МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів аналізу звукового ряду та генерація структурованого опису

Виконав студент 2 курсу, групи ІПЗм-21-4
Пономаренко А. О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми Освітньо-наукова

Керівник доцент каф.ПІ Турута О.П.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

Харків 2023

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« 10 » квітня 2023 р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Пономаренко Артура Олександровича
(прізвище, ім'я, по батькові студентів)

1. Тема роботи Дослідження методів аналізу звукового ряду та генерація структурованого опису

затверджена наказом університету від «10» квітня 2023 р. № 340Ст

2. Термін подання студентом роботи до екзаменаційної комісії «19» травня 2023 р.

3. Вихідні дані до проекту: електронні ресурси за обраною тематикою, алгоритми машинного навчання для розпізнавання іменованих сутностей, методичні вказівки до виконання кваліфікаційної роботи магістра, пояснювальна записка.

4. Перелік питань, що потрібно опрацювати в роботі: аналіз предметної області, постановка задачі, огляд існуючих методів, опис розробленої системи.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд наукової та патентної літератури	11.04.2023	виконано
2	Постановка задачі	14.04.2023	виконано
3	Розробка моделі	17.04.2023	виконано
4	Збір тестових даних	20.04.2023	виконано
5	Проектування та реалізація програмної системи	23.04.2023	виконано
6	Розробка плану проведення експериментальних досліджень	27.04.2023	виконано
7	Проведення експериментальних досліджень	01.05.2023	виконано
8	Аналіз отриманих результатів	05.05.2023	виконано
9	Підготовка пояснювальної записки	07.05.2023	виконано
10	Підготовка презентації та доповіді	10.05.2023	виконано
11	Перевірка на плагіат	12.05.2023	виконано
12	Архівування	12.05.2023	виконано
13	Нормконтроль та рецензування	12.05.2023	виконано
14	Попередній захист	18.05.2023	виконано
15	Допуск до захисту у зав. кафедри	19.05.2023	виконано

Дата видачі завдання 10 квітня 2023 р

Студент_____
(підпис)

Керівник доц. _____(Турута О.П.)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 67 с., 13 рис., 7 табл., 14 джерел.

НЕРОЙННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ АУДІО, РОЗПІЗНАВАННЯ АКОРДІВ, PYTHON, TENSORFLOW

Об’єктом дослідження є методи розпізнавання акордів за допомогою нейронних мереж.

Метою роботи є порівняння використання спектрограм і хромограм для розпізнавання акордів нейронною мережею.

Результатом роботи є набір даних у вигляді записів на гітарі акордів різних тонів і типів, програма для навчання порівняння якості розпізнавання спектрограм і хромограм, аналіз якості використання хромограм в реальних умовах.

Explanatory note contains: 67 pages, 13 illustrations, 7 tables, 14 sources.

NEURAL NETWORKS, AUDIO RECOGNITION, CHORDS RECOGNITION, PYTHON, TENSORFLOW

The object of the research is chord recognition methods using neural networks.

The aim of the study is to compare the use of spectrograms and chromagrams for chord recognition by a neural network.

The outcome of the work is a dataset consisting of recordings of chords played on a guitar with different tones and types, a program for training and comparing the quality of spectrogram and chromagram recognition, and an analysis of the quality of using chromagrams in real-world conditions.

Умови публікації пояснювальної записки

Я, Пономаренко Артур Олександрович студент групи ІПЗМ-21-4 здобувач вищої освіти на другому (магістерському) рівні, кафедра програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження методів аналізу звукового ряду та генерація структурованого опису», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання. Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ

Акорд – ритмічно одночасне сполучення кількох (не менше трьох) різних за висотою звуків. В класичній гармонії акордом вважається лише таке сполучення звуків, в якому звуки розташовані по терціях.

Аплікатура (акорду) – спосіб розташування й порядок чергування пальців при грі на музичному інструменті, а також позначення цього способу в нотах.

Гама – частина звукоряду від тоніки до тоніки наступної октави, де ступені ладу розташовані послідовно один за одним у висхідному або низхідному порядку.

Зведення або мікшування (від англ. mixing - змішування) - стадія створення з окремих записаних треків кінцевого запису, наступний після звукозапису етап створення фонограми, що полягає у відборі та редагуванні (іноді реставрації) вихідних записаних треків, об'єднанні їх в єдиний проект.

Каподастр (італ. Capo – головка, верх; та італ. tasto – лад; дослівно: верхній поріжок) – затиск, що використовується в струнно-щипкових інструментах (гітара, балалайка, мандоліна), для висотної транспозиції шляхом штучного укорочування звучної (робочої) частини струн. [1]

Секунда (лат. secundus – другий) – музичний інтервал між двома сусідніми звуками в гамі, також друга нота в гамі.

Септакорд – чотиризвучний акорд, в основному виді якого звуки розташовані за терціями

Спектрограмма (сонограма) - зображення, що показує залежність спектральної щільності потужності сигналу від часу. Спектрограми застосовуються для ідентифікації мовлення, аналізу звуків тварин, у різних галузях музики, радіо- та гідролокації, обробці мовлення, сейсмології та інших областях.

Сус-акорд – акорд, в якому друга нота замінена на кварту або на секунду.

Темп (італ. tempo від лат. tempus – час) – міра швидкості в музиці. Частота чергування долей і тактів, визначається смисловим змістом в музиці; Ступінь швидкості виконання як правило вказується на початку музичного твору.

Терція – музичний інтервал в три ступені, також третя нота в гамі.

Тональність – звуковисотне положення ладу, у ширшому сенсі – ієрархічна система висотних зв'язків, центральною категорією якої є тоніка.

Транспонування – перенесення музичного твору з однієї тональності в іншу тональність того ж нахилу без усякої зміни [2]

ВСТУП

Дана робота представляє огляд методів аналізу музичної доріжки у вигляді електронного файлу та генерації акордів для заданого файлу; пошук шляхів покращення існуючих алгоритмів для кінцевого користувача. Також ця робота є продовженням дослідження студента Ярового М. В. під назвою “Дослідження методів розпізнавання акордів для навчання гри на музичному інструменті”, в якій автор розглянув методи розпізнавання стандартних акордів, і в якості продовження теми, запропонував дослідження розпізнавання сус-акордів. Тому в даній роботі буде розглянуто можливості розпізнаваня таких, а також септакордів, та інших деяких різновидів акордів.

Музична транскрипція – це процес написання нотних записів виключно на основі запису музики. Щоб розшифрувати музичний твір вручну, транскрипціонер повинен володіти знаннями нотної грамоти та володіти знаннями в аналізі музичного звуку, щоб розшифрувати мелодію. Однак цей процес стає виснажливим і трудомістким, оскільки транскрипціонеру доводиться слухати музику багато разів і записувати ноти одну за одною.

Программи, що надають такий інструментарій вже існують. Наприклад: Chord AI, Chord Tracker, Chordify, MyChord та інші. Але, на жаль, комерційні продукти не мають відкритого коду, тим не менш вони будуть розглянуті для аналізу наявного функціоналу і його ефективності. Також існують деякі відкриті бібліотеки, такі як: autochord [3] та Chord-recognition[4]. Їх можливості також буде розглянуто.

Основою цієї роботи є розробка настільної програми із функцією автоматичної транскрипція акордів зіграних в аудіофайлі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Напрямок дослідження

Завдяки широкому доступу до цифрової музики через Інтернет, стало неможливим вручну обробити цю величезну кількість даних; навіть організація особистої колекції музичних є не тривіальною задачею. Тому інструменти автоматизації є важливими. Music information research (MIR) є міждисциплінарною наукою, чиєю метою є розширення інформаційного пошуку в нетекстових областях. Мета MIR полягає в описі декількох аспектів, що пов'язані з вмістом музики. Деякі застосування MIR включають музичний транскрипт, класифікацію музики [5], генерацію плейлистів [6] та розпізнавання музики [7].

Пошук музичної інформації (MIR) полягає в пошуку та організації великих колекцій музики або музичної інформації, щоб задовольнити певні запити. Це актуально, оскільки багато музики і послуг, пов'язаних з нею, доступно у цифровому форматі. В останні роки було проведено безліч конференцій, симпозіумів та семінарів по цій темі, що зібрали багато дослідників з різних галузей. Окрім того, численні компанії і бібліотеки додають свою підтримку для MIR, через високу привабливість для комерційної діяльності.

Традиційно музику анотують текстовою інформацією, яку надає обкладинка. Ця текстова інформація достатньо для автоматичного характеризування текстів, але не придатна для опису вмісту музики. Звісно, підхід, який базується на тексті, не має гнучкості. Крім того, дослідники також цікавляться розширенням аудіо інформаційного пошуку за допомогою більш людськості природного взаємодії, наприклад, похмурення або пісні ритму.

Наступний крок буде полягати в вирішенні задач з використанням нових інтерфейсів для прослуховування музики, пошуку і рекомендацій. Академічна привабливість цього напрямку досліджень обумовлена масовим невирішеним проблемами розуміння складних музичних звукових сигналів, що передають вміст, та створення тимчасових структур. Існують також кілька привабливих проблем, які ще не були зачеплені, і це поле є дослідницькою темою. У цій роботі

будуть розглядатися деякі важливі проблеми, які можуть у подальшому збільшити привабливість та соціальний вплив наукових досліджень з музики в майбутньому.

Розпізнавання акордів в музиці отримало багато уваги за останні роки, оскільки це може бути корисним для використання в музичних дослідженнях та для полегшення процесу навчання. Завдяки розумним системам та алгоритмам, ми можемо використати програмне забезпечення для автоматичного розпізнавання акордів в музиці. Це дослідження буде проводитися для запропонування нових методів та алгоритмів розпізнавання акордів в музиці, які будуть покращувати роботу програмних засобів та гарантувати більш правильні результати.

Це дослідження також буде присвячене порівнянню різних методів аналізу музики, а також наданню порад та рекомендацій для подальшого використання програмних засобів розпізнавання акордів в музиці. Наша мета – додати до існуючих методів та алгоритмів, використовуваних для розпізнавання акордів в музиці, нові ідеї та просувати їх практичне застосування.

1.2 Використання AI в розпізнаванні звуків

В магістерській роботі на тему “Дослідження методів розпізнавання акордів для навчання гри на музичному інструменті” Яровий Микита проаналізував чотири методи розпізнавання акордів:

- алгоритм на відповідність шаблону;
- алгоритм з використанням ПММ;
- алгоритм з використанням нейронної мережі над хромаграмою;
- покращений алгоритм з використанням нейронної мережі.

Згідно з його дослідження методи, що використовують нейронні мережі, показали значно точніші результати розпізнавання. Тому в цій роботі будуть розглянуті саме вони та можливі шляхи покращення.

Розпізнавання звуків за допомогою штучного інтелекту (AI) – це процес виявлення та інтерпретації аудіосигналів, з метою ідентифікації різних звуків або класифікації їх за типом.

Для розпізнавання звуків за допомогою AI, використовуються алгоритми машинного навчання, які навчаються розпізнавати звуки на основі набору тренувальних даних. Зазвичай, для цього використовуються моделі глибокого навчання, такі як рекурентні нейронні мережі (RNN) або сверточні нейронні мережі (CNN), які здатні автоматично виявляти та аналізувати особливості аудіосигналів.

Процес розпізнавання звуків може включати такі етапи, як аналіз спектрограми звуку, виділення акустичних характеристик, які відрізняють один звук від іншого, та подальшу класифікацію звуку на основі цих характеристик. Наприклад, модель може навчитися розпізнавати різні голоси, шуми в середовищі або музичні інструменти.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ АНАЛІЗУ ЗВУКОВОГО РЯДУ

В рамках даного підрозділу буде переглянуто існуючі рішення. Будуть розглянуті як і готові програми, які представляють цінність для кінцевих користувачів, так і окремі бібліотеки та фреймворки.

2.1 Огляд користувацьких програм

Програми, оглянуті в даному розділі взяті зі статті “The 7 Best Chord Identifier Websites and Apps” [8].

Усі наведені програми мають функціонал розпізнавання акордів. Розпізнаються також темп, розмір, та тональність мелодії.

2.1.1 Chord ai

Chord ai – це мобільний застосунок для обох мобільних платформ: iOS та Android. Але також є можливість запустити програму на компьютерах Mac, що оснащені процесорами від Apple лінійки M.

Є чотири шляхи провадження аудіодоріжки:

- завантажити з пам’яті пристрою, обравши з файлового провідника;
- обрати аудіо зі стрімінгового сервісу Apple Music;
- обрати аудіотрек з платформи SoundCloud;
- обрати відео із треком з платформи YouTube.

Після завантаження файлу виконується аналіз треку, і одразу можна перейти до акордів пісні. Акорди супроводжуються аплікатурою на гітарі клавішах, або укулеле (платна функція). Інтерфейс програми наведено на рисунку 2.1.

Програма також розпізнає тональність, темп та музичний розмір пісні. Перші два можна змінити, але ця функція доступна лише в платній версії. Пісня розділена на музичні такти. Є можливість транспонувати акорди в інші тональності. Для тренування можна зациклити песний проміжок пісні.

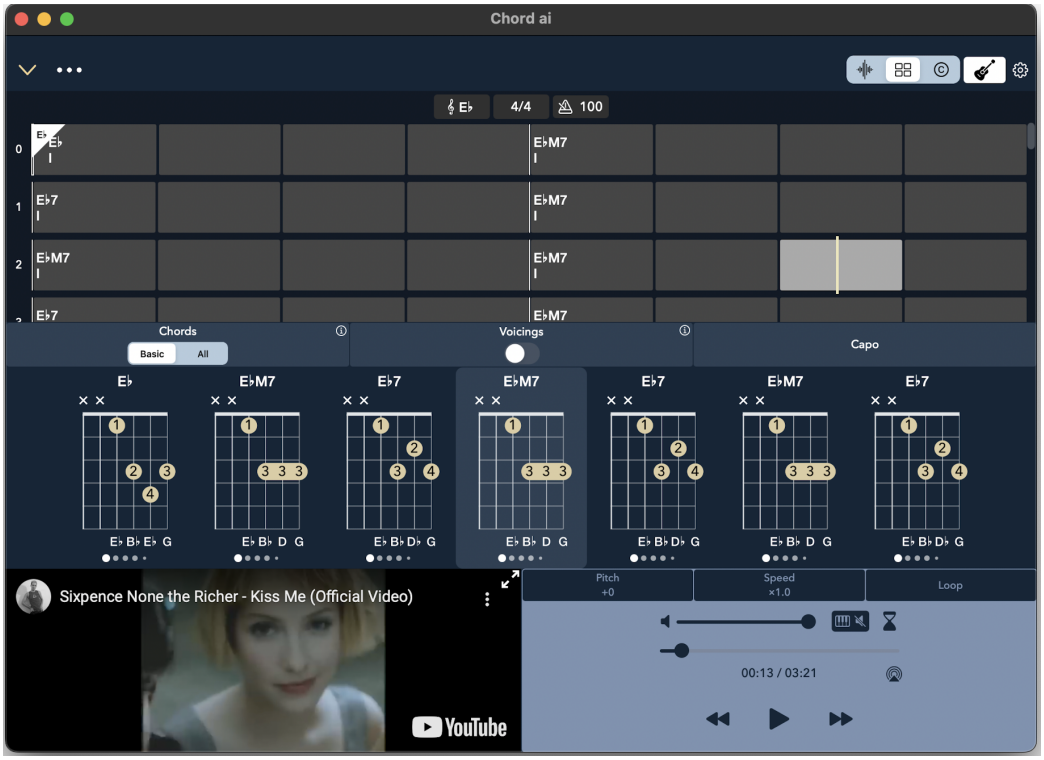

Рисунок 2.1 – Інтерфейс програми Chord ai

Відтворення аудіо, яке обране за допомогою пошуку з YouTube, виконується прямо у вбудованому медіа-плеєрі YouTube.

2.1.2 Chord Tracker

Програма недоступна для завантаження в Україні, тому функцінал можливо оцінити тільки із заявленого виробником (Yamaha Corporation of America) описом.

Програма Yamaha Chord Tracker для iPhone/iPod touch/iPad допомагає тренуватися та виконувати пісні, аналізуючи аудіопісні, збережені на пристрої iOS, а потім відображаючи символи акордів.

Chord Tracker аналізує аудіопісні, збережені в музичній бібліотеці пристрою iOS, а потім відображає акорди пісні на екрані. Також застосунок, подібно до попереднього застосунку Chord ai, відображає аплікатури акордів для гітари та піаніно [9].

На відміну від Chord ai, програма не дозволяє завантажувати аудіо зі стримінгових сервісів. Інтерфейс порграми наведено на рисунку 2.2.

Рисунок 2.2 – Знімки екрану застосунку Chord Tracker надані в App Store

2.1.3 Chordify

Chordify являє собою веб-застосунок. Після реєстрації пропонується обрати один з чотирьох інструментів: гітара, укулеле, піаніно або інший. У безкоштовній версії програми розпізнавання акордів працює лише для пісень із наявної бібліотеки. Він доступний за посиланням: https://chordify.net/.

Програма має багатий функціонал: зациклювання заданого проміжку пісні для навчання, зміна темпу, адаптація акордів для гри із каподастром, транспонування, завантаження MIDI-файлу, але весь цей функціонал доступний лише у преміум-версії, яка не має тріального періоду. Інтерфейс програми наведено на рисунку 2.3.

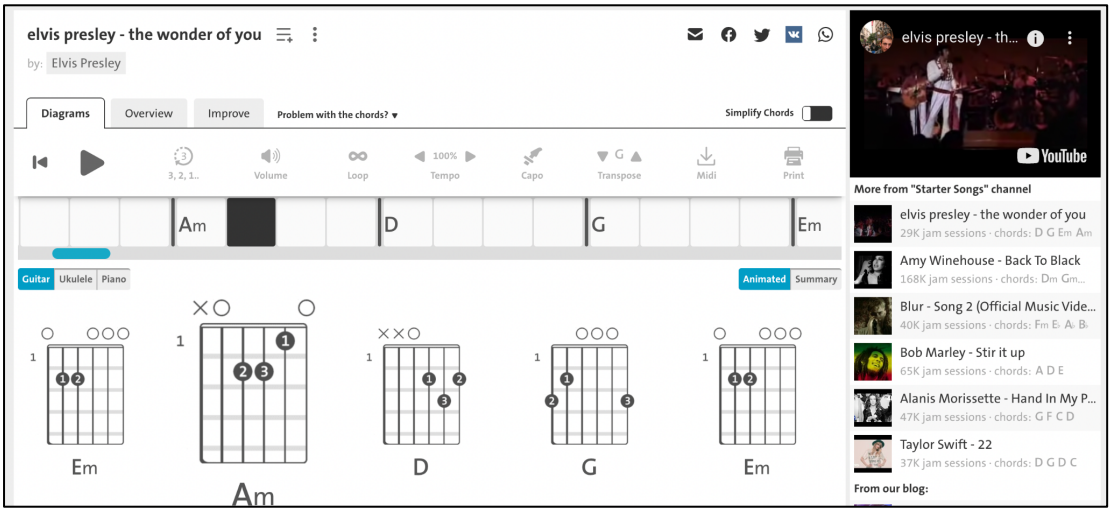

Рисунок 2.3 – Інтерфейс Веб-застосунку Chordify

2.1.4 MyChord

MyChord – це мобільний застосунок для Android і iOS. Він дозволяє або завантажити музичні файли з медіатеки телефону, або виконати пошук в YouTube. Функціонал програми порівняно з попередніми значно бідніший. Після обробки файлу відображаються акорди пісні. На відміну від інших програм, тут часова пряма не поділена на такти, а лише має вказівки часу в секундах. Акорди супроводжуються аплікатурою одного з обраних інструментів: гітари, укулеле, банджо, мандоліни. Інтерфейс програми наведено на рисунку 2.4.

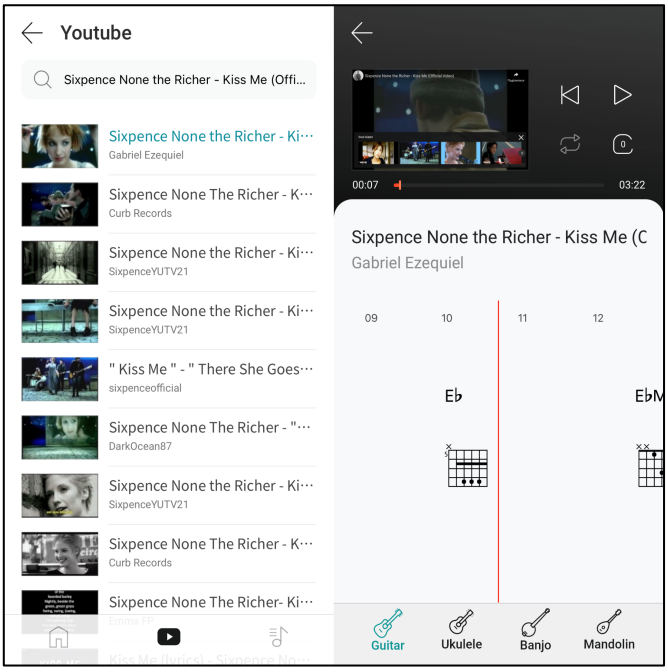

Рисунок 2.4 – Інтерфейс програми MyChord

2.2 Порівняння застосунків

В таблиці 2.1 наведено порівняння сервісів згаданих вище.

Таблиця 2.1 – Порівняння сервісів

Назва застосунку	Аналіз власних файлів	Розділення часового проміжку на такти	Зображення аплукатур	Зміна тональності
Chord ai	+	+	+	+ (тільки преміум акаунт)

Кінець таблиці 2.1

Назва застосунку	Аналіз власних файлів	Розділення часового проміжку на такти	Зображення аплукатур	Зміна тональності
Chord Tracker	+	+	+	–
Chordify	+ (тільки преміум акаунт)	+	+	+
MyChord	+	-	+	-

2.3 Огляд фреймворків та бібліотек

2.3.1 autochord

Python модуль Autochord є однією бібліотекою для розпізнавання акордів в аудіофайлі. Він використовує аналіз акордів для визначення складу акордів в аудіофайлі. Відповідно до описання фреймворк заснований на роботі TensorFlow. TensorFlow – це популярна бібліотека глибокого навчання, яка була розроблена і призначена для наукових дослідників, програмістів-розробників та аналітиків. Вона покликана полегшити та прискорити розробку моделей глибокого навчання для автоматизації аналітичних завдань.

Основною функцією модуля autochord є recognize. Під капотом autochord.recognize() запускає плагін NNLS-Chroma VAMP для видобування хроматичних особливостей з аудіо і подає їх до моделі Bi-LSTM-CRF в TensorFlow для розпізнавання акордів. В даний час модель може розпізнати 25 класів акордів: 12 основних мажорних тріад, 12 мінорних тріад і значення “немає акорда” ('N').

2.3.2 librosa

Наступний фреймворк librosa – це пакет Python для аналізу музики та аудіо. Він забезпечує будівельні блоки, необхідні для створення систем пошуку музичної інформації. [10]

На сайті з документацією зазначено наступні сабмодулі пакету:

- librosa.beat – функції для оцінки темпу та виявлення подій ударів.
- librosa.core – основні функції включають функції завантаження аудіо з диска, обчислення різноманітних представлень спектрограм і різноманітні часто використовувані інструменти для аналізу музики. Для зручності всі функції цього підмодуля доступні безпосередньо з простору імен librosa.* верхнього рівня.
- librosa.decompose – функції для гармонічно-перкусійного поділу джерел (HPSS) та розкладання загальної спектрограми з використанням методів матричної декомпозиції, реалізованих у scikit-learn.
- librosa.display – підпрограми візуалізації та відображення за допомогою matplotlib.
- librosa.effects – обробка звуку у часовій області, наприклад зсув висоти та розтягнення часу. Цей підмодуль також забезпечує обгортки часової області для підмодуля декомпозиції.
- librosa.feature – вилучення та маніпулювання характеристиками. Сюди входить виділення низькорівневих ознак, таких як хромаграми, спектрограма Mel, MFCC та різні інші спектральні й ритмічні характеристики. Також надаються методи маніпулювання функціями, такі як дельта-функції та вбудовування пам’яті.
- librosa.filters – генерація банків фільтрів (хроматичність, псевдо-CQT, CQT тощо). Це в першу чергу внутрішні функції, які використовуються іншими частинами librosa.
- librosa.onset – виявлення початку та розрахунок сили початку.
- librosa.segment – функції, корисні для структурної сегментації, такі як побудова матриці повторювань, подання із затримкою та послідовно обмежена кластеризація.
- librosa.sequence – функції для послідовного моделювання. Різні форми декодування Вітербі та допоміжні функції для побудови матриць переходів.

– librosa.util – Допоміжні утиліти (нормалізація, заповнення, центрування тощо)

За допомогою цієї утиліти ми можем робити цілу низку задач. Зокрема ми можемо отримати спектрограму доріжки (рис. 2.4) для подальшого її аналізу та визначення нот.

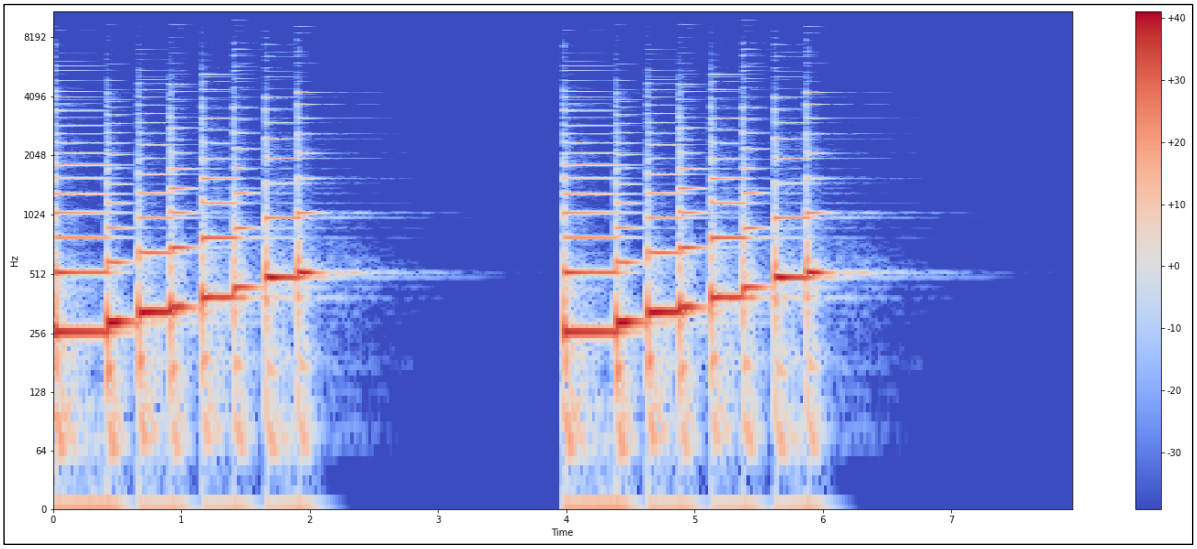

Рисунок 2.4 – приклад спектрограми аудіодоріжки

3 ЗБІР І ПІДГОТОВКА ДАНИХ

Метою практики є збір та підготовка даних. В ході цієї роботи буде виконано запис акордів на гітарі. Акорди мають бути різних тонів і різних видів – це і будуть класи, по яким нейромережа розділятиме акорди. Прикладами акордів є Cm (до мінор), Amaj7 (ля великий мажорний акорд), та інші. На гітарі один і той самий акорд можна зіграти в різних позиціях, саме тому для збільшення обсягу матеріалу для навчання нейронної мережі запис виконано в різних позиціях. Але для нейронної мережі один і той самий акорд записаний в різних позиціях вважається за один і той самий клас.

Після запису аудіофайли будуть перетворені на зображення, а саме спектрограму та хромаграму для подальшого використання для навчання моделі нейронної мережі. Детальніше про це описано в підрозділах нижче.

3.1 Загальний опис збору і підготовки даних

Збір і підготовка даних є одним з найважливіших етапів наукового дослідження, оскільки якість результатів дослідження залежить від якості даних, на основі яких вони були отримані.

Перший етап – це визначення потрібних даних для дослідження і їх збір. Це може включати опитування, експерименти, аналіз статистичних даних, збір даних з веб-сайтів, соціальних мереж тощо.

Після збору даних необхідно провести їхню попередню обробку, яка може включати видалення непотрібної інформації, визначення пропущених значень, очищення даних від помилок, аномалій та інших недоліків.

Далі можна провести аналіз даних, використовуючи різні методи: статистичний аналіз, машинне навчання, аналіз тексту, обробка зображень тощо.

Після аналізу даних можна провести їхню фінальну обробку і підготовку для використання в науковому дослідженні. Це може включати агрегацію даних, створення нових змінних, побудову графіків, таблиць, діаграм тощо.

Дані необхідно представити в такому вигляді, який найліпше підходить для обробки компьютером. В даній задачі, завдяку сучасним гаджетам, збір даних

можливо виконати в домашніх умовах за відносно невеликий час. Сирі дані в цьому дослідженні уявляють з себе записи акордів зіграних на акустичній гітарі. Акорди мають бути зіграні по-різному як лівою, так і правою рукою для різноманіття навчання моделі. Після того записані файли мають бути розділені на окремі короткі фрагменти, де зіграний лише один акорд та погруповані як по тональності, так і по виду акорду.

Перед тим, як навчати нейронну мережу типам аудіо, аудіофайли необхідно перетворити на зображення, які можуть бути використані для подальшого навчання. Це зроблено для того, щоб зберегти інформацію про часову та частотну структуру аудіофайлу, яка може бути використана для навчання нейронної мережі.

В даному випадку, для графічного представлення зображення було обрано саме спектрограму та хромаграму, бо вони відображають інтенсивність частот в аудіофайлі, що важливо для визначення акорду.

3.2 Опис запису і даних

Збір даних для досліджень, пов'язаних із музикою, може здійснюватися різними способами, залежно від конкретної задачі та даних, які необхідні для дослідження.

В даній задачі збір даних являє собою запис акордів зіграних на гітарі. Гітара була обрана, як один з найпопулярніших інструментів, а також, як доступний варіант для запису.

Існує декілька різних способів запису гітари, кожен з яких має свої переваги та недоліки. Ось декілька найпоширеніших способів:

- мікрофонний запис: цей спосіб полягає в тому, що мікрофон розміщується перед гітарним усилителем, який в свою чергу підключений до гітари. Мікрофон записує звук, який відбивається від усилителя. Цей метод зазвичай дає досить природний звук гітари, але може бути чутливим до шумів та інших перешкод;

- директ-запис: цей метод полягає в записі безпосередньо з гітарного процесора або звукової карти. Він може бути зручним, оскільки не вимагає мікрофона та може бути збережений без якихось додаткових обробок. Проте, зазвичай цей метод дає менш природний звук, і він може звучати більш сухо та безжиттєво;
- запис зі змішуванням: цей метод використовує обидва попередні способи, щоб забезпечити більш повний та природний звук. Він полягає в тому, що зазвичай один мікрофон розміщується перед усилителем, а інший записує звук гітарного процесора. Ці записи потім змішуються разом, щоб створити більш повний та насичений звук;
- запис з використанням п'єзо-датчиків: п'єзо-датчики розміщуються під струнами гітари, тому звук записується безпосередньо зі струн. Цей метод може бути корисним для акустичної гітари або гітари без усилителя.

Запис акордів був виконаний на мобільний телефон iPhone 12 mini за допомогою стандартного застосунку “Voice Memos” (або просто “Диктофон”) в кімнаті. Акустична гітара Yamaha LL6 ARE. Для зручності запис виконувався по різних тонам і видам акордів. Тобто на кожному аудіофайлі зіграний акорд певної тональності і певного виду і різних позиціях лівої руки по грифу, і з різною інтенсивністю правою руки – повільно і різко. На початку файлу записувачем називається акорд, наприклад “До мажор септакорд”. Потім в найнижчій позиції акорд грається три рази із різною інтенсивністю. Між акордами робиться пауза, щоб потім розділити файл на декілька частин. Після того акорд грається в іншій позиції, зазвичай вгору по грифу, аж до приблизно 12-ого ладу, до куди фізично можливо грати акорди на гітарі.

Один і той самий акорд можна зіграти на гітарі в різних позиціях. Це означає, що ті ж самі ноти можуть бути зіграні на гітарі в різних місцях на грифі. Кожна позиція має свої переваги та недоліки, і музиканти можуть вибирати позицію в залежності від того, який звук вони хочуть отримати.

В рамках даної задачі аккорд зіграний на різних позиціях класифікується однаково. Тобто наприклад, можна зіграти акорд До мажор у відкритій позиції, а можна із баре на 3-ому ладу. І ми розглядатимемо ці два акорди, як однакові. Бо нас турбує класифікація по тону і по виду, а не по позиції.

3.3 Попередня обробка записів

Обробка аудіо – важлива частина дослідження, пов'язаного з аналізом музики. Щоб зробити аудіо зручним для подальшого аналізу, спочатку його потрібно імпортувати в програмне забезпечення для аналізу аудіо. Далі можна застосувати різні фільтри, щоб покращити якість звуку та зменшити шуми.

По-перше, стандартний формат запису файлу через диктофон вказаного в попередньому підрозділі телефону – m4a. Для зручності обробки всі файли були переведені в mp3. Так як більшість утиліт, що будуть використані далі сумісні саме з цим одним з найпопулярніших і найуживаніших форматів.

Формат mp3 і m4a (також відомий як AAC) є двома з найбільш поширених аудіоформатів. Однією з головних переваг формату mp3 є те, що він є стандартом для стиснення аудіофайлів вже більше 20 років. Тому більшість пристроїв та програм підтримують формат mp3. Крім того, за допомогою формату mp3 можна значно скоротити розмір файлу без втрати якості звуку.

Загальна тривалість записаного матеріалу – 42 хвилини 25 секунд. Середня довжина запису одного виду і тону акорду триває 1 хвилину. Ці треки необхідно розділити, щоб в кожному файлі залишився один окремо зіграний акорд. Щоб зекономити час, процес було автоматизовано за допомогою скрипта написаного на python. Для цього було використано бібліотеку pydub.

Pydub - це бібліотека для роботи з аудіофайлами в Python. Один з головних класів, що входять до складу Pydub, – це AudioSegment. Він дозволяє завантажувати, зберігати, редагувати та конвертувати аудіофайли в різних форматах.

Однією з корисних функцій AudioSegment є silence. Вона дозволяє знайти місця в аудіофайлі, де звуковий рівень є менше за задане значення, і повертає

список об'єктів AudioSegment, що містять тишу. Наприклад, це можна використовувати для автоматичного вирізання пауз або тиші в аудіофайлах.

```
audio_file = AudioSegment.from_mp3(file_name)
# Split audio by silent
non_silence_regions = silence.split_on_silence(
    audio_file, min_silence_len=min_silence_len,
    silence_thresh=silence_thresh
)
# Skip the very first region,
# because this is the region where I say the chord name
non_silence_regions.pop(0)
# Export regions as files
for i, silence_region in enumerate(non_silence_regions):
    pure_file_name = file_name.replace('.mp3', '')
    region_file_name = "%s_%i.mp3" % (pure_file_name, i)
    silence_region.export(region_file_name, format="mp3")
```

Вище наведено фрагмент скрипта. Програма розділяє аудіо по тиші. Найперший фрагмент видаляється, тому ще це уривок, де під час запису називається акорд. Далі новостворені фрагменти експортуються в нові файли. Надалі ці файли необхідно підчистити.

Очищення даних - це важливий етап в підготовці матеріалів для навчання нейронної мережі. Очищення даних включає в себе обрізання та попередню обробку даних, що дозволяє забезпечити якість та точність навчання нейронної мережі.

Один з перших кроків – це видалення непотрібних даних, таких як дубльовані записи, неправильно позначені елементи або даних, що не відповідають тематиці дослідження. Далі потрібно виконати попередню обробку даних, таку як нормалізація даних, конвертування відформатованих даних у зрозумілий для моделі формат, виконання фільтрації шуму та видалення артефактів.

У випадку з аудіо- та відеоданими, може бути корисно виконати відсікання тиші, фільтрацію шуму та видалення артефактів, таких як піскування, скрипіння або інші аномалії в звуку.

Важливо також розуміти, що очищення даних має великий вплив на результат навчання моделі, тому цей процес потребує виваженості та уважності. В

результаті коректної підготовки даних можна отримати більш точну та надійну модель нейронної мережі, яка буде давати більш якісні прогнози на нових даних.

Оскільки в силу людського фактору неможливо зробити ідеальний запис, то отримані дані треба очистити від сміття. Під час запису часто між тими моментами, як грається акорд, звучить усілякий шум – голос, шорох, стук чи будь-що інше схоже. Тому коли скрипт розділив суцільний трек на окремі фрагменти, то в деяких з них замість зіграного акорду звучить некорисний шум. Від таких “сміттєвих” даних треба позбутися. І це було виконано вручну, бо кількість записаного матеріалу це дозволяє виконати за прийнятний проміжок часу. Тобто усі записані дані було прослухано і відфільтровано – фрагменти із сміттєвим шумом було видалено.

3.4 Доповнення даних

Штучне доповнення даних – це процес додавання штучно створених або змінених даних до набору даних, які використовуються для навчання нейромережі. Це може бути корисно для поліпшення точності моделі та зменшення перенавчання.

Існує кілька підходів до штучного доповнення даних:

- розширення даних: цей підхід полягає в збільшенні кількості даних за рахунок створення нових прикладів, що багато в чому схожі на вже існуючі. Наприклад, зображення можуть бути збільшені, зменшені, повернені, зміщені, або змінені якимось іншим чином. Цей підхід є особливо ефективним у випадках, коли набір даних невеликий;
- додавання шуму: інший спосіб доповнення даних полягає в додаванні різного роду шумів до існуючих даних. Наприклад, можна додати гаусівський шум до зображень або додати випадкові забруднення до аудіо-файлів. Цей підхід допомагає навчити модель більш стійкою до шуму та зменшити перенавчання;
- синтез нових даних: цей підхід полягає в створенні нових даних, які відповідають реальним даним. Наприклад, можна синтезувати

зображення або аудіо-файли, які були б схожими на реальні дані. Цей підхід є найбільш складним, але він може бути корисним у випадках, коли набір даних недостатній для навчання моделі.

Штучне доповнення даних є важливою складовою процесу навчання нейромережі, тому що воно дозволяє моделі більш точно вивчати закономірності.

У випадку з даною задачею доповнення даних було виконано для створення нових класів, а конкретно акордів із дієзними (бемольними) тональностями. Це було зроблено для збереження ресурсів під час збору даних, тобто для економії часу під час запису акордів.

Тобто акорди були записані для таких тонів: до, ре, мі, фа, соль, ля, сі (C, D, E, F, G, A, B). Для тонів до дієз, ре дієх, фа дієз, соль дієз, ля дієз (C#, D#, F#, G#, A#) аудіофайли були згенеровані на основі перших. Зміна тону полягає у зміні такого параметру аудіо, як частота (rate). Наприклад, якщо нота ля четвертої октави звучить (коливається) на частоті 440 Гц, то A# – 466.16 Гц – це сталі значення.

Для того, щоб збільшити тональність звуку на півтону скористаємось формулою 3.1:

$$r_1 = r_0 * 2^{n/N} \tag{3.1}$$

де r_1 – нова частота,

r_0 – частота вхідного файлу,

n – кількість напівтонів на яку змінюється частота (додатне значення підвищує частоту, від'ємне – зменшує),

N – стала кількість нот в октаві, дорівнює 12-ти.

Аудіофайли було підвищено на пів тону за допомогою скрипту написаному на Python. Був використаний вже згаданий класу AudioSegment з утіліти Pydub. Ось так виглядає фрагмент коду цього скрипта:

```
// 1
sound = AudioSegment.from_mp3(file_name)
```

```
    // 2
    new_sample_rate = int(sound.frame_rate * (2 ** (1 / 12)))
    // 3
    hipitch_sound = sound._spawn(sound.raw_data, overrides={'frame_rate':
new_sample_rate})
    hipitch_sound = hipitch_sound.set_frame_rate(sound.frame_rate)
    // 4
    hipitch_sound.export("%s" % (output_destination), format="mp3")
```

Алгоритм наступний:

– завантажуємо аудіофайл;

– вираховуємо нову частостоту;

– редагуємо частоту файлу;

– експортуємо новий файл.

3.5 Перетворення аудіо в графічну інформацію

Перетворення аудіо-даних в графічну інформацію може бути корисним для навчання нейромережі для завдань обробки сигналів, таких як розпізнавання мови, розпізнавання звуку, розрізнення музичних інструментів та інших. Одним з можливих підходів для перетворення аудіо в графіку є використання спектрограм.

Спектрограма – це графік залежності частоти від часу зображення звуку. Щоб перетворити аудіо в спектрограму, потрібно спочатку розділити аудіо на невеликі фрагменти, названі фреймами, і застосувати дискретне перетворення Фур'є (ДПФ) [11] до кожного з цих фреймів. ДПФ перетворює звуковий сигнал з часової області в частотну область. Після цього отримані значення можуть бути візуалізовані на спектрограмі, де кожен стовпчик відповідає часовому фрейму, а висота стовпчика відображає інтенсивність частотного складу відповідного фрейму.

Хромаграма – це спектрограма, але сигнал поділений на 12 логарифмічно розподілений частин, де кожна з них відповідає одному з 12-и тонів в октаві. В результаті маємо дванадцяти-вимірний вектор, який представляє розподілення тонів в сигналі.

З кожного з аудіофайлів було створено спектрограму та хромаграму за допомогою скрипта написаного на Python. Основні бібліотеки використані у скрипті – librosa, np, matplotlib. Нижче наведено код для створення хромаграми:

```
input_file = sys.argv[1]
output_file = sys.argv[2]
# Load audio file
y, sr = librosa.load(input_file)
# Compute STFT
D = np.abs(librosa.stft(y))
# Map frequencies to chromatic scale
chroma = librosa.feature.chroma_stft(S=D, sr=sr)
# Plot chromagram
plt.figure(figsize=(2.5, 1))
librosa.display.specshow(chroma)
plt.tight_layout(pad=0)
plt.savefig(output_file)
```

І ще один подібний скрипт, але в цей раз для створення спектрограми:

```
input_file = sys.argv[1]
output_file = sys.argv[2]
# Load audio file
samples, sample_rate = librosa.load(input_file)
spectrogram = librosa.stft(samples)
spectrogram_db = librosa.amplitude_to_db(abs(spectrogram))
# Save image
plt.figure(figsize=(3, 3))
librosa.display.specshow(spectrogram_db, sr=sample_rate)
plt.tight_layout(pad=0)
plt.savefig(output_file)
```

На рисунку N наведено приклад хромаграми для акорду ля мажор. По вертикальній осі можна чітко побачити, що рисунок поділений на 12 частин, що відповідають 12 нотам в октаві. Горизонтальна вісь представляє часову пряму. Яскравість кольору відображає інтенсивність звуку на конкретній частоті в конкретний проміжок час. Чим яскравіший колір – тим інтенсивніше звук, чим темніше – тим менш інтенсивний. На рисунках 3.1 та 3.2 також зображена хромаграма аудіо, на якому звучить акорд ля мажор, але зіграний на гітарі в іншій позиції. Можна побачити, що ноти мають трохи іншу інтенсивність, але водночас, найінтенсивніші “лінії” майже однакові.

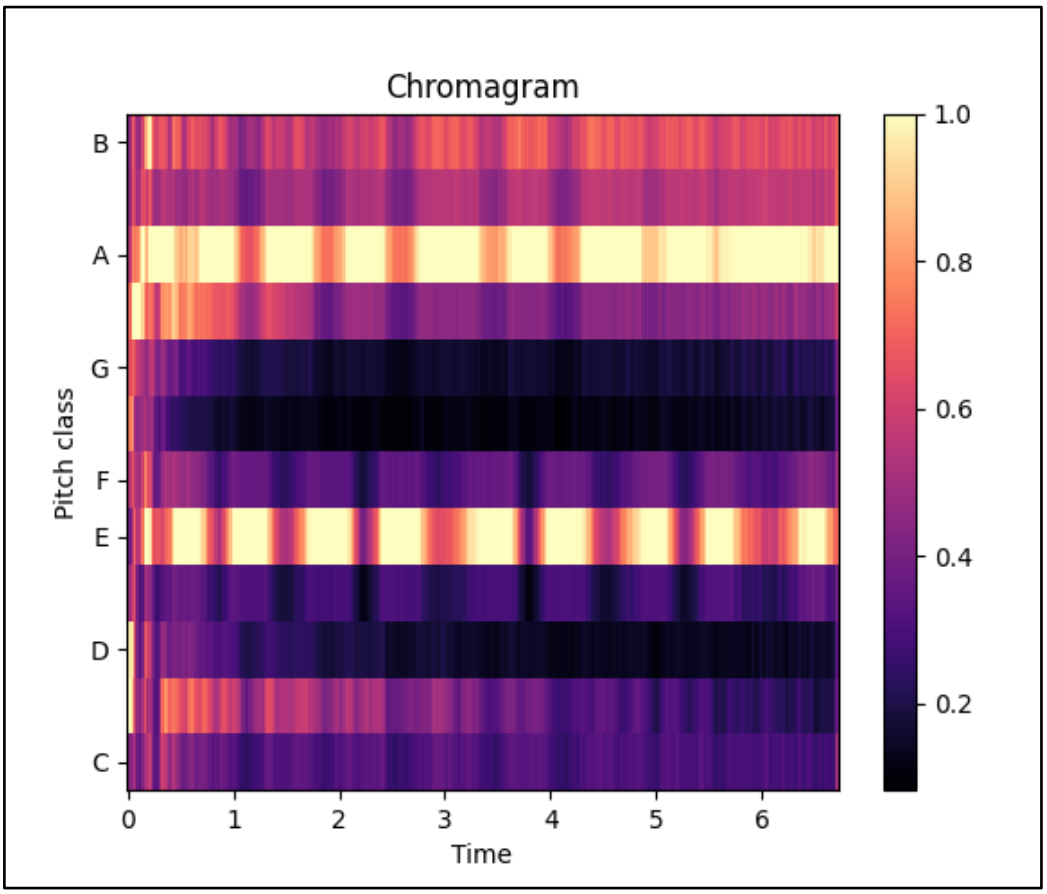

Рисунок 3.1 – Хромаграма акорду ля мажор (1)

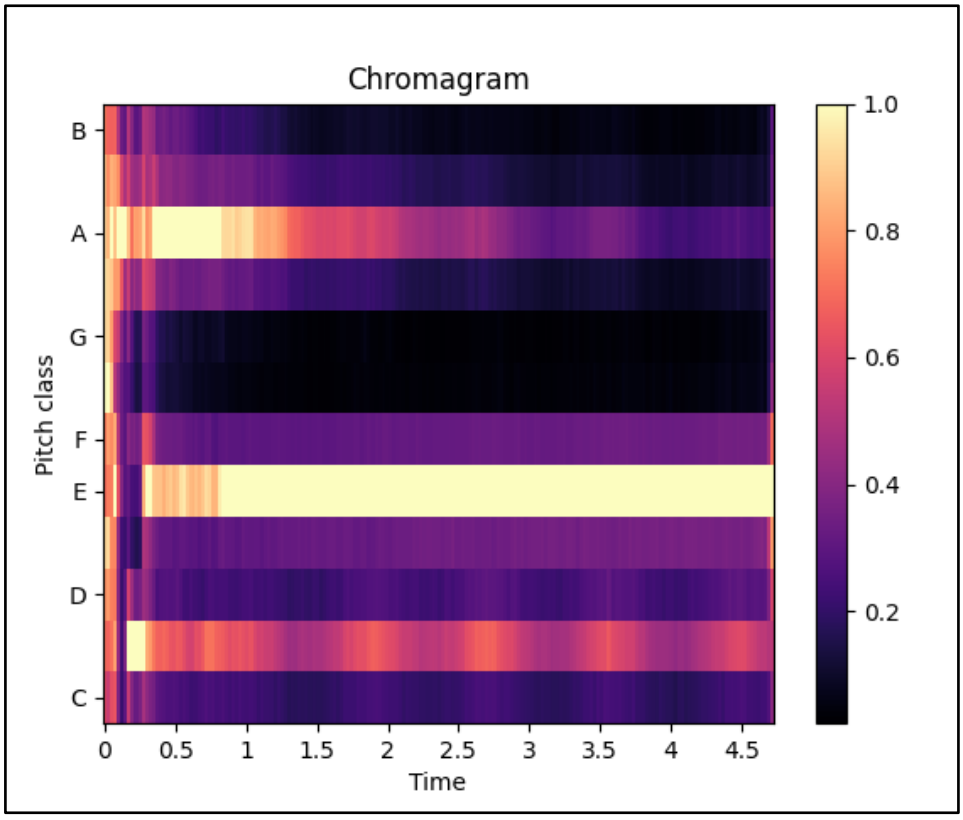

Рисунок 3.2 – Хромаграма акорду ля мажор (2)

Для порівняння з іншим акордом, звернемо увагу на рисунок 3.3, на якому наведено приклад іншого акорду, а саме сі мажор. На спектрограмі можна чітко побачити, що домінуть інші “лінії”, бо звучать інші ноти.

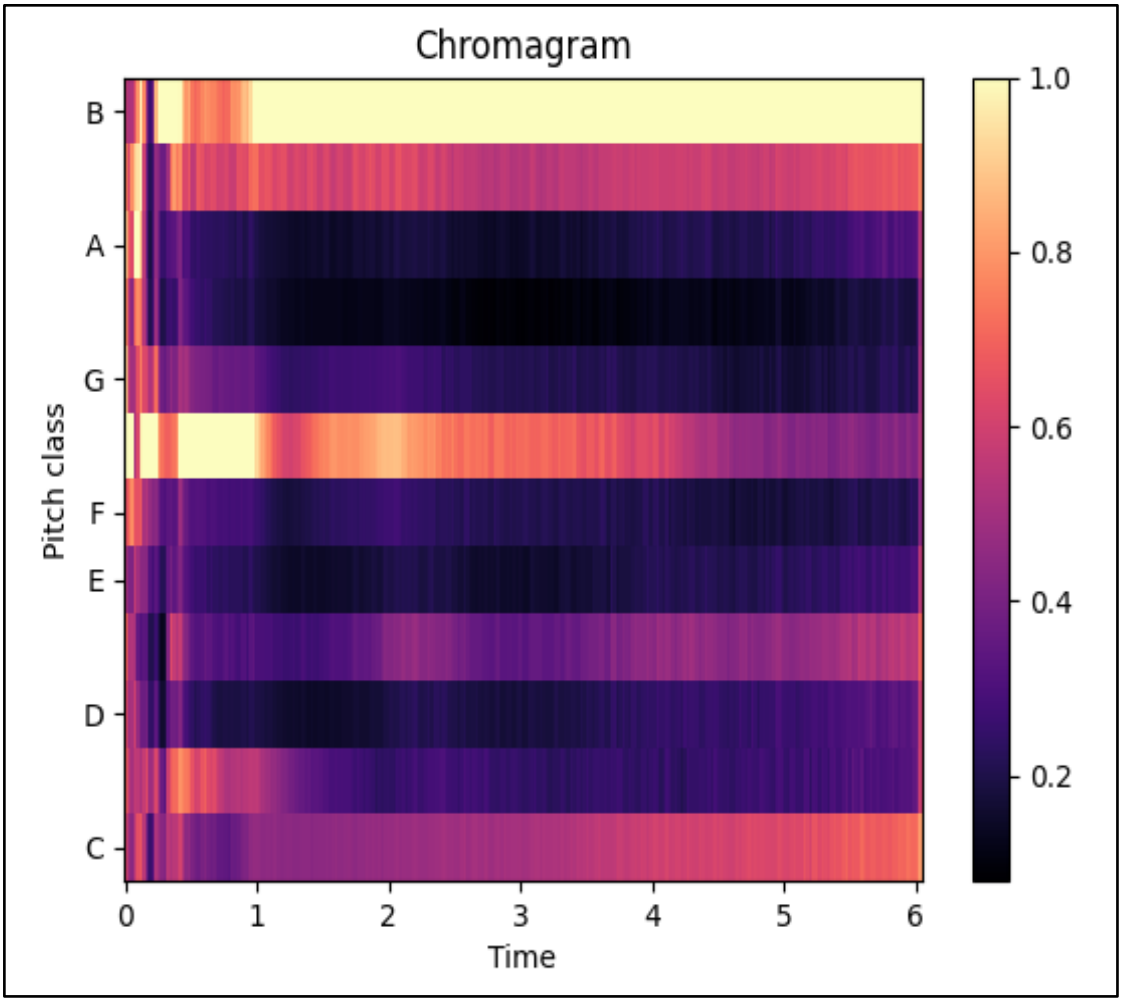

Рисунок 3.3 – Хромаграма акорду сі мажор

Таким чином ми ще раз переконалися, що хромаграма годиться для розпізнавання акорду. Адже однакові акорди мають спільні риси, в той час коли різні акорди дещо відрізняються.

Приклад спектрограми наведено на рисунку 3.4. Це спектрограма аудіо, на якому звучить акорд ля мажор. Як можна побачити по вертикальній осі вказується частота, по горизонтальній – час, колір визначає інтенсивність звуку. Спектрограма містить значно більше деталей, аніж хромаграма, адже показує не тільки 12 тонів, а частоту по всій шкалі.

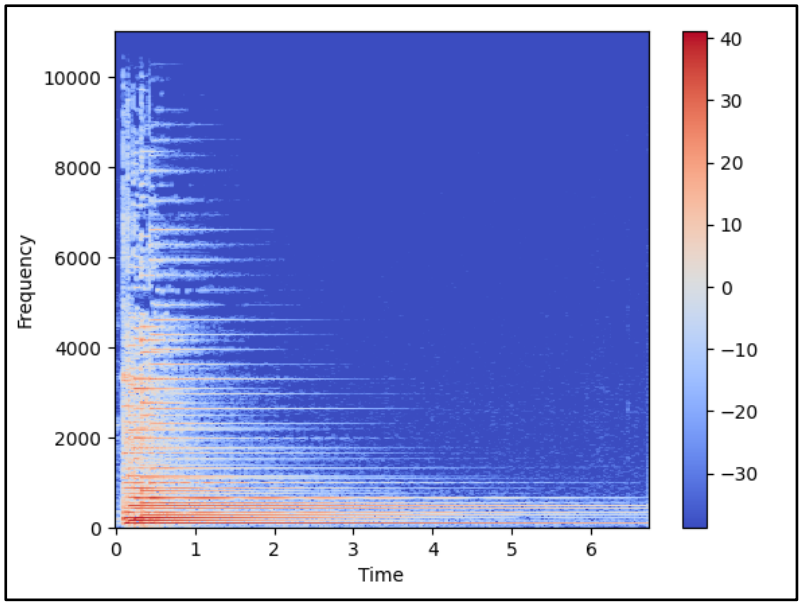

Рисунок 3.4 – Спектрограма акорду ля мажор

Ймовірно значне збільшення кількості деталей негативно впливатиме на якість навчання моделі, і як результат, розпізнавання інших акордів. Тому припускаємо, що хромаграма покаже себе значно краще для поставленої задачі.

4 ПОСТАНОВКА ЗАДАЧІ

В свої роботі “Дослідження методів розпізнавання акордів для навчання гри на музичному інструменті” студент Яровий М. В. дослідив різні способи розпізнавання акордів [12]:

– алгоритм на відповідність шаблону,

– алгоритм з використанням ПММ,

– алгоритм з використанням нейронної мережі над хромаграмою.

За результатами цього дослідження автор виявив, що найефективнішим алгоритмом є той, що використовує машинне навчання, тобто останній. Студент Яровий перетворював аудіо записи на хромаграму і використовував ці зображення для навчання моделі. Штучний інтелект використовується в великій кількості галузей [13]. І в тому числі його слушно використати і для цієї задачі.

Хромаграма є підвидом спектрограми. Спектрограма є графічним зображенням спектрального складу звукового сигналу. Спектрограма показує, які частоти присутні в звуці та яка їхня інтенсивність в залежності від часу. Вона складається з горизонтальних смуг, які представляють різні частоти, і вертикальних стовпців, які відображають інтенсивність звуку в цих частотах на певний момент часу.

Хромаграма в свою чергу – це спектрограма, але сигнал поділений на 12 логарифмічно розподілених частин, де кожна з них відповідає одному з 12-и тонів в октаві. В результаті маємо дванадцяти-вимірний вектор, який представляє розподілення тонів в сигналі.

Задачу яку я ставлю перед собою в межах цієї роботи – це порівняти ефективність використання хромаграми та спектрограми в процесі розпізнавання акордів.

5 РЕАЛІЗАЦІЯ ПРОГРАМИ

Для виконання поставленої задачі треба було обрати найзручніші інструменти реалізації, щоб виконати роботу було швидко і якісно. Тому для створення програми було обрано мову python і фреймоворк tensorflow.Python є однією з найбільш поширених мов програмування для машинного навчання. Python має велику кількість бібліотек та інструментів, що роблять його дуже популярним серед дослідників та практиків машинного навчання. TensorFlow - це відкрите програмне забезпечення для машинного навчання, розроблене компанією Google. Воно надає інструменти для розробки та навчання нейронних мереж, а також для виконання різноманітних обчислювальних завдань. TensorFlow має високорівневе API, що дозволяє зосередитись не на детялях реалізації, а на самій задачі.

Перш за все необхідно підготувати датасети. Зазвичай з виборки частина даних береться для навчання, інша частина для валідації. В даному випадку співвідношення було обрано 0.8/0.2. Тобто 80% для навчання і 20% – валідація. Нижче наведено код створення датасетів:

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset='training',
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size
)
```

Далі проводимо стандартизацію даних:

```
normalization_layer = tf.keras.layers.Rescaling(1./255)
normalized_ds = train_ds.map(
                 lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
```

```
first_image = image_batch[0]
```

Тепер зображення чорно-білі. Кешуємо датасети:

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
num_classes = len(class_names)
```

Створюємо базову Керас модель:

```
model = Sequential([
  layers.Rescaling(1./255,
                   input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses
                .SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

І нарешті, тренуємо модель. Кількість епох – 10.

```
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

Далі, показуємо результати навчання на графіках:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
```

```
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

І завершальна частина – це безпосередньо тестування навченної моделі.

```
image_path = "/Users/arturponomarenko/Desktop/G.png"
img = tf.keras.utils.load_img(
    image_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

plt.imshow(img)
plt.show()

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent
confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

6 ОПИС НАУКОВОГО МЕТОДУ ТА МАТЕМАТИЧНОЙ МОДЕЛІ

6.1 Науковий метод

У даній роботі використовуються теоретичний та емпіричний методи дослідження. Теоретичний метод передбачає аналіз наявних теоретичних відомостей та вхідних даних, необхідних для експерименту та моделювання. Емпіричний метод заснований на проведенні ряду експериментів. Ці методи тісно пов'язані між собою: результати експерименту використовуються для формулювання гіпотези, яку можна підтвердити чи спростувати наступними експериментами. Якщо результати експериментів постійно повторюються, можна говорити про наявність залежності.

У процесі моделювання програмної системи постійно виникає необхідність оновлювати або створювати нову модель. Під час тренування нейронних мереж дослідник намагається досягти високої точності результатів на тестових даних. Якщо ціль досягнута, дослідник може перейти до використання моделі для експерименту на реальних даних, які можуть містити зайві або випадкові дані. Система, заснована на нейронних мережах, повинна розпізнавати зайві дані та зменшувати їх вплив на результати вимірювань.

6.2 Математична модель для оцінки алгоритмів

Для того, щоб ефективно досліджувати та порівнювати алгоритми, необхідно мати математичну модель. Ця модель має дозволяти відобразити всі характеристики та можливості алгоритму. Математична модель є важливим інструментом дослідження, оскільки вона дозволяє розрахувати та проаналізувати результати роботи алгоритмів, які ми досліджуємо, і отримати результати, на основі яких можна зробити певні висновки.

В даній роботі для аналізу якості розпізнавання буде використано два підходи. По-перше, якість навчання моделі. Ці дані можна отримати вже на етапі навчання. Оскільки під час навчання проводиться валідація, то маємо на виході точність розпізнавання і точність валідації для кожної з епох навчання.

Для інших експериментів, де вже цих даних буде недостатньо будуть рахуватися кількість правильно розпізнаних акордів відносно всієї кількості.

$$Y = \frac{n}{N} * 100\% \tag{6.1}$$

Згідно з формулою вище Y – точність розпізнавання, n – кількість правильно розпізнаних акордів, N – загальна кількість акордів, що розпізнавалися.

7 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

7.1 Експеримент зі спектрограмою

Перший експеримент полягає в тому, щоб використати для навчання нейромережі звичайну спектрограму замість хромаграми, і порівняти різницю якості навчання моделі.

З початку для кожного записаного акорду було створено відповідне зображення спектрограми і хромаграми. Всі ці зображення погруповані в директоріях по назві акорду. Саме таке розташування файлів необхідно для фреймворку TensorFlow, щоб протаврувати зображення відповідними класами. На рисунку 7.1 зображена структура папок, в яких містяться відповідні зображення.

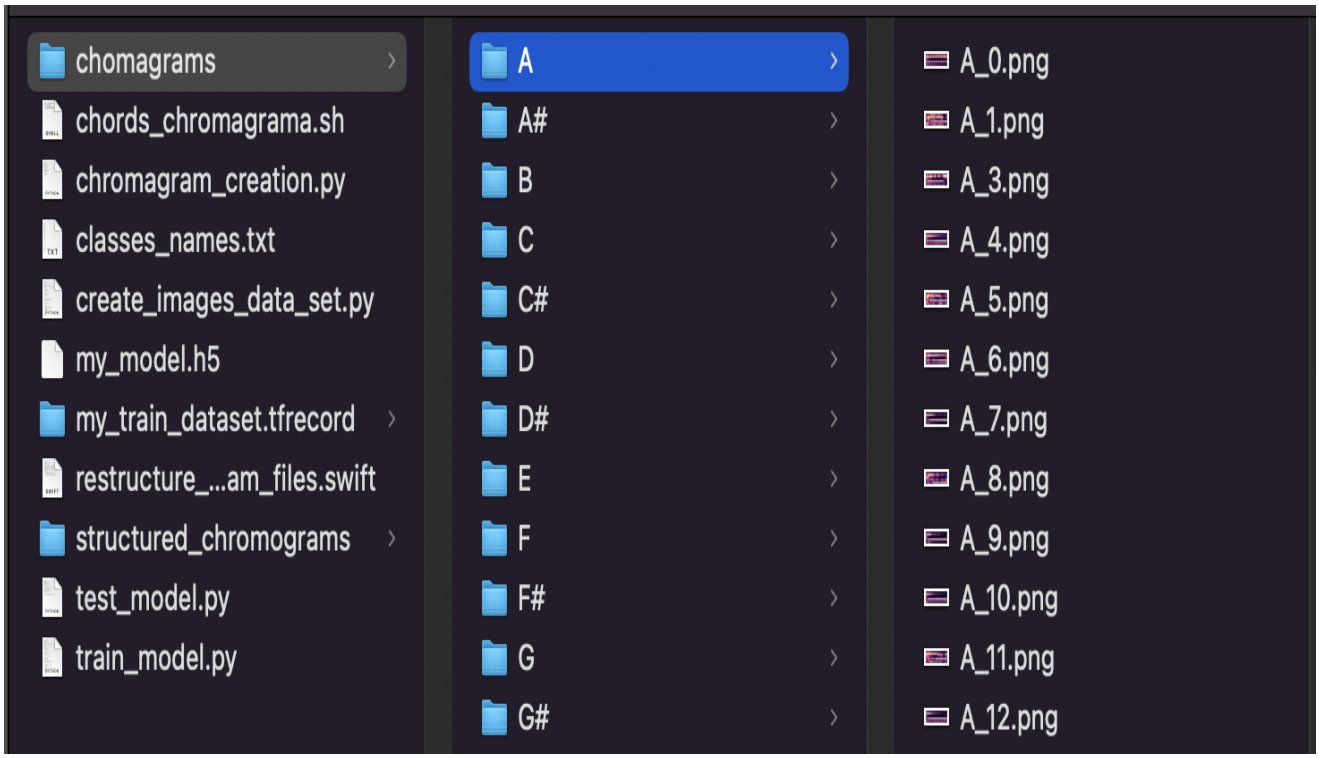

Рисунок 7.1 – Директорія із хромаграмами акордів

Перші результати ми вже можемо побачити на етапі навчання моделі. Так як частина даних використовується для валідації, то на графіку ми зможемо побачити у відсотках з якою точністю модель розпізнає акорди. На рисунку 7.2 зображено результати навчання моделі по зображенням хромаграми протягом 10 епох.

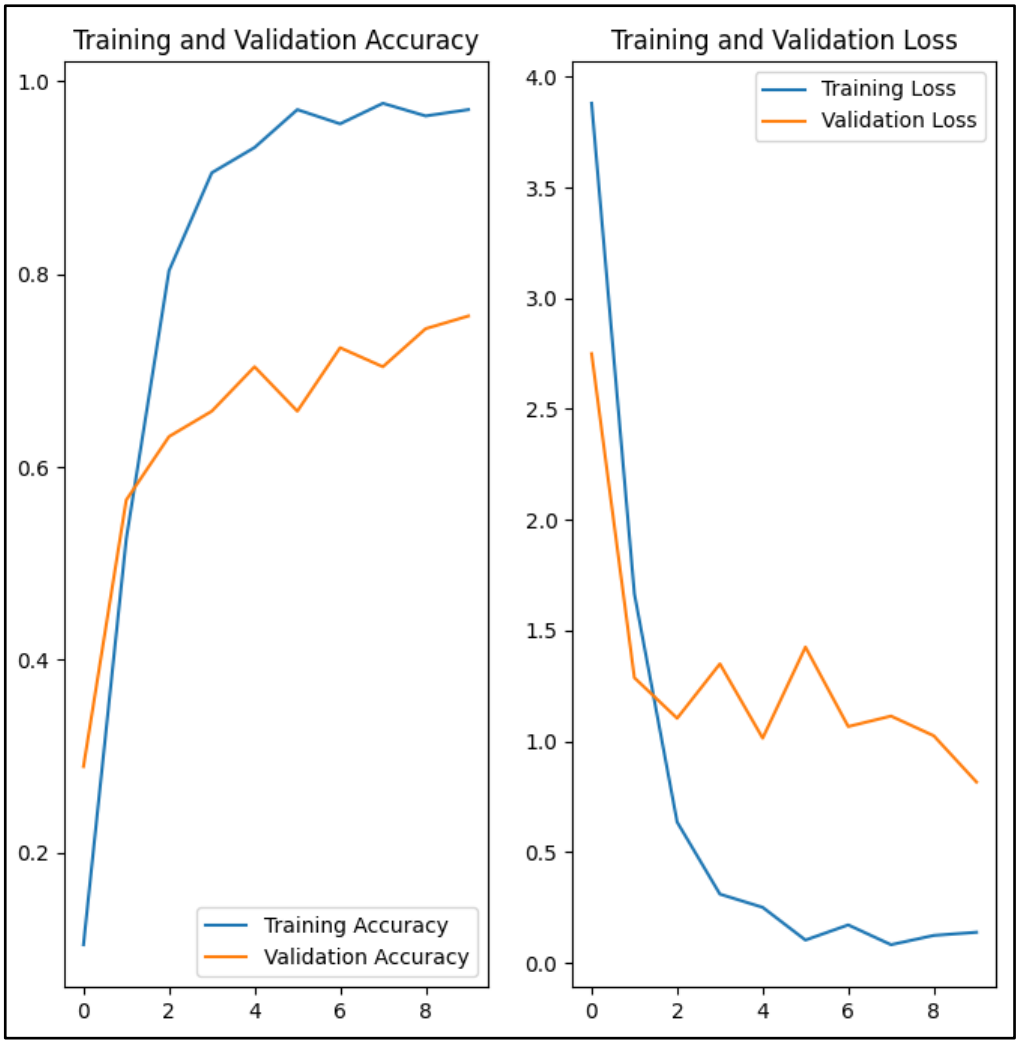

Рисунок 7.2 – Результати навчання моделі на зображеннях хромаграм за 10 епох

Також відповідні дані в детальному вигляді наведені нижче в таблиці 7.1. Як ми можемо побачити, на останній епосі точність вже складає 97%, що є достатньо високим показником.

Таблиця 7.1 – Результати навчання моделі на зображеннях хромаграм за 10 епох

Epoch	**Loss**	**Accuracy**	**Validation loss**	**Validation accuracy**
1	3.8798	0.1047	2.7497	0.2895
2	1.6665	0.5254	1.2871	0.5658
3	0.6355	0.8036	1.1039	0.6316

Кінець таблиці 7.1

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
4	0.3102	0.9051	1.3497	0.6579
5	0.2505	0.9313	1.0142	0.7039
6	0.1029	0.9705	1.4254	0.6579
7	0.1712	0.9558	1.0667	0.7237
8	0.0819	0.9771	1.1139	0.7039
9	0.1238	0.964	1.025	0.7434
10	0.138	0.9705	0.816	0.7566

Проведемо відповідне дослідження для зображень спектрограм. Нижче в таблиці 7.2 наведено результати навчання моделі по зображеннях спектрограм.

Таблиця 7.2 – Результати навчання моделі на зображеннях спектрограм за 10 епох

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
1	4.8372	0.0131	4.3099	0.0197
2	4.2809	0.0087	4.2796	0.0098
3	4.2736	0.0175	4.2824	0.0033
4	4.2732	0.024	4.2878	0.0066
5	4.2675	0.0218	4.3096	0.0066
6	4.2646	0.0218	4.3151	0.0066
7	4.2582	0.0175	4.3435	0.0066
8	4.2539	0.0218	4.3551	0.0066
9	4.2515	0.0153	4.3623	0.0066
10	4.2498	0.0153	4.3673	0.0066

Як ми можемо побачити точність сягає максимум 2%, що надзвичайно низько порівняно із моделею, що була навчена на зображеннях хромаграми. Нижче на рисунку 7.3 наведено ці ж дані, але у вигляді графіку. На ньому можемо

побачити, що на відміну від першого випадку тут не простежується тендеція, графік доволі хаотичний.

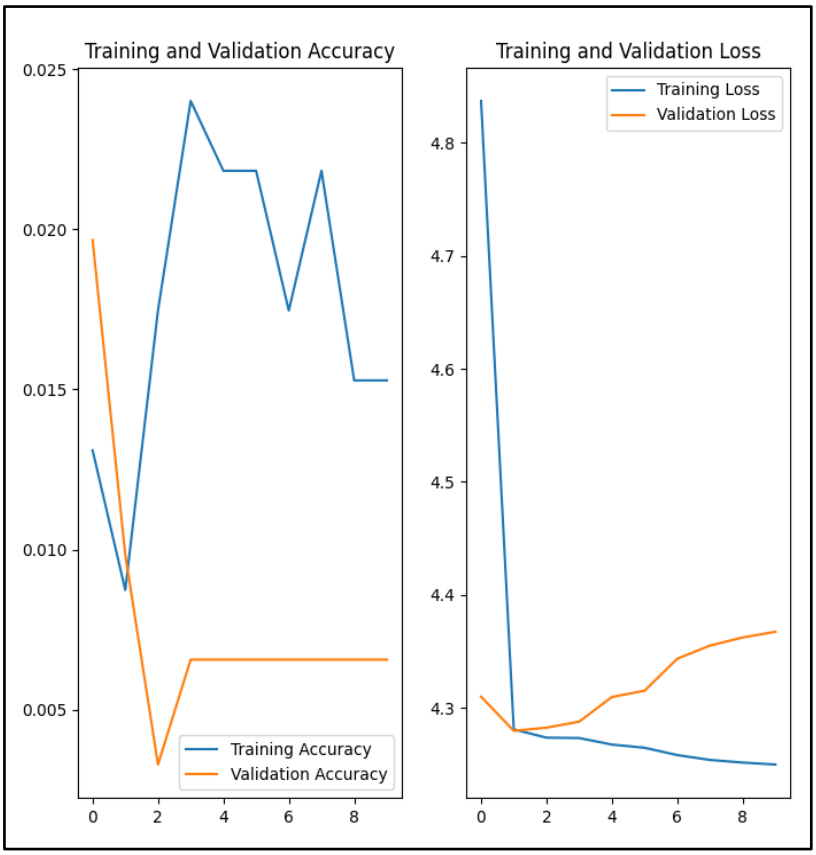

Рисунок 7.3 – Результати навчання моделі на зображеннях спектрограм за 10 епох

З цього експерименту можемо зробити висновок, що навчання моделі на зображеннях спектрограми дає набагато кращий результат – 97% проти 2%. Пояснити такий результат можна тим, що спектрограми мають значно більше інформації, аніж хромаграми – на них зображений весь спектр частот, в той час коли на хромаграмах графік зображує тільки самі ноти. Навіть якщо дві однакові ноти зіграні в різних октавах для хромаграми вона всеодно визнається як одна і та сама. Оскільки задачею є розпізнавання акорду не зважаючи на октаву, то хромаграма для цього ідеально підходить.

Іншими словами можна сказати, що зображення хромаграми втрачає інформацію щодо октав, але разом з тим акумулює інформацію з різних октав, що таким чином дозволяє отримати зображення з більш концентрованою інформацією. На рисунку 7.4 зображено для порівняння хромаграма (зліва) і

спектрограма (зправа) для одного і того самого аудіозапису, на якому звучить акорд ля мажор.

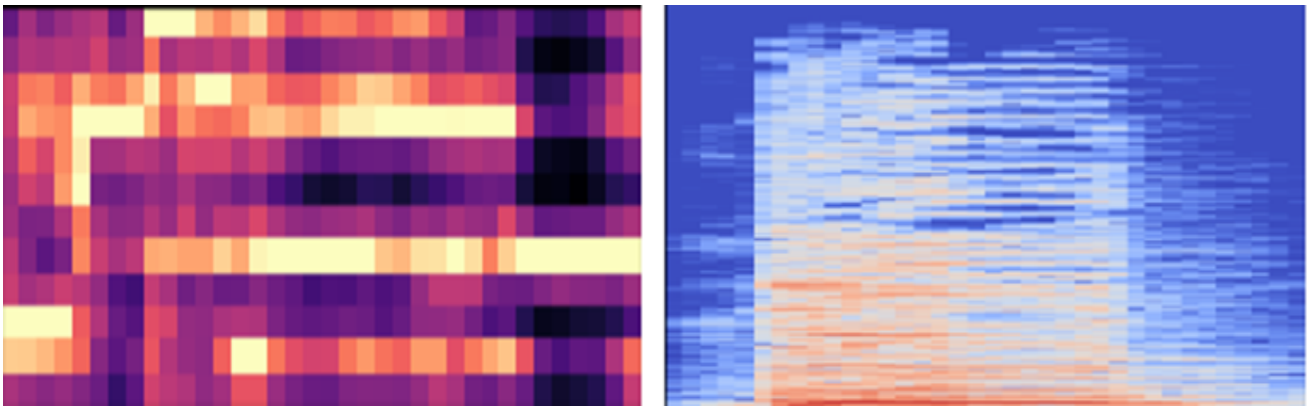

Рисунок 7.4 – Хромаграма (зліва) та спектрограма (справа) акорд ля мажор

Відповідно можна зробити висновок, що за рахунок того, що хромаграма містить менше інформації, це дозволяє нейронній мережі навчити модель більш якісно.

7.2 Експеримент з акордами не з виборки

Оскільки вже було визначено, що зображення хромаграми дозволяють навчити модель нейронної мережі більш якісно, то пропоную для подальших експериментів користуватися саме ними. Мета наступного експерименту – подивитись яким чином поведе себе модель, якщо для розпізнавання їй дати акорд, який вона не знає. Оскільки різні акорди можуть між собою перетинатися по вмісту нот, то я пропоную аналізувати саме цей показник. Наприклад акорд до мажор складається з нот: до, мі, соль, а акорд мі мінор – мі, сі, соль. Відповідно ці два акорди мають спільні дві ноти: соль і мі. Тому на це треба зважати. Якщо хоча б пару нот збігаються, то розпізнавання можна назвати успішним.

Для експерименту були зіграні 4 акорди: C6add9 (до, мі, ля, ре), D7add9 (ре, фа дієз, до, мі), G#m5+6(соль дієз, фа, сі, мі) , G7add13 (соль, фа, сі, мі). Маємо такі результати (див. табл. 7.3).

Таблиця 7.3 – Розпізнавання акордів

	Оригінальний акорд	Ноти оригінального акорду	Акорд розпізнаний програмою	Ноти акорду розпізнаного програмою	Впевненість розпізнання
1	C6add9	до, мі, ля, ре	Dm7	ре, фа, ля, до	86.53%
2	D7add9	ре, фа дієз, до, мі	D7	ре, фа дієз, ля, до	79.01%
3	G#m5+6	соль дієз, фа, сі, мі	E	мі, соль дієз, сі	56.39%
4	G7add13	соль, фа, сі, мі	E	мі, соль дієз, сі	51.34%

Як ми можемо побачити в першому випадку в акорді C6add9 (до мажор з 6 і 9 ступенями) програмою було розпізнано Dm7 (ре мінор септ) із впевненістю 86.53%. При цьому у них збігається аж 3 з 4 нот, тому результат можна визнати цілком позитивним.

У другому випадку в акорді D7add9 (ре мажор септ з 9 ступенем) програма розпізнала D7 (ре мажор септ) із впевненістю 79.01%. В цих двох акордах також збігається 3 ноти, тому вердикт аналогічний.

У третьому випадку в акорді G#m5+6 (соль дієз мінор з підвищеним 5 ступенем і з 6 ступенем) програма розпізнала E (мі мажор) із впевненістю 56.39%. В цьому випадку також збігається 3 ноти.

Програма працює очікувано. Але для кращого користувацького досвіду, я вважаю, треба показувати окремі ноти, що присутні в акорді. Тобто якщо впевненість розпізнавання нижче заданої, то програма розпізнає не цілий акорд, а ноти, і на основі цих нот робить припущення, що це за акорд.

В даному експерименті ми побачили, що найбільша впевненість складає 89%, тобто поріг для такого функціоналу можна вказати в 90%, або навіть трохи більше.

7.3 Експеримент з неналаштованою гітарою

В реальному житті зовсім рідко можна зустріти ідеальні умови запису. Тим паче, коли мова йде про гітару, то одним з факторів, що може суттєво вплинути на розпізнавання – є налаштованість гітари. Є два варіанти як може бути розлаштована гітара. Перший – коли струни налаштовані одна відносно іншої, але жодна з них не налаштована відносно камертона, і другий варіант, коли струни розлаштовані хаотично. Для цих двох випадків проведемо експеримент, і подивимось як програма впорається із розпізнаванням.

7.3.1 Експеримент 1 з неналаштованою гітарою

В даному випадку всі струни будуть налаштовані, але не абсолютно, а відносно одна одної. Для збору даних знадобиться записувати гітару і перед кожним записом проводити переналаштування. Це забирає чимало часу, і тим паче є ризик порвати струни під час запису, адже часте перетягування струн призводить до їхнього зношування. Для того, щоб мінімізувати ресурси на запис, я пропоную замість ручного підлаштування я пропоную зробити запис один раз на правильно налаштованій гітарі, і потім програмним шляхом зменшити і збільшити висоту тону, точно так, як було зроблено під час збору записів для навчання моделі для отримання дієзних (бемольниз) акордів.

Записи має сенс зменшувати/збільшувати до чверті тону, адже більша різниця робить акорди вже ближчим до іншого сусіднього акорду. Різниця між сусідніми нотами складає півтону. В даному експерименті за пів тону вважається 1. Тобто +1 – це нота вища на пів тону, а -1 – відповідно нижче на півтону. +0.5 – означає підвищення тону на чверть тону, 0 – означає оригінальний запис.

Було проведено запис 7 акордів: Am7, B7, Cmaj7, Dm7, Em7, Fmaj7, G7. Після того, кожний із записів було зменшено і збільшено від -0.6 до +0.6 півтони. Нижче в таблиці вказано результати розпізнавання для кожного з записів. На початку рядка вказано зміну тону. Далі на перетині зміни тону і назви акорду вказано два значення – це власне акорд, що було розпізнано програмою (в

кращому випадку він збігається з оригінальним), і впевненість розпізнавання. Результати експерименту наведені в таблиці 7.4.

Таблиця 7.4 – Результати експерименту 1

Зміна в напівтонах	**Am7**	**B7**	**Cmaj7**	**Dm7**	**Em7**	**Fmaj7**	**G7**
-0.6	B	A#7	Bmaj7	Dm7	D#m7	Fmaj7	F#7
	89.10	91.69	69.58	99.67	99.91	99.04	99.98
-0.5	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	89.62	49.86	99.97	99.84	99.73	99.01	92.96
-0.4	Am7	B	Cmaj7	Dm7	Em7	Fmaj7	G7
	91.43	52.12	99.98	99.83	99.97	99.43	91.31
-0.3	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	94.16	51.41	99.97	99.88	99.63	99.61	95.17
-0.2	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	94.96	53.11	99.98	99.89	99.83	99.58	94.99
-0.1	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	93.63	55.31	99.98	99.88	99.92	99.63	92.37
0	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	92.09	61.08	99.98	99.85	99.88	99.54	96.88
+0.1	Am7	B7	Cmaj7	Dm7	Em7	Fmaj7	G7
	86.59	53.65	99.98	99.90	99.89	99.13	96.99
+0.2	Am7	B	Cmaj7	Dm7	Em7	Fmaj7	G7
	89.11	54.44	99.96	99.68	99.95	99.05	93.96

Кінець таблиці 7.4

Зміна в напівтонах	**Am7**	**B7**	**Cmaj7**	**Dm7**	**Em7**	**Fmaj7**	**G7**
+0.3	Am7	B	Cmaj7	Dm7	Em7	Fmaj7	G7
	92.97	60.76	99.97	99.87	99.69	99.42	95.26
+0.4	Am7	C7	Cmaj7	D#m7	Em7	Fmaj7	G7
	94.80	99.28	99.98	99.33	99.76	99.67	96.75
+0.5	A#m7	C7	C#maj7	D#m7	Fm7	F#maj7	G#7
	99.97	99.58	98.69	100.00	84.99	98.90	98.40
+0.6	A#m7	C7	C#maj7	D#m7	Fm7	F#maj7	G#7
	99.97	99.54	97.23	99.99	87.84	98.94	97.10

Спочатку звернімо увагу на рядок зі значенням зміни тону 0. Всі акорди розпізнані абсолютно вірно, і всі, за виключенням B7, мають впевненість розпізнавання більше 90%, навіть всі, за виключенням B7 і Cmaj7 – більше 95%. Тобто 7/7.

Акорди, тональність яких була знижена від -0.1 до -0.5, всі за виключенням одного розпізнані абсолютно вірно. Більше детально кількість розпізнаних акордів можна подивитися на таблиці 7.5.

Таблиця 7.5 – Результати розпізнавання

Зміна в напівтонах	**Кількість правильно розпізнаних акордів цілком**	**Кількість правильно розпізнаних акордів по першій ноті**
-0.6	2/7	2/7
-0.5	7/7	7/7
-0.4	6/7	7/7

Кінець таблиці 7.5

Зміна в напівтонах	**Кількість правильно розпізнаних акордів цілком**	**Кількість правильно розпізнаних акордів по першій ноті**
-0.3	7/7	7/7
-0.2	7/7	7/7
-0.1	7/7	7/7
0	7/7	7/7
+0.1	7/7	7/7
+0.2	6/7	7/7
+0.3	6/7	7/7
+0.4	5/7	5/7
+0.5	0/7	0/7
+0.6	0/7	0/7

В таблиці N вказано результати розпізнавання. У другому стовпці вказано кількість повністю розпізнаних акордів, тобто якщо акорд Cmaj7, то очікується саме Cmaj7, а в третій колонці вказано кількість акордів, в яких правильно розпізнано першу ноту, тобто для Cmaj7 допустима відповідь просто C.

Таким чином ми можемо побачити, що при зміні тону на 0.3 полутони в будь-яку сторону програма все ще видає правильні відповіді із ймовірністю 6/7 (тобто 85%), і з ймовірністю 7/7 (100%) це буде акорд з тією самою першою нотою. При відхиленні більше 0.4 полутони ймовірність падає.

Цікаво що за відхилення 0.5 у сторону пониження тону програма все ще видає правильні результати – 7/7. Але за такого ж відхилення але в сторону підвищення тону, програма не дала жодної правильної відповіді. Але треба зауважити, що всі варіації акордів вказані правильно, але програма визначає їх вже як на півтону підняті.

Схоже, що округлення під час створення хромаграми працює саме так, що якщо тон нижче еталоного рівно на чверть тону, то це все ще вважається цим

самим тоном, але якщо тон більше на чверть тону, то це вже вважається наступна нота.

7.3.2 Експеримент 2 з неналаштованою гітарою

Інший випадок розлаштування гітари – це коли всі струни розлаштовані хаотично. Цей параметр доволі складно виміряти, тому скористаємось умовними позначками – ледве розлаштована, середньо розлаштована і сильно розлаштована. І проведемо 3 експерименти з тими самим акордами: Am7, B7, Cmaj7, Dm7, Em7, Fmaj7, G7. Результати експерименту наведені в таблиці 7.6.

Таблиця 7.6 – Результати експерименту 2

	Am7	**B7**	**Cmaj7**	**Dm7**	**Em7**	**Fmaj7**	**G7**
1	C	B7	Cmaj7	Dm7	F#7	F	D#maj7
	39.19	77.35	45.15	93.14	58.43	93.40	62.67
2	Fm	D#m-7	C7	Cm	F#7	Fm	G
	88.33	68.82	88.40	32.69	70.34	54.60	58.29
3	C7	G#m-7	Cmaj7	Em7	F#7	F	G
	44.17	92.74	54.23	99.80	67.07	85.66	75.58

В таблиці 7.7 наведено підсумовані результати розпізнавання.

Таблиця 7.7 – Результати розпізнавання

Номер експерименту	**Кількість правильно розпізнаних акордів цілком**	**Кількість правильно розпізнаних акордів по першій ноті**
1	3/7	4/7

Кінець таблиці 7.7

Номер експерименту	**Кількість правильно розпізнаних акордів цілком**	**Кількість правильно розпізнаних акордів по першій ноті**
2	0/7	3/7
3	1/7	3/7

Як ми можемо побачити, вже в першому експерименті, де гітара ледве розлаштована, кількість розпізнавань доволі низька – лише 3/7 (42%) повністю, і 4/7 (57%) – за першою нотою. Наступні експерименти мають ще менші показники 0/7 (0%) та 3/7 (42%) і 1/7 (14%) та 3/7 (42% відповідно).

З цього експерименту можна зробити висновок, що оскільки нейронна мережа навчалась на налаштованій гітарі, то результати розпізнавання аудіо, що записані на неналаштованій, дуже низькі. В такому випадку необхідно, або додати в навчання записи неналаштованої гітари, або попереджати користувача кінцевої программи, і просити налаштувати гітару перед використанням.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було проведено збір даних для научного дослідження, проаналізовано наявні методи розпізнавання, порівняно використання хромаграми і спектрограми, а також проведено експерименти із акордами, що не були наявні під час навчання нейромережі, і також експеримент із розлаштованою гітарою. Як результат було створено програму для розпізнавання акорду із аудіозапису.

Під час збору даних було записано аудіовиконання акордів на гітарі. Запис виконувався за допомогою звичайного диктофону на мобільному телефоні. Було записано акорди різних всі тонів (за виключенням дієзних) і різних видів (мінор, мажор, септакорди та інші). Після дані були опрацьовані, щоб на кожному окремому примірнику аудіо звучав один окремо зіграний акорд, та очищені від сміттєвих даних, яких важко уникнути під час запису.

Після того, отримані дані було доповнено. А саме, для скорочення часу запису, було створено пропущені дієзні тони акордів шляхом програмного підвищення тону оригінальних записів. Таким чином для навчання нейронної мережі додалися класи тонів дієзних акордів.

Аудіо було перетворено на зображення, а саме два види зображеннь: це спектрограми і хромаграми. Під час тренування нейромережі було виявлено, що модель, яка була навчена на хромограмах порається із розпізнаванням значно краще, аніж навчена на спектрограмах.

Модель, що навчена на хромограмах для зразків акордів, що не були в переліку навчання, видавала результати акордів, що приблизно збігаються по складу нот. Також під час експерименту із розлаштованою гітарою було визначено, що модель все ще доволі точно розпізнає акорди, якщо струни налаштовані відносно одна одної, але коли струни розлаштовані хаотично, то результати розпізнавання низькі.

В планах подальшої роботи є розробка бібліотеки, як pymovements [14].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mark Phillips, Jon Chappell: Gitara dla bystrzaków. Gliwice: Septem, 2008, ss. 251-252, seria: Dla bystrzaków. ISBN 978-83-246-0993-2

2. Юрій Юцевич. Музика: словник-довідник. – Тернопіль, 2003. – 404 с. – ISBN 966-7924-10-6

3. cjbayron. autochord [Електронний ресурс] / cjbayron – Режим доступу до ресурсу: https://github.com/cjbayron/autochord.

4. Das O. Chord-Recognition [Електронний ресурс] / Orchisama Das – Режим доступу до ресурсу: https://github.com/cjbayron/autochord.

5. J.-J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR), pages 157–163, 2002.

6. A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 173–178, 2008., L. Xiao, L. Liu, F. Seide, and J. Zhou. Learning a music similarity measure on automatic annotations with application to playlist generation. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing, pages 1885–1888, 2009.

7. M. Carre. Systèmes de Recherche de Documents Musicaux par Chantonnement. PhD thesis, École Nationale Supérieure des Télécommunications, Paris, 2002.

8. Clark B. The 7 Best Chord Identifier Websites and Apps (2023) [Електронний ресурс] / Brian Clark – Режим доступу до ресурсу: https://www.musicianwave.com/best-chord-identifier-websites-apps/.

9. Yamaha Corporation of America. Chord Tracker - US [Електронний ресурс] / Yamaha Corporation of America – Режим доступу до ресурсу: https://apps.apple.com/us/app/chord-tracker-us/id992502589.

10. librosa development team. librosa [Електронний ресурс] / librosa development team – Режим доступу до ресурсу: https://librosa.org/doc/latest/index.html.

11. Яровий М. В. Дослідження методів розпізнавання акордів для навчання гри на музичному інструменті / Яровий М. В.. – Харків, 2021. – 61 с.

12. A. Yerokhin, A. Nechyporenko, A. Babii and O. Turuta, "Usage of F-transform to finding informative parameters of rhinomanometric signals," *2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT)*, Lviv, Ukraine, 2015, pp. 129-132, doi: 10.1109/STC-CSIT.2015.7325449.

13. О. В Турута. Штучний інтелект крізь призму фундаментальних прав людини / О. В Турута, О. П. Турута // Науковий вісник Ужгородського національного університету. Серія: Право / О. В Турута, О. П. Турута., 2022. – С. 49–54.

14. pymovements: A Python Package for Eye Movement Data Processing / Daniel G. Krakowczyk, David R. Reich, Jakob Chwastek та ін.] // Preprint for ETRA '23: 2023 Symposium on Eye Tracking Research and Applications / Daniel G. Krakowczyk, David R. Reich, Jakob Chwastek та ін.]., 2023.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

15. A. Yerokhin, A. Nechyporenko, A. Babii and O. Turuta, "Usage of F-transform to finding informative parameters of rhinomanometric signals," *2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT)*, Lviv, Ukraine, 2015, pp. 129-132, doi: 10.1109/STC-CSIT.2015.7325449.

16. О. В Турута. Штучний інтелект крізь призму фундаментальних прав людини / О. В Турута, О. П. Турута // Науковий вісник Ужгородського національного університету. Серія: Право / О. В Турута, О. П. Турута., 2022. – С. 49–54.

17. pymovements: A Python Package for Eye Movement Data Processing / Daniel G. Krakowczyk, David R. Reich, Jakob Chwastek, Deborah N. Jakobi, Paul Prasse, Assunta Süss, Oleksii Turuta, Paweł Kasprowski, Lena A. Jäger] // Preprint for ETRA '23: 2023 Symposium on Eye Tracking Research and Applications / Daniel G. Krakowczyk, David R. Reich, Jakob Chwastek та ін.]., 2023.