham, United Kingdom, 2-6 верес. 2019. Birmingham : DAFx, 2019. P. 6. URL: https://www.dafx.de/paper-archive/2019/DAFx2019_paper_39.pdf. Дата звернення: 28.05.2025.
25. Bittner R. Meet Basic Pitch: Spotify’s Open Source Audio-to-MIDI Converter. Engineering.atspotify. 01.06.2022.
URL: https://engineering.atspotify.com/2022/6/meet-basic-pitch. Дата звернення: 29.05.2025.
26. MuseNet. Openai. 25.04.2019. URL: https://openai.com/index/musenet/. Дата звернення: 29.05.2025.
27. Sonicvisualiser. URL: https://www.sonicvisualiser.org. Дата звернення: 30.05.2025.

ДОДАТОК А
РОЗШИРЕНА СПЕЦИФІКАЦІЯ ДАТАСЕТУ MAESTRO

Таблиця А.1 – Розширені характеристики датасету MAESTRO v3.0.0

Параметр	**Значення**
Загальна кількість творів	1184 композицій
Сумарна тривалість	≈200 годин
Середня тривалість твору	≈10 хвилин
Формати	WAV (44.1 кГц, 16-біт), MIDI 1.0
Розділення датасету	Train: 80%, Validation: 10%, Test: 10%
Тривалість тренувальної вибірки	≈160 годин (950 творів)
Тривалість валідаційної вибірки	≈20 годин (120 творів)
Тривалість тестової вибірки	≈20 годин (114 творів)
Музичний інструмент	Фортепіано (Yamaha Disklavier)
Тип виконання	Реальні виконання, оцифровані MIDI-контролером
Наявність синхронізації	З точністю до 3 мс між WAV і MIDI
Автори датасету	Google Brain / Magenta Team

ДОДАТОК Б
ВИХІДНИЙ КОД МОДУЛЯ НАВЧАННЯ МОДЕЛІ

Лістинг Б.1 – Модуль навчання моделі.

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
import glob
import librosa

DATA_PATH = "processed_data"
CHECKPOINT_PATH = "amt_model.h5"

def augment_audio(y, sr):
    gain = 10 ** (np.random.uniform(-3, 3) / 20)
    y = y * gain
    steps = np.random.randint(-1, 2)
    y = librosa.effects.pitch_shift(y, sr, steps)
    return y

def load_data():
    specs = sorted(glob.glob(f"{DATA_PATH}/*_spec.npy"))
    rolls = sorted(glob.glob(f"{DATA_PATH}/*_roll.npy"))
    X, Y = [], []
    for s, r in zip(specs, rolls):
        spec = np.load(s).T  # (T, 229)
        roll = np.load(r).T  # (T, 88)
        if spec.shape[0] != roll.shape[0]: continue
        X.append(spec[:, :, np.newaxis])
        Y.append(roll)
    return np.concatenate(X), np.concatenate(Y)

def weighted_loss(y_true, y_pred):
    bce = tf.keras.losses.binary_crossentropy(y_true, y_pred)
    weights = tf.where(y_true == 1, 2.0, 1.0)
    return tf.reduce_mean(bce * weights)

def build_model(input_shape=(229,), num_notes=88):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        layers.Reshape((229, 1, 1)),
        layers.Conv2D(32, (3, 1), activation='relu',
padding='same'),
        layers.MaxPooling2D((2, 1)),
        layers.Conv2D(64, (3, 1), activation='relu',
padding='same'),
        layers.MaxPooling2D((2, 1)),
```

```
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(num_notes, activation='sigmoid')
    ])
    return model

X, Y = load_data()
model = build_model()
model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
              loss=weighted_loss,
              metrics=["accuracy"])

model.fit(X, Y, epochs=10, batch_size=128, validation_split=0.1)
model.save(CHECKPOINT_PATH)

pred = model.predict(X)
pred_bin = (pred > 0.5).astype(np.int32)
true_bin = Y.astype(np.int32)

acc = accuracy_score(true_bin.flatten(), pred_bin.flatten())
prec = precision_score(true_bin.flatten(), pred_bin.flatten(),
average='micro', zero_division=0)
rec = recall_score(true_bin.flatten(), pred_bin.flatten(),
average='micro', zero_division=0)
f1 = f1_score(true_bin.flatten(), pred_bin.flatten(),
average='micro', zero_division=0)

print(f"Accuracy: {acc:.3f}, Precision: {prec:.3f}, Recall:
{rec:.3f}, F1: {f1:.3f}")
```