Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Магістерська робота

освітній ступінь – магістр

на тему: «**КЛАСИФІКАЦІЯ ЕМОЦІЙ В АУДІОЗАПИСАХ УКРАЇНСЬКОЇ МОВИ МЕТОДАМИ ПРИРОДНОЇ ОБРОБКИ МОВИ**»

Виконав: студент 2-го року навчання,
Спеціальності 122 Комп’ютерні науки

Жулкевський Владислав Дмитрович

Керівник Ігнатенко Олексій Петрович
доктор фізико-математичних наук

Рецензент _____

Магістерська робота захищена з оцінкою

_____

Секретар ЕК _____

«__» _____ 20__ р.

Київ 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інф орматики

ЗАТВЕРДЖУЮ
доктор фізико-математичних наук
Ігнатенко О. П.

_____

(підпис)

„ ____”_____2022р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Жулкевському Владиславу Дмитровичу

факультету інформатики 2 курсу магістерської програми

ТЕМА: **Класифікація емоцій в аудіозаписах української мови методами природної обробки мови**

Зміст ТЧ до дипломної роботи:

Вступ

Розділ 1. Огляд існуючих методів аналізу звуку

Розділ 2. Збір датасету

Розділ 3. Реалізація моделей для розпізнавання емоцій в аудіо

Висновок

Список використаної літератури

Дата видачі „ ___” _____ 2022 р. Керівник_____ (підпис)

Завдання отримав ___ ____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапу курсового проекту (роботи)	Термін виконання
1.	Отримання завдання на кваліфікаційну роботу	05.07.2022
2.	Огляд літератури за темою роботи	06.07.2022 - 25.10.2022
3.	Огляд технологій представлення аудіо та написання першого розділу «Огляд існуючих методів аналізу звуку»	21.10.2022 - 22.11.2022
4.	Огляд та тестування моделей глибинного навчання для розпізнавання емоцій в аудіо	23.11.2022 - 08.01.2023
5.	Створення власних моделей для задачі розпізнавання емоцій	08.01.2023 - 11.03.2023
7.	Проведення експериментів на розпізнавання емоцій для створених моделей	11.03.2020- 29.03.2020
8.	Написання розділу «Збір датасету»	29.03.2023 - 15.04.2023
9.	Експерименти та підбір гіперпараметрів для моделей з найкращими результатами	15.04.2023- 30.04.2020
10.	Аналіз результатів експериментів та написання розділу «Реалізація моделей для розпізнавання емоцій в аудіо»	30.04.2023- 14.05.2023
11.	Попередній захист	15.05.2023
12.	Корегування роботи за результатами попереднього захисту	15.05.2023 – 30.05.2023
13.	Створення презентації	30.05.2023 - 06.06.2023
14.	Подання роботи на кафедру для перевірки на плагіат	07.06.2023
15.	Захист кваліфікаційної роботи	12.06.2023

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

ДПФ – Дискретне перетворення Фур’є

ШПФ – Швидке перетворення Фур’є

MFCC – Мелчастотні кепстральні коефіцієнти

LFCC – Лінійночастотні кепстральні коефіцієнти

CNN – згорткова нейронна мережа

RNN – рекурентна нейронна мережа

AST – Трансформер звукової спектрограми

ВСТУП

Розпізнавання емоцій людей у звукових записах - це актуальна та важлива задача в сфері обробки мовних сигналів та штучного інтелекту. З розвитком технологій та збільшенням об'єму аудіо-інформації, що генерується людьми, стає дедалі більш важливим забезпечити її якісний аналіз. Обробка аудіо даних включає в себе різні техніки та методи, такі як аналіз частот, аналіз тональності, виявлення та розпізнавання різних звуків, включаючи мову та емоції. В контексті аналізу емоцій виникає проблема як правильно інтерпретувати ту чи іншу емоцію людини. Тому розпізнавання емоцій є особливо важким завданням, оскільки воно вимагає розуміння відтінків голосу, що можуть відображати емоційний стан людини. Іншою проблемою, пов’язаною із обробкою вхідних аудіо сигналів, є нестабільність аудіозаписів. Рівень шуму, якість запису, особливості мовлення людини та інші фактори можуть суттєво впливати на точність розпізнавання. Окрім цього, багато поточних методів аудіоаналізу зосереджені на англійській мові, що викликає додаткові проблеми для обробки аудіозаписів інших мов, зокрема і української. Це робить задачу ще більш складною, але і важливішою для дослідження.

Кваліфікаційна робота присвячена розробці моделей розпізнавання емоцій людей в аудіозаписах. Основною метою роботи є дослідження методів та алгоритмів машинного та глибинного навчання, які можна використовувати для виявлення мовних сигналів та їх аналізу на предмет виявлення емоцій співрозмовника.

У роботі будуть проаналізовані існуючі методи розпізнавання емоцій в мовленні, описані їх переваги та недоліки, вибрані найбільш ефективні методи для подальшого використання в розробці моделей. Буде проведений аналіз створених моделей з використанням датасету звукових записів, який дозволить оцінити її точність.

Перелік завдань цієї кваліфікаційної роботи:

1. Дослідження існуючих методів аналізу аудіо.

2. Підготовка набору даних для навчання власних моделей розпізнавання голосу та сентиментального аналізу.
3. Розробка програмних моделей на основі нейронних мереж для сентиментального аналізу аудіо.
4. Аналіз результатів вихідних моделей на підготовлених датасетах української та англійської мов.

Робота складається з 3 розділів.

Перший розділ призначено висвітленню методів представлення та обробки звукових сигналів. Зокрема представлено основні характеристики звуку, використання частотного та часового представлень звукових хвилей для подальшої роботи з аудіо у вигляді спектрограм, кепстральних коефіцієнтів тощо.

В другому розділі описаний збір та обробка даних з різних датасетів. Зокрема, використано дані кол-центру, які містять репліки українською мовою, датасет аудіо з студії озвучування мультфільмів. Також для порівняння роботи моделей було обрано розмічений датасет англійською мовою RAVDESS. У розділі описані методи попередньої обробки даних та представлення звуку у вигляді кепстральних коефіцієнтів та спектрограм.

У третьому розділі наводиться вибір та реалізація моделей для навчання та тренування на обраних датасетах. Зокрема, розписуються архітектура моделі CNN, RNN, ResNet та трансформера. У розділі відображуються результати метрик даних моделей на тестових даних та порівнюються із іншими моделями у зведеній таблиці результатів.

У результаті виконання дослідження проаналізовано точність роботи створених та існуючих моделі для виявлення емоцій. Кваліфікаційна робота має на меті розв'язати важливу задачу визначення емоцій людей у звукових записах за допомогою сучасних методів та алгоритмів машинного навчання. Результати роботи можуть бути корисними у багатьох сферах, включаючи медицину, психологію, маркетинг та соціальну роботу.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ АНАЛІЗУ ЗВУКУ

1.1 Представлення звуку

Звук - це коливання або поширення змін тиску в середовищі, що створює звукові хвилі. Ці звукові хвилі мають характеристики такі як частота, амплітуда, період та фаза хвилі, які можуть бути виміряні та проаналізовані. Розглянемо детальніше кожну з цих характеристик звуку та звукових хвиль [1].

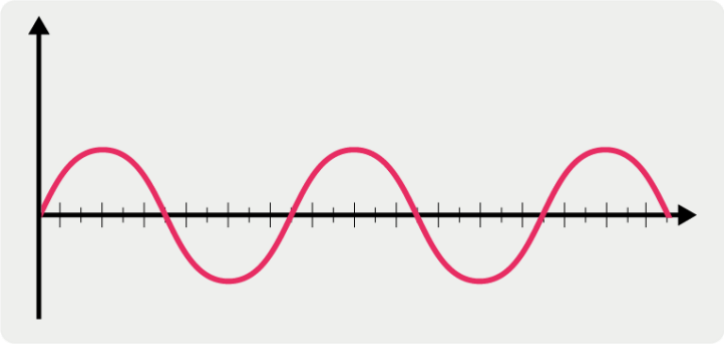


Рисунок 1. Звукова хвиля у часовій шкалі

Амплітуда звукової хвилі відображає максимальне відхилення частин середовища від їх рівноважного положення при поширенні звуку (Рис. 2). Вона відображається як висота пікових значень хвилі і визначає гучність звуку. Більша амплітуда відповідає гучнішому звуку, тоді як менша амплітуда - тихішому звуку.

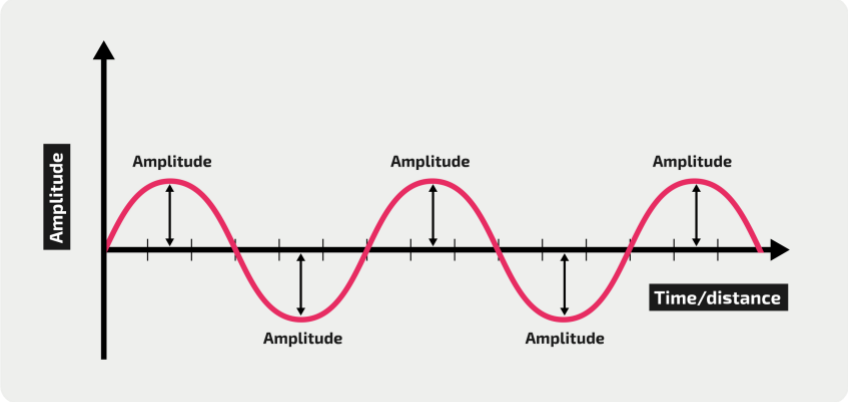


Рисунок 2. Амплітуда звукової хвилі

Частота звуку визначається кількістю коливань або циклів, які звукова хвиля здійснює за одиницю часу (Рис. 3). Якщо частота звукової хвилі збільшується, то за секунду відбувається більше циклів, і звук виходить більш високого тону. Якщо частота зменшується, то видається звук нижчої висоти.

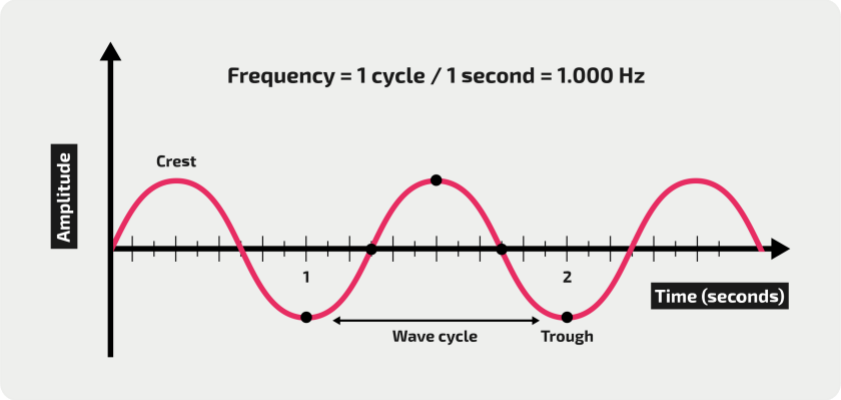


Рисунок 3. Частота хвилі

Хвиля має повторюваний патерн, а одне завершене повторення називається циклом. Період часу - це час, необхідний для завершення циклу. Повний цикл відбувається, коли звукова хвиля проходить між двома послідовними точками.

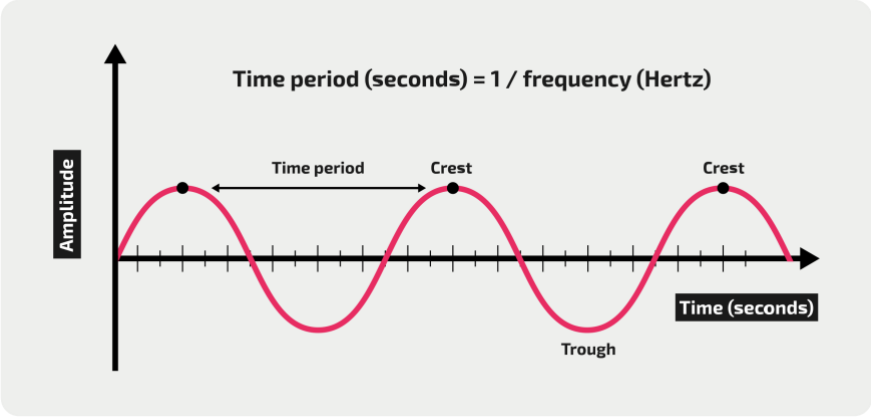


Рисунок 4. Період часу звукової хвилі

Фаза — кількісна характеристика, що визначає відмінність між двома подібними коливаннями, які починаються в різні моменти часу. Для детального розуміння фази, представимо синусоїдальну хвилю наступним рівнянням:

$$y(t) = Asin(\omega t + \varphi_0),$$

де

- A – амплітуда хвилі,
- ω – кутова частота, яка вказує, скільки циклів відбувається за секунду, в радіанах за секунду,
- φ – початкова фаза сигналу, тобто значення фази коливань (повної) в початковий момент часу, тобто при $t = 0$.

В цій формулі вираз $(\omega t + \varphi_0) = \varphi_t$ і є фазою хвилі. Одиницею фази є радіан (1 рад).

Знаючи характеристики звуку, нам потрібно знати як представити звук у цифровому форматі. І для цієї задачі, найпоширенішою технікою є дискретизація. Під час дискретизації звуку комп'ютер вимірює аналоговий сигнал через певні проміжки часу, а потім кожному проміжку присвоюється унікальний двійковий ланцюжок. Після цього оцифрований звук може зберігатися і оброблятися комп'ютером як послідовність нулів та одиниць.

Частота дискретизації - це кількість проміжків за секунду, які беруться з форми хвилі для створення дискретного цифрового сигналу. Частота дискретизації визначає діапазон частот, що захоплюються в цифровому аудіо. Чим вища частота дискретизації, тим більше даних ми отримуємо з аудіосигналу. Тобто висока частота дискретизації призводить до менших втрат інформації, але більших обчислювальних витрат, а низька частота дискретизації має більші втрати інформації, але швидше обчислюється.

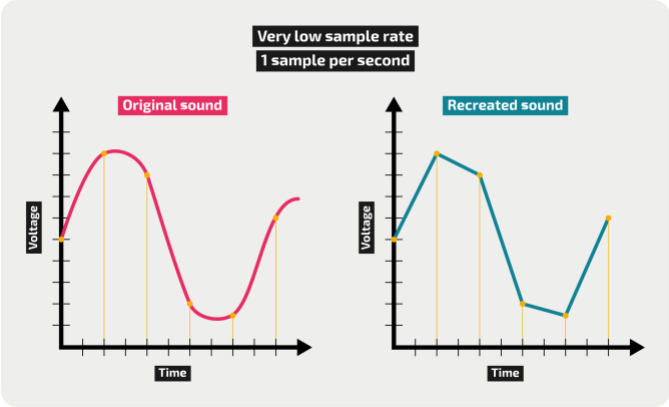


Рис. 5.Приклади низької частоти дискретизації звукової хвилі

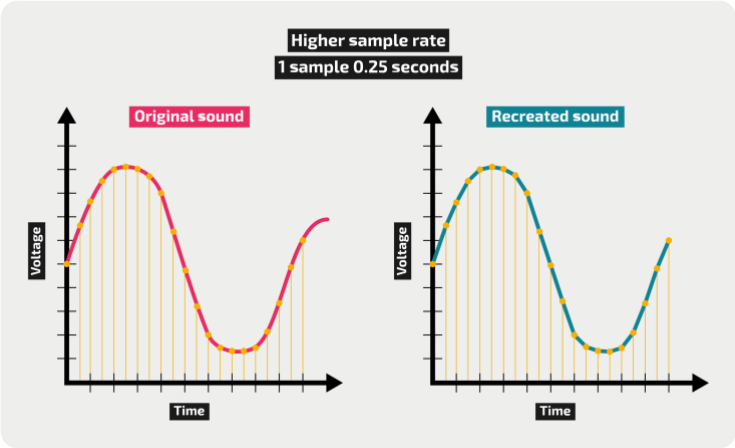


Рис. 6. Приклади високої частоти дискретизації звукових хвиль

Зрозумівши як представляються звукові дані у комп'ютерних системах, можемо розглянути засоби обробки цього представлення звуку для отримання характеристик, які будуть використані для дослідження.

1.2 Основні методи обробки звуку

З попереднього розділу ми дізналися, як можна легко охарактеризувати хвилю за допомогою періоду/частоти, амплітуди, фази та представити звук за допомогою дискретизації. Проте це легко зробити для простих періодичних сигналів, таких як синусоїдальні або косинусоїдальні хвилі. Для складних хвиль це не так просто, тому існують методи представлення звукових хвиль для аналізу, і кожен з них може бути використаний для різних застосувань. Деякі з найпоширеніших методів описані далі.

1.2.1 Дискретне перетворення Фур’є

Для аналізу хвиль складної форми використовують найбільш відомий метод, а саме дискретне перетворення Фур'є, який розкладає будь-який сигнал на суму простих синусоїдальних і косинусоїдальних хвиль, для яких ми можемо легко виміряти частоту, амплітуду і фазу. Перетворення Фур'є можна застосовувати до неперервних або дискретних хвиль. Використовуючи перетворення Фур’є, ми можемо розкласти сигнал на серію синусоїд, кожна з яких матиме свою частоту. На рисунку 7 показано ідею ДПФ, яка полягає в тому, що результуючий сигнал насправді є результатом суми 3 різних синусоїд. Сигнал у часовій області може бути перетворений у фігуру в частотній області, яка називається амплітудним спектром ДПФ, де частоти сигналу показані у вигляді вертикальних смуг. Висота стовпчика після нормалізації є амплітудою сигналу в часовій області. Можемо побачити, що 3 вертикальні смуги відповідають 3 частотам синусоїди, які також зображені на рисунку 7 [2].

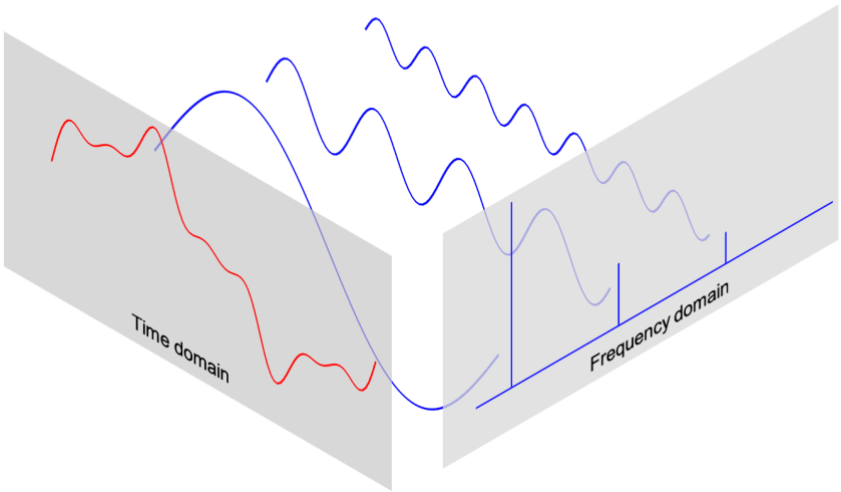


Рис. 7. Приклад переведення сигналів з часової в частотнуобласть за допомогою ДПФ

Для того, щоб перетворити послідовність рівномірно розподілених сигналів в інформацію про частоту всіх синусоїдальних хвиль, які потім

необхідно підсумувати до часової області сигналу, було визначено формулу дискретного перетворення Фур’є. Вона має наступний вигляд:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) * e^{-j\frac{2\pi}{N}kn}, k = 0,1 \dots, N-1,$$

де

- N – кількість семплів
- n – поточний семпл
- k – поточна частота, де $k \in [0, N-1] k \in [0, N-1]$
- x_n – значення синусоїди на семплі n
- X_k – ДПФ, яке включає інформацію про амплітуду та фазу хвилі.

Для обчислення ДПФ можна скористатися алгоритмом швидкого перетворення Фур'є. ШПФ - це ефективний алгоритм для обчислення ДПФ звукової послідовності. Вперше його описано в класичній статті Кулі і Тьюкі в 1965 році [3], але насправді ідею можна простежити до неопублікованої роботи Гаусса в 1805 році. Це алгоритм "розділяй і володарюй", який рекурсивно розбиває ДПФ на менші ДПФ, щоб зменшити обсяг обчислень. В результаті, він успішно зменшує складність ШПФ з $O(n^2)$ до $O(nlogn)$, де n – розмір даних. Таке скорочення часу обчислень є значним, особливо для даних з великими N завдяки чому ШПФ стало широко використовуватися для обчислень ДПФ.

1.2.2 Вейвлет-аналіз

З практичної точки зору перетворення Фур’є має деякі обмеження і недоліки. Воно має добру локалізацію у частотній області, але не має локалізації у часовій. Частина недоліків долається за допомогою віконного перетворення Фур’є. Проте і у нього є недолік, а саме фіксоване вікно, яке має фіксовану роздільну здатність у часі та частоті для всієї площини перетворення, що не адаптовано до складних сигналів. Саме тому для задач з локалізацією на часовій площині можна використовувати вейвлет-аналіз [4].

Для початку, наведемо визначення вейвлету. Вейвлет - це хвилеподібне коливання, локалізоване в часі. Вейвлети мають дві основні властивості: масштаб і розташування. Масштаб визначає, наскільки "розтягнутим" або "стиснутим" є вейвлет. Ця властивість пов'язана з частотою, визначеною для хвиль. Розташування визначає, де вейвлет знаходиться в часі або просторі. Приклади вейвлетів можна переглянути у додатку А.

Таким чином, вейвлет-аналіз - це метод аналізу звуку, який використовується для розкладання складного сигналу на суму простих вейвлет-функцій різного масштабу та частоти. У вейвлет-аналізі використання вейвлет-функції вирішує проблему відсікання сигналу. В цьому випадку вейвлет зсувається вздовж сигналу і для кожної нової позиції обчислюється спектр. Потім цей процес повторюється багато разів з трохи коротшим (або довшим) вейвлетом для кожного нового циклу. Врешті-решт результатом буде колекція часово-частотних зображень сигналу з різною роздільною здатністю. Саме так вейвлет-аналіз може допомогти визначити короткочасні зміни в сигналі та виявити деталі, які не були видимі під час Фур'є-аналізу.

1.2.3 Спектрограма

Спектрограма - це візуальне представлення сигналу в залежності від часу та частоти. Спектрограма дозволяє визначити, які частоти домінують в різний час та як змінюється спектр звуку з часом. Розбиваючи сигнал на коротші сегменти, ми можемо зосередитися на властивостях сигналу в певний момент часу. Розбиваючи сигнал на проміжки, які називаються вікнами, і обраховуючи дискретне перетворення Фур'є для кожного вікна, ми отримуємо короткочасне перетворення Фур'є сигналу. Зокрема, для вхідного сигналу і вікна, перетворення визначається наступною формулою:

$$STFT\{x_n\}(h,k) = X(h,k) = \sum_{n=0}^{N-1} x_{n+h} w_n e^{-i2\pi\frac{kn}{N}},$$

де

- $X(h,k)$ - STFT вхідного сигналу,

- x_n – початковий сигнал, представлений у вигляді функції чи вектора,
- w_n – обрана віконна функція, яка використовується для ізоляції короткого сегмента вхідного сигналу для аналізу,
- N – це довжина вікна, використаного для аналізу
- h – момент часу для обраного вікна,
- k – індекс частотного біна. Він визначає роздільну здатність частотного аналізу. Діапазон k зазвичай від 0 до N−1.

STFT є одним з найпоширеніших інструментів в аналізі та обробці мовлення. Він описує еволюцію частотних компонентів у часі. Результатом STFT є комплексна величина, хоча якщо спектр є вектором, то вихід STFT є матрицею. Як наслідок, ми не можемо безпосередньо візуалізувати комплексну величину. Замість цього, STFT зазвичай візуалізують за допомогою лог-спектрів. Такі двовимірні лог-спектри можна візуалізувати за допомогою теплової карти, відомої як спектрограма [6].

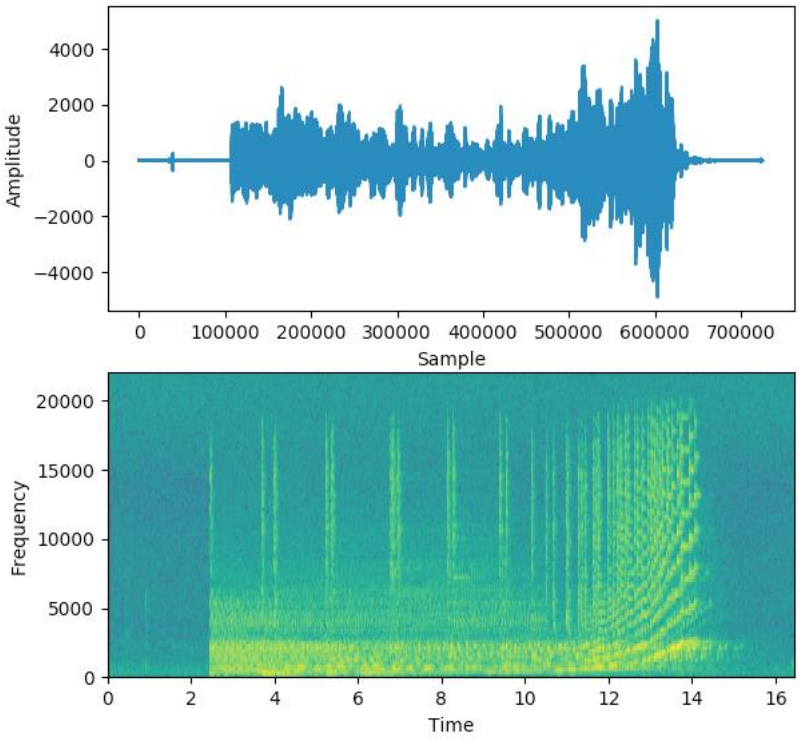


Рис. 8. Приклад спектрограми для вхідного сигналу

1.2.4 Кепстральний аналіз

Кепстральний аналіз є методом обробки сигналів, що поєднує спектральний аналіз з логарифмічною шкалою. Він дозволяє аналізувати спектральну інформацію сигналу у домені кепстра.

Перший крок у кепстральному аналізі - це взяти логарифм від модуля спектра сигналу, отриманого за допомогою перетворення Фур'є. Логарифмування дозволяє перейти від множників до доданків, знижуючи вплив амплітудної нелінійності. Після логарифмування виконується обернене перетворення Фур'є для отримання кепстрального спектра, також відомого як кепстр. Кепстральний спектр відображає енергію сигналу в залежності від часових затримок, які можуть бути пов'язані з різними частотними компонентами сигналу.

Формула кепстрального коефіцієнту може бути представлена наступним чином:

$$C(n) = \mathcal{F}^{-1}\{\log(|\mathcal{F}\{x(t)\}|^2)\}$$

де:

- $C(n)$- кепстральний коефіцієнт для індексу n,
- $x(t)$- вхідний сигнал у домені часу,
- $\mathcal{F}\{x(t)\}$ - перетворення Фур'є вхідного сигналу,
- $\mathcal{F}^{-1}$ - обернене перетворення Фур'є.

На рисунку 9 можна переглянути різницю у представленні звукового сигналу у вигляді спектрограми та кепстральних коефіцієнтів.

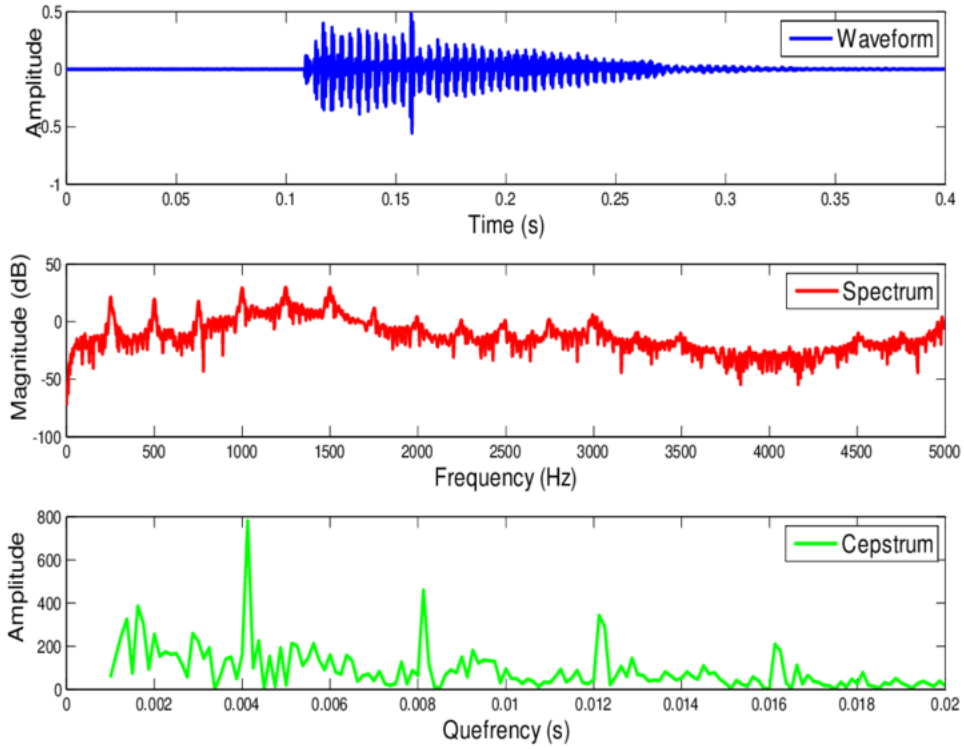


Рис. 9. Графіки представлення хвилі у вигляді спектруму та кепструму [7]

1.2.5 Спектрограма Мела

Спектрограма Мела - це спектрограма, яка замість лінійної шкали на осі частот використовує шкалу Мела. Іншими словами, вона відображає частоти в спектрограмі за шкалою Мела, щоб краще відобразити те, як людина сприймає звук. Шкала Мела - це перцептивна шкала висот тонів, які слухачі вважають рівними за відстанню один від одного. Точка відліку між цією шкалою і звичайним вимірюванням частоти визначається шляхом присвоєння тону з частотою 1000 Гц, що на 40 дБ вище порогу чутності слухача, висоти звуку в 1000 мелів. Починаючи з частоти близько 500 Гц, слухачі оцінюють все більші інтервали як такі, що дають рівні прирости висоти тону. Таким чином, у 1987 році Девідом Шонесі була виведена формула для перетворення частоти в герцах до частоти в мелах [8]:

$$m = 2595log_{10}\left(1 + \frac{f}{700}\right),$$

З досліджень цих спектрограм було виявлено, що при роботі з низькими частотами на етапі попередньої обробки аудіо варто використовувати саме мел-спектрограму, адже вона забезпечує кращу роздільну здатність для нижчих частот, ніж спектрограма.

1.2.6 Мелчастотні кепстральні коефіцієнти (MFCC)

MFCC (Mel Frequency Cepstral Coefficients) - це метод перетворення вхідного звукового сигналу, який обчислюється з короткострокового спектру сигналу та надає представлення його кепстральних характеристик [9].

Алгоритм обчислення MFCC складається із 6 головних кроків:

1. Розбиття на проміжки: Сигнал розбивається на короткі проміжки, які перекривають один одного. Типові довжини проміжків коливаються від 20 до 40 мілісекунд.
2. Віконне згладжування: Кожен проміжок множать на функцію вікна, зазвичай на вікно Хеммінга або Ханнінга, для зменшення спектрального витоку.
3. ДПФ: Застосовується дискретне перетворення Фур'є до кожної віконної рамки для отримання частотного спектра.
4. Фільтр-банк Мела: Спектр проходить через набір трикутних фільтрів, розміщених на шкалі Мела.
5. Логарифмування: Для перетворення значень фільтр-банку в логарифмічну шкалу береться логарифм від результатів минулого кроку.
6. Дискретне косинусне перетворення (DCT): Отримані значення фільтр-банку набувають незалежності за допомогою DCT для отримання кепстральних коефіцієнтів.

Головний недолік цього методу - це його низька стійкість до шуму, оскільки шумові сигнали змінюють всі коефіцієнти, якщо хоча б одна частотна смуга зміщена. Для підвищення стійкості MFCC до зашумлених мовних

сигналів використовують різні методи нормалізації як для навчальних, так і для тестових датасетів.

1.2.7 Лінійночастотні кепстральні коефіцієнти (LFCC)

Цей метод є схожим на попередній метод MFCC, проте LFCC використовує лінійну шкалу частот. Тобто на 4 кроці алгоритму з обчислення MFCC для алгоритму LFCC буде використовуватися лінійна шкала. Також LFCC використовує прямокутні фільтри замість трикутних фільтрів, як у MFCC. При обрахуванні LFCC не відбувається логарифмічного стиснення, яке виконується на 5 кроці обраховування MFCC. Для задач класифікації аудіо здебільшого використовується MFCC, проте LFCC може краще виявляти ознаки при фоновому шумі.

Таким чином, у цьому розділі було детально розглянуто способи представлення аудіо для подальшого використання в дослідженні. Зокрема розглянуто дискретизацію звукового сигналу, основні характеристики звуку та їх використання для подальшої обробки звукового сигналу. Із наведених у цьому розділі методів обробки сигналу, одними з найпоширеніших є спектрограма, мелспектрограма, мелчастотні кепстральні коефіцієнти та лінійночастотні кепстральні коефіцієнти. У розділі 3 розповідається про використання цих методів для попередньої обробки датасету аудіо української мови.

2 ЗБІР ДАТАСЕТУ

2.1 Видобування та аналіз даних

Оскільки відкритих розмічених датасетів аудіозаписів українською мовою для задачі виявлення емоцій наразі немає, для проведення навчання моделей та експериментів розпізнавання емоцій було вирішено самостійно зібрати датасет українських аудіо з різними емоціями. Під час отримання завдання для кваліфікаційної роботи від наукового керівника також було отримано початковий датасет аудіо з розміченими емоціями. Цей датасет належить компанії, яка займається підтримкою клієнтів. Датасет представлений 85 аудіозаписами формату .mp3 з кол-центру цієї компанії, у яких ведеться діалог двох осіб. Якщо розділити усі репліки кожної людини, ми отримаємо 441 репліку тривалістю від 1 до 5 секунд. Частота дискретизації аудіо цього датасету складає 8000 Гц, кількість каналів - 1. Основні класи емоцій в датасеті – це позитивна емоція (радість), нейтральна та негативна (злість) емоції. На аудіозаписах можна почути репліки як від жінок так і чоловіків різного віку, проте у датасеті розмічених даних про цю інформацію немає. Також у датасеті наявні репліки українською та російською мовами. Розподіл емоцій та тривалості аудіо даного датасету можна побачити на рисунках . Бачимо, що тривалість аудіофайлів змінюється від 0.5 секунд до 41 секунди. Кількість нейтральних емоцій у датасеті становить 363, кількість позитивних – 9, кількість негативних – 69. Розподіл довжин та класів цього датасету представлений на рисунках 2.1 та 2.2.

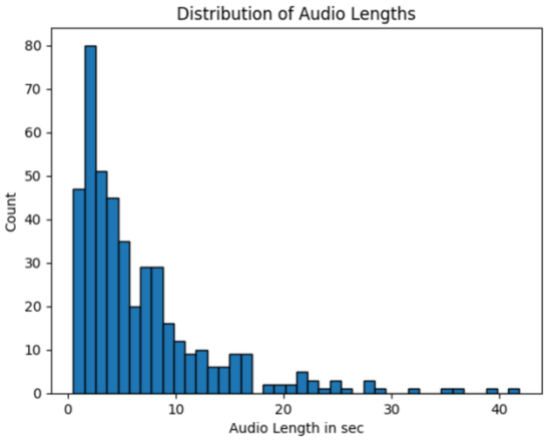


Рисунок 2.1. Розподіл реплік в датасеті кол-центру по довжині в секундах

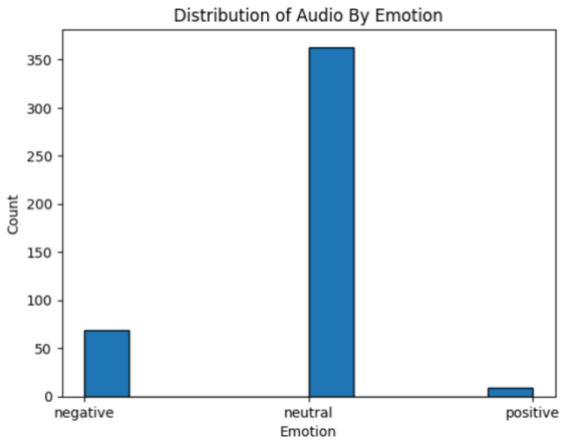


Рисунок 2.2. Розподіл реплік в датасеті кол-центру по емоціях

Так як представлений датасет є малим з точки зору кількості записів, а також через те, що цей датасет є сильно не збалансованим у кількості класів, було вирішено збільшити наявний датасет власними даними. Для цього було використано дані з озвучки українською мовою мультфільму «Нестерпні прибульці» студією «Clan Kaizoku». На аудіозаписах присутні один жіночий голос та один чоловічий. Після попередньої обробки даних, було отримано 859 аудіозаписів формату .wav. Частота дискретизації цих аудіозаписів становить 48кГц, кількість каналів - 1. Розподіл тривалості аудіофайлів та емоцій серед цих аудіо представлений на рисунках . Максимальна тривалість аудіо – 9.58 секунд, мінімальна – 0.32. Розподіл класів: 212 – позитивних емоцій, 301 – нейтральні, 345 – негативні. Розподіл довжин та класів датасету представлений на рисунках 2.3 та 2.4.

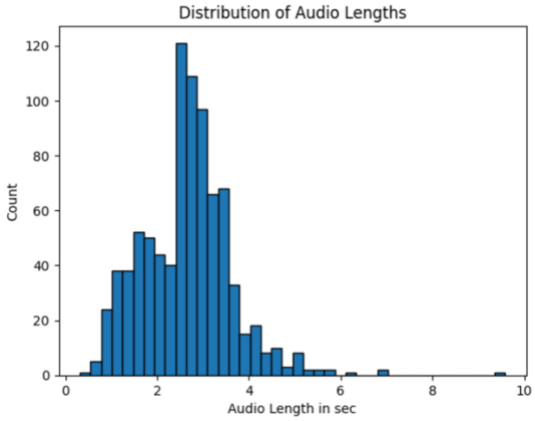


Рисунок 2.3. Розподіл реплік з датасету студії звукозапису по довжині в секундах

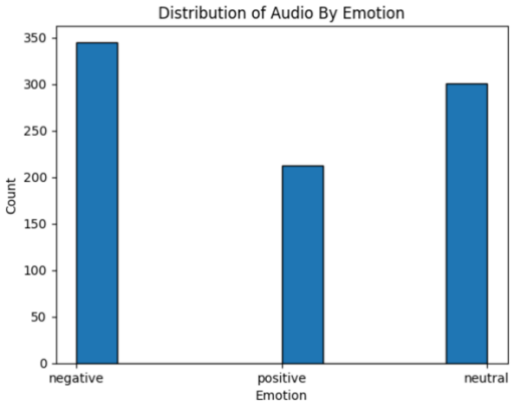


Рисунок 2.4. Розподіл реплік з датасету студії звукозапису по емоціях

Також для перевірки результатів моделей взято датасет RAVDESS, у якому містяться аудіозаписи англійською мовою для визначення емоцій [11]. Датасет RAVDESS має 1440 аудіо від 24 спікерів. Датасет має 8 класів емоцій, проте у роботі буде використовуватися всього 3, які були представлені раніше. Для проведення тренувань на даному датасеті довелося його відфільтрувати, залишивши аудіо із мітками радість, нейтральність та злість. У підсумку зазначений датасет має 479 аудіозаписів. Розподіл емоцій цього датасету представлений на рисунку. Усі аудіо мають тривалість від 3 до 5 секунд. Частота дискретизації аудіо – 48кГц, кількість каналів - 1. У датасеті представлено 192 позитивні та негативні мітки, і 96 нейтральних. Розподіл довжин та класів даного датасету представлений на рисунках 2.1 та 2.2.

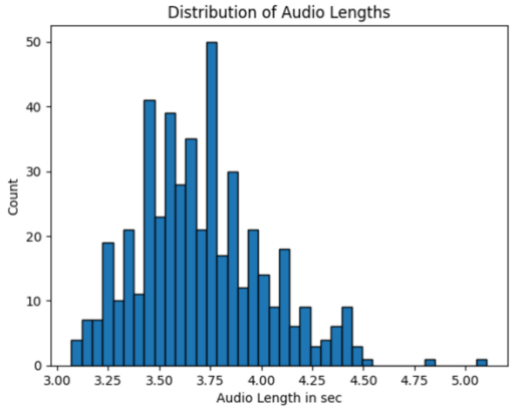


Рисунок 2.5. Розподіл реплік з RAVDESS датасету по довжині в секундах

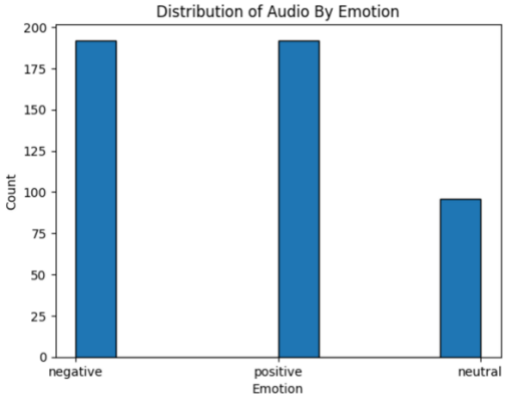


Рисунок 2.6. Розподіл реплік з RAVDESS датасету по емоціях

2.2 Підготовка датасетів

Для того, щоб зібрати всі аудіозаписи у датасет, необхідно було розділити великі аудіо тривалістю кілька хвилин на частини, у яких були б присутні різні репліки людей. Для цього було використано бібліотеку для мови python pandas ,pydub та noisereduce, а також фреймворк для обробки звуку ffmpeg [12].

Для обробки датасету від кол-центру потрібно було нарізати аудіо по часовим відміткам, які були вказані у txt файлах та перевести дані файли з форматиу .mp3 до формату .wav. Для цього було використано фреймворк ffmpeg, який зміг нарізати аудіо за вказаними секундами та зберігати новий файл на диск. Таким самим методом була зроблена нарізка аудіо для датасету української озвучки. При створенні нарізаних аудіо, дані про ці аудіо заносилися в датафрейм pandas, для зручного зчитування даних у подальших кроках. У таблицях зберігався шлях до файлу та відповідна емоція репліки. Лістинг коду для нарізки аудіо наведений у додатках.

Під час попереднього переслуховування аудіо, було виявлено, що дані з озвучки мультфільму містять фоновий шум англійської озвучки або фонову музику, який може негативно впливати на розпізнавання аудіо. Тому було вирішено прибрати фоновий шум, використовуючи бібліотеку noisereduce. Ця бібліотека прибирає фоновий шум з аудіо за рахунок обчислення спектрограми сигналу та оцінки шумового порогу для кожної частотної смуги. Потім цей поріг використовується для обчислення маски, яка відсікає наявний шум нижче обрахованого порогу. Лістинг коду для прибирання шуму можна побачити у додатку .

Після проведення всіх операцій над аудіо, за допомогою бібліотеки pandas усі окремі файли збираються у один загальний файл датасету. На рисунку 2.7 можна побачити перші 5 рядочків файлу з розміченими емоціями.

	Name	Emotion
0	my_noise_converted/НП_01_0_28_000_0_29_500_neg...	negative
1	my_noise_converted/НП_01_0_34_040_0_36_580_pos...	positive
2	my_noise_converted/НП_01_0_36_830_0_39_790_neu...	neutral
3	my_noise_converted/НП_01_0_39_790_0_42_500_pos...	positive
4	my_noise_converted/НП_01_0_52_000_0_53_710_neu...	neutral

Рисунок 2.7. Перші 5 рядків csv файлу з розміткою емоцій

2.3 Попередня обробка аудіо

Для навчання моделей машинного навчання нам потрібно представити всі дані з аудіофайлу у вигляді тензорів з уніфікованими розмірами. Під час зчитування файлу, дані представляються у вигляді звукової хвилі у частотному розрізі. Перед тим, як робити перетворення звуку, спочатку потрібно привести всі дані до однієї частоти дискретизації та до однієї тривалості. Враховуючи, що тривалість даних у датасеті є різною, було розроблено два методи для збільшення та зменшення вхідного сигналу до одного розміру. Для збільшення хвилі можна використовувати різні методи, такі як заповнення аудіо нулями, копіювання звуку до потрібної довжини та обрізання зайвої частини, метод ковзаючого вікна тощо. Зокрема, у роботі було використано перші два із зазначених методів. У випадку з наповненням аудіо нулями було використано модуль torch.nn.functional.pad, який наповнює тензор нулями до заданого розміру. Для копіювання аудіо кілька разів було розроблено власний метод, який повторював n-ну кількість раз вхідний сигнал, де n визначається за формулою $n = \left\lceil \frac{\max\_samples}{current\_samples} \right\rceil$. Після того, як аудіо склоновано n разів, частина сигналу, яка перевищує максимально визначену кількість просто обрізається. Так само якщо початковий сигнал був більшим за максимально допустиме значення дискретизації сигналу, то він обрізався до цього значення.

При попередній обробці також важливо слідкувати за частотою дискретизації сигналу. Якщо дані з датасетів мають різні дискретизації, це може призвести до спотворення результатів навчання моделі. Тому важливо привести дані до однієї частоти дискретизації, що робиться за допомогою модуля torchaudio.transforms.Resample.

Враховуючи наявність різноманітних моделей, які підходять під задачу класифікації аудіо, можливі ситуації, коли потрібно представити сигнал з різною кількістю каналів. Для цього був також створений метод, який робить перетворення вхідного тензора до тієї кількості каналів, яка потрібна для вхідних розмірностей тензорів у моделі.

Далі у роботі було використано методи перетворення звуку у мелспектрограми, MFCC та LFCC, які були згадані у розділі 1.2. Ці методи допомагають представити характеристики аудіо у вигляді спектрограм. Кожен метод використовує різні параметри, які відповідно визначають результуючі представлення сигналу. Для визначенння функцій перетворень використовувалися готові функції з модуля torchaudio.transforms. Для MFCC основними параметрами є sample_rate - частота дискретизації, n_mfcc - кількість кепстральних коефіцієнтів мела, які зберігатимуться після роботи трансформера. Також існують додаткові параметри, які задаються у словнику melkwargs, із них використовуються n_fft – для задання кількості точок у вікні ШПФ, hop_length – для задання кількості відліків, на яку переміщується вікно під час обчислення спектрограми, center – визначає як вирівнюється значення спектрограми. При True, значення вирівнюється до середини сигналу, при False – обрахування спектрограми починається з початку сигналу. На рисунку 2.8 представлено лістинг можливого визначення функції трансформації нашого вхідного сигналу до MFCC.

```
N_MFCC = 64
N_FFT = 1024
HOP_LENGTH = 512

mfcc_transform = torchaudio.transforms.MFCC(
    sample_rate=SAMPLE_RATE,
    n_mfcc=N_MFCC,
    melkwargs={"n_fft": N_FFT, "hop_length": HOP_LENGTH, "center": False}
)
```

Рисунок 2.8. Визначення трансформації MFCC для представлення аудіо сигналу

Набір параметрів для LFCC є ідентичним до MFCC, окрім назви цих параметрів. Зокрема замість n_mfcc використовується n_lfcc і також визначає кількість вихідних коефіцієнтів функції трансформації. Лістинг коду наведений на рисунку 2.9.

```
lfcc_transform = torchaudio.transforms.LFCC(
     sample_rate=SAMPLE_RATE,
     n_lfcc=40,
     speckwargs={"n_fft": 400, "hop_length": 160, "center": False})
```

Рисунок 2.9. Визначення трансформації LFCC для представлення аудіо сигналу

Також у роботі використовується Мелспектрограма для трансформації початкового сигналу. ЇЇ параметрами є n_fft, hop_length, sample_rate, а також n_mels – кількість каналів Мел-фільтрів. Лістинг трансформації представлено на рисунку 2.10.

```
mel_spectrogram = torchaudio.transforms.MelSpectrogram(
        sample_rate=SAMPLE_RATE,
        n_fft=1024,
        hop_length=512,
        n_mels=64,
)
```

Рисунок 2.10. Визначення трансформації MelSpectogram для представлення аудіо сигналу

Таким чином, усі параметри, вказані для трансформації початкового сигналу де-факто є гіперпараметрами моделі, оскільки безпосередньо впливають на кількість та розміри вхідних тензорів для моделей.

Приклади представлення аудіо у вигляді звукової хвилі та наведених перетворень можна побачити на рисунку 2.11.

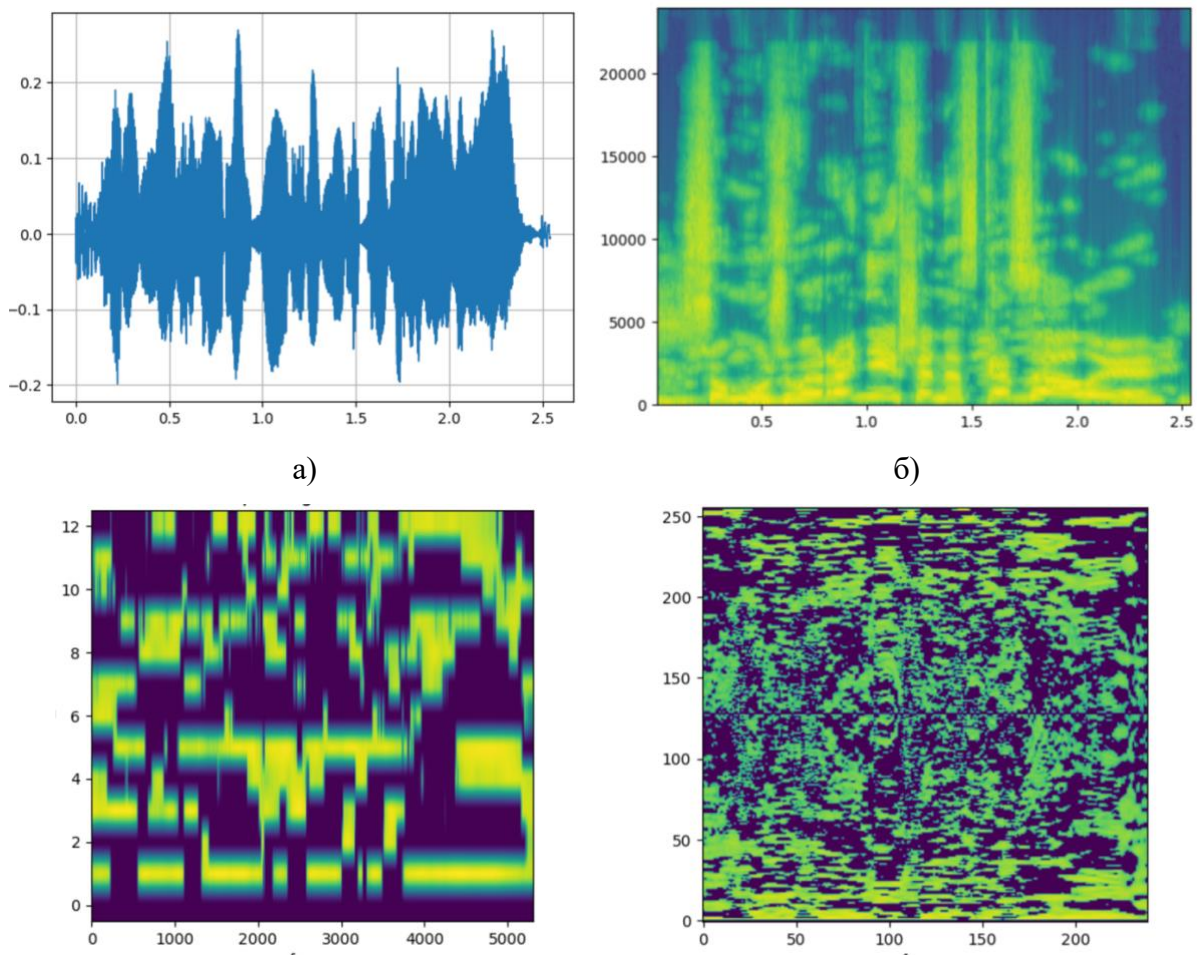


Рисунок 2.11. Представлення даних з аудіофайлу тривалістю 2.5 секунди та частотою дискретизації 48кГц у формі а) хвилі в частотному розрізі, б) спектрограми Мела, в) MFCC, г) LFCC

Після визначення алгоритму попередньої обробки аудіосигналу, було вирішено створити власний клас CustomDataset, який наслідує клас torch.utils.data.Dataset для зручного доступу до даних датасету. Цей клас буде відповідати за завантаження даних з аудіофайлів та перетворень, вказаних раніше. У результаті, клас реалізовуватиме метод доступу за індексом, який

буде повертати перетворений сигнал до одного з представлень MFCC, LFCC чи Спектрограми Мела разом із класом емоції, якій належить даний аудіосигнал. Лістинг коду цього класу представлений у додатку. Після того, як було написано реалізацію даного класу, екземпляри даного класу можна використовувати у класі DataLoader. DataLoader є класом у бібліотеці PyTorch, який допомагає у завантаженні та підготовці даних для навчання моделі. Він дозволяє пакетно завантажувати дані та надає зручний ітеративний інтерфейс для доступу до даних під час навчання моделі. Також він може перемішувати дані перед кожною епохою навчання, що є корисним для забезпечення випадкового розподілу даних під час навчання. За допомогою пакетної обробки ми можемо обробляти більші обсяги даних і використовувати більш оптимізовані функції пакетного обчислення у фреймворку PyTorch при цьому підтримуючи паралельну підготовку даних, що надає можливість підготувати наступний пакет даних під час обробки поточного.

Таким чином, у цьому розділі було розглянуто основні датасети, які будуть використовуватися у дослідженнях, показано процес їх збору та підготовки, а також процес обробки вхідного сигналу для отримання представлень, які надаватимуть нам необхідні ознаки для навчання моделей. Розглянуто реалізацію методів представлення у вигляді Мел спектрограми, MFCC та LFCC. Створено реалізацію власного класу датасету CustomDataset для ефективного завантаження даних для моделей.

3 РЕАЛІЗАЦІЯ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ ЕМОЦІЙ В АУДІО

3.1 Реалізація згорткової нейронної мережі

З підготовленим українським та англійським датасетом, можна приступати до створення моделей для навчання. Автором роботи було вирішено розробити згорткову нейронну мережу для задачі класифікації емоцій. Для розробки моделі використовувався фреймворк PyTorch [13] з модулем PyTorchAudio для допоміжних функцій при обробці аудіо. Сама модель складається з 15 шарів. Розглянемо ці шари детальніше:

1. Conv2d: 2D згортковий шар з 1 вхідним каналом і 16 вихідними каналами. Ядро для згорткового шару має розмір 3x3, крок (stride) дорівнює 1, а padding дорівнює 2.

2. BacthNorm2d: шар нормалізації, який нормалізує вихід згорткового шару.

3. Conv2d: Другий згортковий шар приймає 16 вхідних каналів (вихід з попереднього шару) і видає 32 вихідних каналу. Усі інші параметри такі ж, як і в першому згортковому шару.

4. BacthNorm2d: Далі йде наступний шар нормалізації, який нормалізує вихід другого згорткового шару.

5. Conv2d: Третій згортковий шар приймає 32 вхідних каналів і видає 64 вихідних каналу.

6. BacthNorm2d: Ще один шар нормалізації, який нормалізує вихід третього згорткового шару.

7. Conv2d: Четвертий згортковий шар приймає 64 вхідних каналів і видає 128 вихідних каналу.

8. BacthNorm2d: останній шар нормалізації, який нормалізує вихід з останнього згорткового шару.

9. Шар Dropout, який "вимикає" деякі нейрони в процесі навчання випадковим чином, щоб уникнути перенавчання.

10. Лінійний шар (Linear), який приймає вектор фіксованого розміру і видає вектор розміру NUM_CLASSES, який відповідає кількості класів у нашій задачі класифікації, тобто 3 класи емоцій.

11. Шар LogSoftmax, який приймає вихід лінійного шару і видає ймовірності для кожного класу.

Також кожен з згорткових шарів в моделі активується за допомогою функції ReLU перед подальшою нормалізацією, що додає до моделі ще 4 проміжних шари. У результаті отримаємо модель, яка матиме 3857763 параметрів для навчання (Рис. 3.1).

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 66, 130]             160
              ReLU-2          [-1, 16, 66, 130]               0
       BatchNorm2d-3          [-1, 16, 66, 130]              32
            Conv2d-4          [-1, 32, 68, 132]           4,640
              ReLU-5          [-1, 32, 68, 132]               0
       BatchNorm2d-6          [-1, 32, 68, 132]              64
            Conv2d-7          [-1, 64, 70, 134]          18,496
              ReLU-8          [-1, 64, 70, 134]               0
       BatchNorm2d-9          [-1, 64, 70, 134]             128
           Conv2d-10         [-1, 128, 72, 136]          73,856
             ReLU-11         [-1, 128, 72, 136]               0
      BatchNorm2d-12         [-1, 128, 72, 136]             256
        Dropout2d-13         [-1, 128, 72, 136]               0
           Linear-14                     [-1, 3]       3,760,131
       LogSoftmax-15                     [-1, 3]               0
================================================================
Total params: 3,857,763
Trainable params: 3,857,763
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.03
Forward/backward pass size (MB): 61.71
Params size (MB): 14.72
Estimated Total Size (MB): 76.45
----------------------------------------------------------------
```

Рисунок 3.1. Шари та кількість параметрів для моделі CNN

Далі переходимо до процесу навчання та оцінювання точності моделі. Навчання проводилося з використанням таких метрик як точність та багатокласовий f1score з модуля torch.metrics, а у якості початкового оптимізатора було обрано Adam з модуля torch.optim. Learning rate становить 0.001, кількість епох 10. Під час попередньої обробки аудіо використовується копіювання сигналу для приведення усіх сигналів до одного розміру. Також

однією із метрик, яка використовувалася для перевірки результатів класифікації, є матриця невідповідностей, яка показує розподіл правильно та неправильно класифікованих даних. Для тренування, валідації та тестування моделі використовується 70%, 15% та 15% даних відповідно.

Для перших експериментів було зроблена наступна порівняльна таблиця 1. У цій таблиці наведені результати навчання та тестування моделей на двох датасетах – на датасеті RAVDESS та на об'єднаному датасеті кол-центру та озвучених аудіо. По результатам тренування та тестування датасетів можна побачити, що найкращі результати для обох датасетів дає представлення звуку у вигляді мелчастотних кепстральних коефіцієнтів MFCC. Для датасету RAVDESS точність та f1Score становить приблизно 0.68 та 0.53 відповідно, а для датасету реплік кол-центру та озвучування точність та f1Score становить приблизно 0.60 та 0.46 відповідно. Також бачимо, що для власного датасету другий найкращий результат дав LFCC з точність 0.57 та f1score – 0.57. Найгірший результат дала спектрограма з точністю розпізнавання – 0.54 та f1score 0.38. Проте для датасету RAVDESS другий найкращий результат дало представлення Мел Спектрограма з точністю 0.59 та f1score 0.51, а найгірший результат – LFCC з точністю 0.54 та f1score 0.42.

Dataset	Metric	LFCC	MFCC	MelSpectogram
Call center + voice acting	Accuracy	0.5684523837906974	**0.6026785714285714**	0.5372023837906974
	F1Score	0.3869362899235317	**0.4614541360310146**	0.38173965045384
RAVDESS	Accuracy	0.5416666666666666	**0.6770833333333334**	0.59375
	F1Score	0.42312441269556683	**0.5296610395113627**	0.5126200318336487

Таблиця 1. Порівняння результатів метрик CNN на тестових даних для різних видів перетворень звукового сигналу

Визначивши найкращий результат, який показує CNN для різних трансформацій, ми можемо підібрати гіперпараметри для покращення поточних результатів. Спочатку, був зменшений learning_rate від 0.001 до 0.0001 поточної моделі. Точність та f1score для власного датасету зменшилися до 0.57 та 0.42 відповідно, а для англійського датасету RAVDESS навпаки покращилися через

меншу кількість даних, відповідно зменшився ступінь перенавчання. Метрики навчання на датасеті RAVDESS – точність 0.71, f1score – 0.68. Після цього було вирішено додати нормалізацію даних після трансформації у кепстральні коефіцієнти, що дало невелике покращення результату точності до 0.64 для власного датасету, проте f1score зменшився до 0.43. Для RAVDESS датасету точність покращилася до 0.71, а f1score показав найвищу відмітку серед усіх варіантів моделі – 0.684. Наступним кроком було вирішено змінити параметри MFCC трансформації. Попередніми параметрами трансформації були 64 кепстральні коефіцієнти мела (n_mfcc), 1024 точок у вікні ШПФ (n_fft), 512 відліків вікна спектрограми та center - False, оскільки аудіо має початок репліки з 0 секунди. Було визначено інші параметри для цієї трансформації, а саме n_mfcc - 16, n_fft - 400, hop_length – 64. Зі зміною цих параметрів точності розпізнавання для обох датасетів погіршилися, зменшившись до 0.56 та 0.59, а f1score – зменшився для власного датасету до 0.49 та зріс до 0.62 для датасету RAVDESS. Також було вирішено змінити метод приведення довжини аудіо до одного розміру, а саме зробити заповнення сигналу нулями до визначеної довжини. Така зміна також дала погіршення результату навчання до 0.56 точності та f1score - 0.38.

Результати усіх моделей можна побачити у таблиці 2. Жирним шрифтом виділені найкращі результати точності та f1 score. Таким чином, за допомогою додаткових технік обробки у вигляді нормалізації вдалося визначити найкращі параметри для згорткової нейронної мережі відразу для обох датасетів.

Dataset	Metric	Base MFCC 64, Copying, LR = 0.001, Without normalization	LR - 0.0001	With normalization	MFCC 16	Padding
Call center + voice acting	Accuracy	0.6026785714285714	0.5758928571428571	**0.6383928571428571**	0.5595238123621259	0.5639880980764117
	F1Score	**0.4614541360310146**	0.4172184807913644	0.4280605741909572	0.4880585031849997	0.378029784985951
RAVDESS	Accuracy	0.6770833333333334	**0.7083333333333334**	**0.7083333333333334**	0.625	0.59375
	F1Score	0.5296610395113627	0.6770833333333334	**0.6843018929163615**	0.6191267172495524	0.4649765590826670

Таблиця 2. Результати підбору гіперпараметрів та змін у попередній обробці сигналів

На рисунках 3.2 та 3.3 зображено матриці невідповідності для найкращих результатів метрик для обох датасетів. Класи пронумеровано від 0 до 2, де 0 – це позитивна емоція, 1 – нейтральна, 2 – негативна. З матриць бачимо, що найкраще модель розпізнавала нейтральну емоцію, найгірше – позитивну.

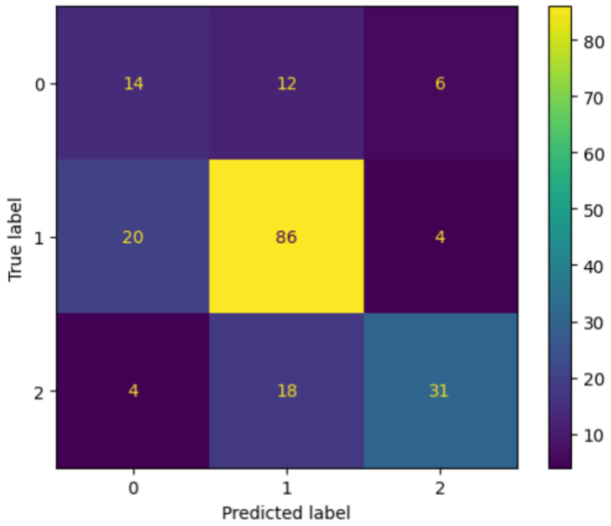


Рисунок 3.2. Матриця невідповідності для результатів тренування CNN на власному датасеті

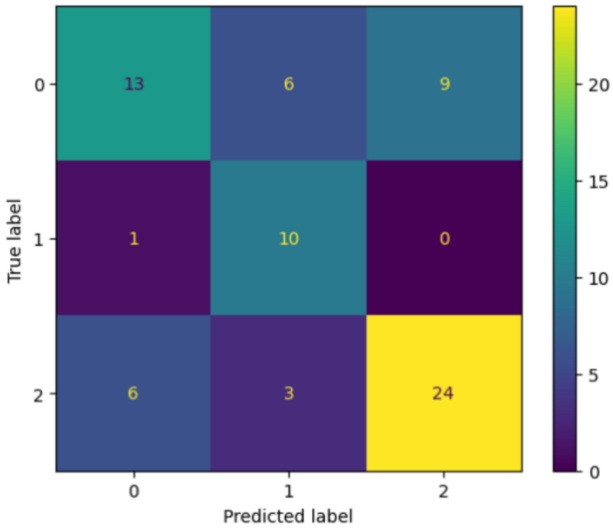


Рисунок 3.3. Матриця невідповідності для результатів тренування CNN на RAVDESS датасеті

3.2 Реалізація рекурентної нейронної мережі

Рекурентна нейронна мережа також може слугувати моделю для розпізнавання емоцій з аудіо. У роботі було реалізовано модель рекурентної нейронної мережі з кількома шарами. Основним шаром у даній моделі є LSTM шар. LSTM (Long Short-Term Memory) шар є одним з типів рекурентних нейронних мереж. Він був розроблений з метою подолання проблеми зникання градієнта, що виникає у звичайних рекурентних нейронних мережах.

LSTM шар складається з клітинного стану і трьох воріт: ворота забування (forget gate), ворота введення (input gate) і ворота виведення (output gate). Кожне з цих воріт виконує операції з вхідними даними для керування потоком інформації у клітинному стані. Основна ідея полягає в тому, що LSTM шар може зберігати довготривалу залежність в послідовності даних, яка може бути

важлива для правильного передбачення або класифікації. Кожен елемент послідовності оброблюється окремо і передає свою інформацію через часові кроки. Кожна операція в LSTM шарі має свої ваги, які автоматично налаштовуються під час процесу навчання з використанням методу зворотного поширення помилки (backpropagation). Ці ваги змінюються таким чином, щоб мінімізувати помилку моделі на тренувальних даних. Загалом, LSTM шар використовується для моделювання довготривалих залежностей в послідовних даних, і його архітектура дозволяє ефективно керувати потоком інформації через часові кроки, забезпечуючи більш точні передбачення або класифікацію.

Таким чином, LSTM шар було ініціалізовано відповідно до розмірів вхідних даних та кількості прихованих нейронів. Також застосовується параметр dropout між шарами LSTM для регуляризації моделі. Далі визначені три лінійні шари, які здійснюють класифікацію на основі вихідного стану LSTM. Кожен шар крім останнього має ReLU активацію. В результаті модель буде мати 276803 тренованих параметрів, як видно з рисунку 3.4.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
AudioLSTM                                [16, 3]                   --
├─LSTM: 1-1                              [16, 64, 128]             264,192
├─Linear: 1-2                            [16, 64]                  8,256
├─ReLU: 1-3                              [16, 64]                  --
├─Linear: 1-4                            [16, 64]                  4,160
├─ReLU: 1-5                              [16, 64]                  --
├─Linear: 1-6                            [16, 3]                   195
==========================================================================================
Total params: 276,803
Trainable params: 276,803
Non-trainable params: 0
Total mult-adds (M): 270.73
==========================================================================================
Input size (MB): 0.52
Forward/backward pass size (MB): 1.07
Params size (MB): 1.11
Estimated Total Size (MB): 2.70
==========================================================================================
```

Рисунок 3.4. Шари та кількість параметрів моделі RNN

Для навчання цієї моделі було також використано метрики точність та f1score. У якості оптимізатора було використано Adam оптимізатор. Початкова кількість епох становила 20, а learning rate – 0.0001.

Перші результати застосування цієї моделі з різними представленнями звуку можна побачити у наступній таблиці 3:

Dataset	Metric	LFCC	MFCC	MelSpectogram
Call center + voice acting	Accuracy	0.4439102571744185	**0.5112179494821109**	0.45993589896422166
	F1Score	0.35035193195709813	**0.41896237432956696**	0.39933934120031506
RAVDESS	Accuracy	0.5125	0.525	**0.675**
	F1Score	0.4365427404642105	0.4736339569091797	**0.6241839826107025**

Таблиця 3. Порівняння результатів метрик RNN на тестових даних для різних видів перетворень звукового сигналу

З результатів бачимо, що найкраща точність для власного датасету становить 0.511, а f1score – 0.42 при представленні сигналу у вигляді MFCC. Для RAVDESS датасету найкращі результати розпізнавання були досягненні при обробці сигналу до спектрограми Мела: точність – 0.675, f1score – 0.624. LFCC показав найгірші показники для тестування на тестовому даних обох датасетів.

Матриці невідповідностей для обох датасетів представлені на рисунках.

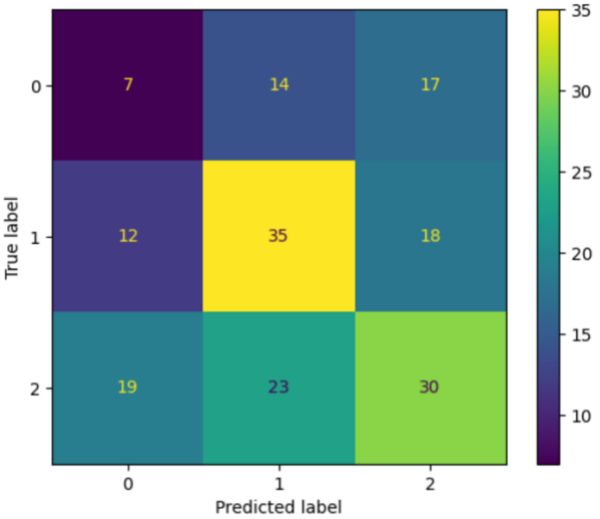


Рисунок 3.5. Матриця невідповідності для результатів тренування CNN на власному датасеті

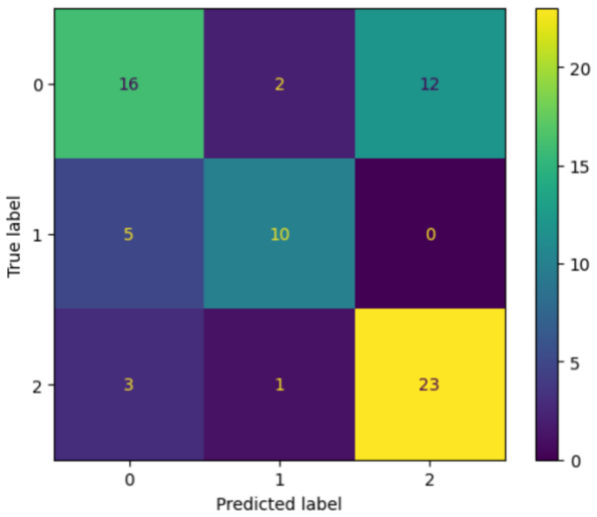


Рисунок 3.6. Матриця невідповідності для результатів тренування CNN на RAVDESS датасеті

3.3 Передавальне навчання з моделлю ResNet50

Для порівняння результатів було вирішено скористатися готовою моделю ResNet50. Архітектура моделі ResNet представлена у додатку Д.

Ця мережа була навчена та призначена для класифікації картинок, проте оскільки аудіо у цій роботі представляються у вигляді спектрограм та кепстральних коефіцієнтів, то ми можемо передати ці дані на вхід до цієї мережі. Проте нам потрібно підібрати розмір вхідного зображення, щоб отримати кращі результати розпізнавання класів за допомогою цієї моделі. Для ResNet50 вхідні розміри тензорів складають [224,224]. При зчитуванні аудіо файлу моно звук представлений тільки одним каналом, а для моделі потрібно 3 канали. Відповідно було використано метод тензору repeat, який перетворює 1 канал на 3 ідентичні канали. Також було використано трансформацію Resize, для приведення вхідних даних до розміру [224, 224]. Також моделі були передані попередньо натреновані на датасеті IMAGENET1K_V2 ваги [14]. Результати роботи моделі можна побачити у таблиці. Бачимо, що ця модель має одні з найгірших показників розпізнавання на тестовому датасеті. Для власного датасету точність моделі склала 0.5669642857142857, а f1score - 0.3368342880691801, для датасету RAVDESS точність - 0.5, а f1score - 0.37497853934764863.

Dataset	Call center + voice acting dataset	RAVDESS
Accuracy	0.5669642857142857	0.5
F1Score	0.3368342880691801	0.37497853934764863

Таблиця 4. Результати метрик RNN на тестових даних різних датасетів

Також нижче наведені матриці невідповідності для тестових даних двох датасетів.

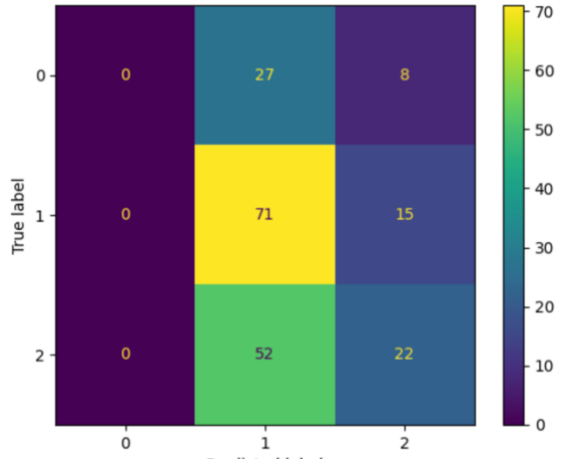


Рисунок 3.7. Матриця невідповідності для результатів тренування моделі ResNet на власному датасеті

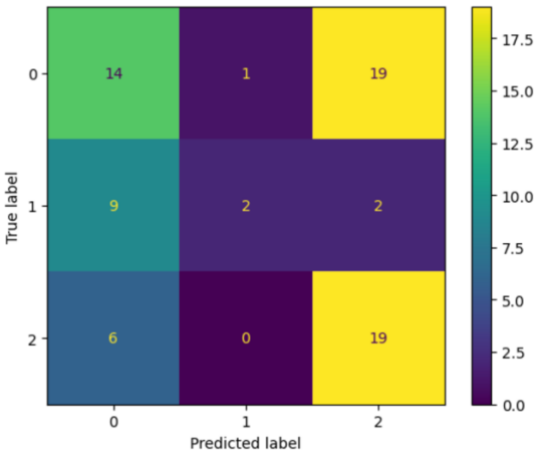


Рисунок 3.8. Матриця невідповідності для результатів тренування моделі ResNet на RAVDESS датасеті

З матриці невідповідності бачимо, що для власного датасету модель не розпізнала жодної позитивної емоції. А для RAVDESS датасету модель майже не розпізнала жодної нейтральної емоції, що і спричинює погані результати для цієї моделі.

3.4 Трансформери

У доповіді «Attention is all you need» [15] було представлено трансформер – архітектуру нейронної мережі, яка базується виключно на механізмах самоуваги (self-attention), усуваючи необхідність у рекурентних або згорткових шарах. Цей механізм дозволяє моделі зважувати важливість різних позицій у послідовності вхідних даних. На відміну від традиційних RNN, трансформер може обробляти всі позиції паралельно, що призводить до високого рівня розпаралелювання обчислень і скорочення часу навчання.

Трансформер складається з енкодера та декодера. Енкодер приймає вхідну послідовність і перетворює її на послідовність зображень, використовуючи нейронні мережі з механізмом самоуваги і прямим поширенням. Декодер генерує вихідну послідовність, звертаючи увагу на

представлені дані з енкодера і передбачаючи по одному токену за раз. Механізм самоуваги в трансформерах дозволяє йому фіксувати залежності між усіма вхідними та вихідними позиціями, що дає змогу краще моделювати довгострокові залежності. Він обчислює вагу уваги для кожної вхідної позиції на основі релевантності інших позицій, що дозволяє моделі фокусуватися на важливих частинах вхідних даних під час прогнозування.

Таким чином, для вирішення задачі класифікації емоції використано модель трансформеру MIT/ast-finetuned-audioset-10-10-0.4593, яка представлена на платформі Hugging face у розділі класифікації аудіо з одним з найбільших показників завантаженості. З опису моделі: «Даний трансформер для спектрограм аудіо еквівалентний ViT, але застосовується до аудіо. Аудіо спочатку перетворюється на зображення (у вигляді спектрограми), після чого застосовується Visual Transformer. Модель отримує найкращі результати за кількома критеріями класифікації аудіо» [16]. Архітектура моделі представлена на рисунку 3.9.

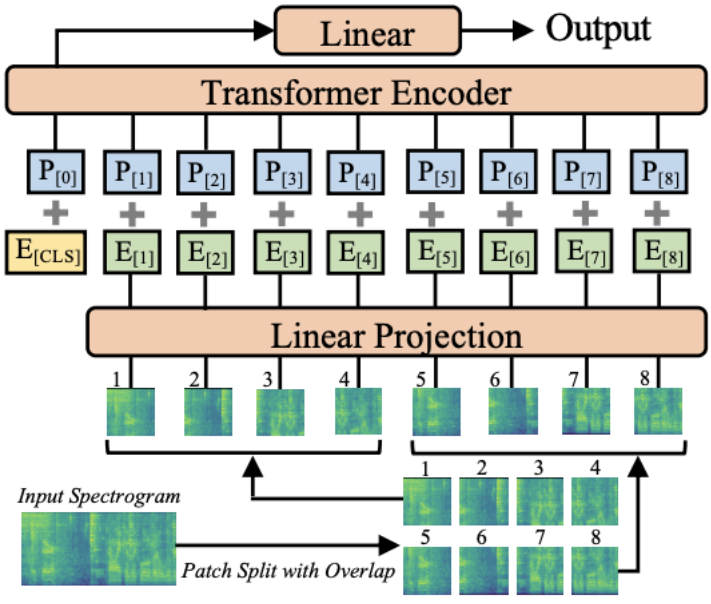


Рисунок 3.9. Архітектура моделі AST (Трансформер звукової спектрограми)

Для навчання моделі AST було використані бібліотеки transformers, datasets та evaluate від hugging face. Для користування даною моделю потрібно використати спеціальні класи AudioFeatureExtractor та

AutoModelForClassification. AudioFeatureExtractor створює спектрограми з вхідних сигналів з аудіофайлів, а AutoModelForClassification використовується у ролі моделі, якій зазначається конкретна модель з платформи Hugging face. Клас AudioFeatureExtractor потребує іншого представлення вхідних даних, тому було використано PyTorch CustomDataset, представлений у розділі 2.3, для зчитування даних та відповідної обробки до формату, який приймає даний клас. Також для тренування та рахування метрик був використаний клас Trainer, який надає зручний інтерфейс для показу результатів навчання на тестовому, валідаційному та тестовому датасетах.

Таким чином, модель була протестована на 3 конфігураціях для власного датасету і на одній конфігурації для датасету RAVDESS. Спочатку, для тренування було визначено learning rate 0.005 та кількість епох 10, і результати для власного датасету були наступними: точність склала 0.5692307692307692, а f1score - 0.5389010989010988. Далі через оверфітинг моделі вже на 5-6 епосі, було вирішено зменшити кількість епох до 5. В результати точність покращилася до 0.615, а f1-score – майже не змінився - 0.5386617398987871. Також було вирішено зменшити learning rate до 0.0005. Після цієї зміни показники точності та f1score зменшилися до 0.5743589743589743 та 0.5147650417215636 відповідно. Таким чином, найкращий результат було досягнуто з learning rate до 0.005 та 5-ма епохами. Тому саме цю конфігурацію було використано для тренування та тестування моделі на датасеті RAVDESS. В результаті тестування отрималися найкращі результати розпізнавання для даного датасету серед усіх моделей, які були представлені у роботі. Точність склала – 0.875, f1score – 0.8760791532443627. Усі результати тестування моделі на обох датасетах представлені у таблиці 5.

Dataset	Call center + voice acting			RAVDESS
Train params	LR 0.005 + 10 epochs	LR 0.005 + 5 epochs	LR 0.0005 + 5 epochs	LR 0.005 + 5 epochs
Accuracy	0.5692307692307692	**0.6153846153846154**	0.5743589743589743	**0.875**
F1Score	**0.5389010989010988**	0.5386617398987871	0.5147650417215636	**0.8760791532443627**

Таблиця 5. Результати метри на тестуванні моделі AST

Нижче наводяться матриці невідповідності для найкращих результатів для двох датасетів.

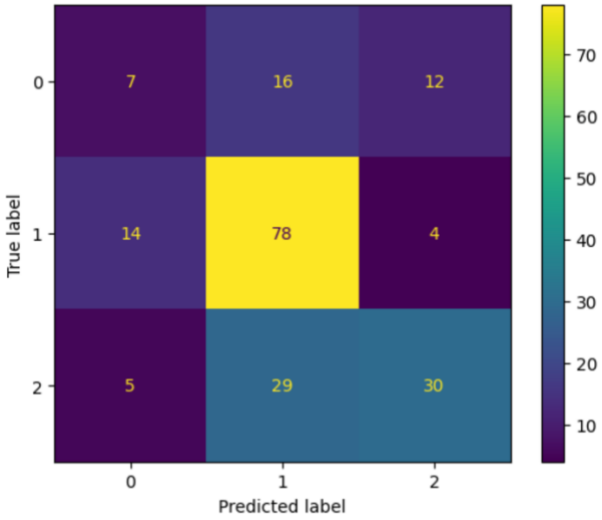


Рисунок 3.10. Матриця невідповідності для результатів тренування моделі AST на власному датасеті

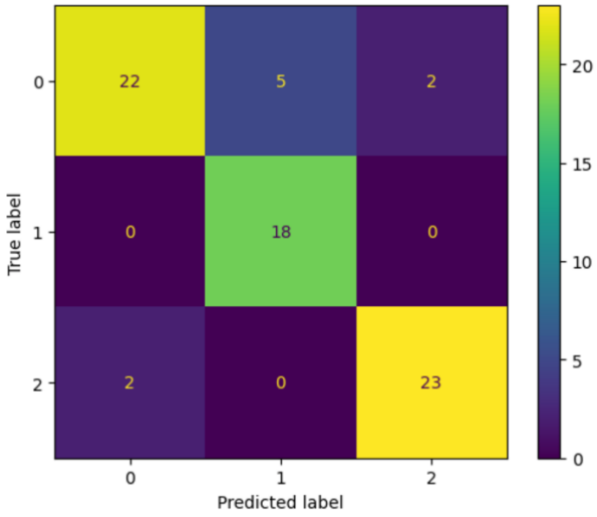


Рисунок 3.11. Матриця невідповідності для результатів тренування моделі AST на RAVDESS датасеті

Бачимо, що для RAVDESS датасету кількість неправильно розпізнаних емоцій дуже мала. Для власного датасету найкраще розпізнавалася нейтральна емоція, проте і хибно розпізнаних нейтральних емоцій теж дуже багато.

3.5 Аналіз результатів

В результаті виконання експериментів на всіх моделях, можна зробити порівняльну таблицю з найкращими метриками кожної моделі при навчанні на кожному навчальному датасеті. У зведеній таблиці результатів для датасету кол-центру, об’єднаного з аудіофайлами студії звукозапису, бачимо, що

найкращі результати розпізнавання емоцій показали CNN модель та трансформер. Згорткова мережа показала найбільшу точність, проте трансформер показав найбільший f1score. Найгірший f1Score показала мережа ResNet, найгіршу точність – RNN.

	CNN	RNN	ResNet	Transformer
Accuracy	**0.6383928571428571**	0.5112179494821109	0.5669642857142857	0.6153846153846154
F1Score	0.4280605741909572	0.41896237432956696	0.3368342880691801	**0.5386617398987871**

Таблиця 6. Найкращі результати роботи моделей на власному датасеті.

Для оцінювання метрик розпізнавання емоцій моделі було протестовано на датасеті RAVDESS, який представлений аудіо англійською мовою. Найкращу точність та f1score розпізнавання емоцій показав трансформер із точністю 0.875 та 0.876 відповідно. Найгірший результат по обом метрикам показала мережа ResNet.

	CNN	RNN	ResNet	Transformer
Accuracy	0.7083333333333334	0.675	0.5	**0.875**
F1Score	0.6843018929163615	0.6241839826107025	0.37497853934764863	**0.8760791532443627**

Таблиця 7. Найкращі результати роботи усіх моделей на датасеті RAVDESS

Підсумовуючи, у цьому розділі було наведено реалізації та результати тренування та тестування моделей на різних представленнях аудіосигналу та конфігурацій моделей. У результаті дослідження було виявлено, що найкращий результат по метрикам точність та f1 score надає модель трансформера звукової спектрограми.

ВИСНОВКИ

У цій роботі було проведено дослідження задачі розпізнавання емоцій з аудіо. Було розглянуто основні характеристики аудіо, методи представлення аудіо, техніки попередньої обробки звукових сигналів. У процесі роботи було створено власні моделі згорткової та рекурентної нейронної мережі для вирішення цієї задачі. Також були використані готові моделі ResNet та трансформер для порівняння результатів виявлення емоцій в аудіо. У розділі 3.5 наведено наведено порівняльну таблицю з найкращими результатами по двом метриками - f1score та точність та проведено аналіз результатів. Виявлено, що найкращою моделлю для задачі класифікації емоцій серед усіх моделей є трансформер. Це може пояснюватися тим, що однією із головних переваг цієї моделі є механізм самоуваги. Трансформер звукової спектрограми показав найкращі результати для обох датасетів, з показниками точності 0.615 та f1score 0.53 для українського датасету та показниками точності 0.875 та 0.876 для англійського датасету. Таким чином, можна зробити висновок, що він найбільше підходить для цієї задачі серед усіх визначених моделей.

Невелика точність результатів для датасету українською мовою може пояснюватися тим, що кожне аудіо має різні характеристики, зокрема різну довжину, різний формат, частоту дискретизації, якість тощо. При розпізнаванні емоцій важливу роль відіграє однорідність даних, які передаються у модель і оскільки у датасеті з аудіо українською мовою присутні голоси з різною тональністю та самі аудіо мають різну гучність та довжину, задача класифікації попередньо зводиться до правильної попередньої обробки цих даних. Інформація передана через аудіо канал, тобто звуковими хвилями, має дуже різну природу, що створює неабиякий виклик для правильної обробки цих даних. Також важливою проблемою для задачі виявлення емоцій є неможливість проаналізувати семантичне значення сказаного. Можливим рішенням цієї проблеми може бути приведення даних з аудіо в текст. У подальшому це може бути одним із напрямків вдосконалення цієї роботи.

Також можливим поліпшенням результатів роботи має стати збір більшої та більш однорідної вибірки аудіо для навчання запрограмованих моделей. Варто подбати і про покращення попередньої обробки аудіо для вилучення однорідних ознак, які можуть надалі використовуватися для навчання моделей. Одними з варіантів покращень може слугувати використання технік зменшення розмірності аудіо із мінімальною втратою характеристик аудіо, а також використання технік нормалізації та аугментації для кращого виділення важливих ознак.

Слід зазначити, що, незважаючи на складності, класифікація емоцій з аудіо є перспективним напрямком, який може бути корисним у багатьох галузях, таких як оцінювання задоволеності клієнтів, розробка більш природніх штучних систем мовленнєвого взаємодії, психологічна та медична діагностика, тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Sound fundamentals — Isaac Computer Science [Електронний ресурс]. Режим доступу - https://isaaccomputerscience.org/concepts/data_rep_sound_fund?examBoard=all&stage=all&topic=sound_representation
2) Kong Q., Bayen A., Siauw T. Python Programming and Numerical Methods: A Guide for Engineers and Scientists. Elsevier Science & Technology, 2020.
3) Cooley J. W., Tukey J. W. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*. 1965. Т. 19, № 90. С. 297.
4) Anosov, Andrii & Проценко, М.М & Дубинко, О.Л & Павлунько, М.Я. Застосування вейвлет-перетворення для аналізу цифрових сигналів (2020).
5) The Wavelet Transform. An Introduction and Example | by Shawhin Talebi | Towards Data Science [Електронний ресурс]. Режим доступу - https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34
6) Spectrogram and the STFT — Introduction to Speech Processing [Електронний ресурс]. Режим доступу - https://speechprocessingbook.aalto.fi/Representations/Spectrogram_and_the_STFT.html
7) Bouguerra, Sara. (2016). Text Dependent Speaker Recognition Using HTK. 10.13140/RG.2.2.12773.45287.
8) Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America, 8*, 185–190.
9) Douglas O'Shaughnessy (1987). *Speech communication: human and machine*. Addison-Wesley. p. 150.
10) de Benito, Diego & Lozano-Diez, Alicia & Toledano, Doroteo & Gonzalez-Rodriguez, Joaquin. Exploring convolutional, recurrent, and hybrid deep

neural networks for speech and music detection in a large audio dataset. EURASIP Journal on Audio, Speech, and Music Processing (2019).

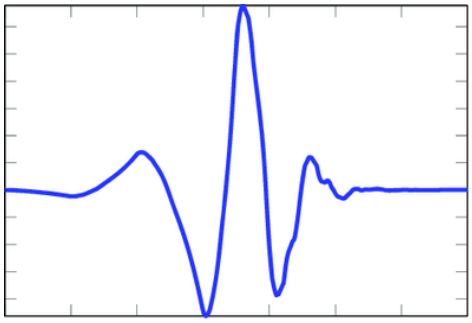
11) Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391.
12) FFMPEG Documentation [Електронний ресурс]. Режим доступу - https://ffmpeg.org/documentation.html.
13) PyTorch documentation — PyTorch 2.0 documentation [Електронний ресурс]. Режим доступу - https://pytorch.org/docs/stable/index.html.
14) Vryniotis, Vasilis. How to Train State-Of-The-Art Models Using TorchVision's Latest Primitives | PyTorch [Електронний ресурс]. Режим доступу - https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/.
15) Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
16) MIT/ast-finetuned-audioset-10-10-0.4593 · Hugging Face [Електронний ресурс]. Режим доступу - https://huggingface.co/MIT/ast-finetuned-audioset-10-10-0.4593

ДОДАТКИ

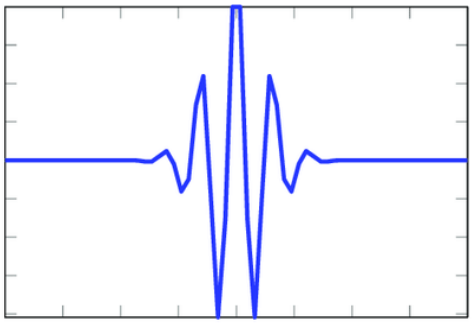
Додаток А

(обов’язковий)

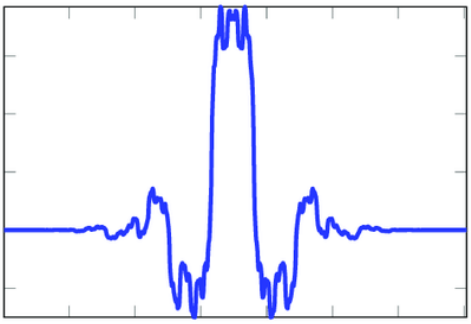
Приклади визначення вейвлетів різної форми [5]



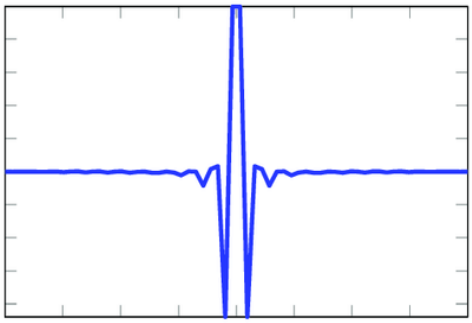
(a) db Wavelet



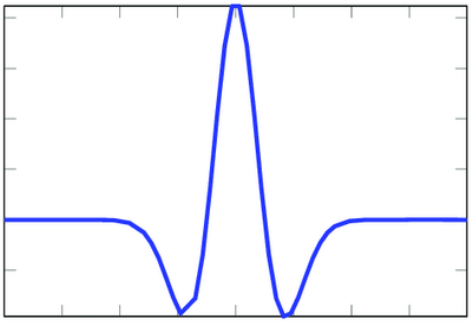
(b) Morlet Wavelet



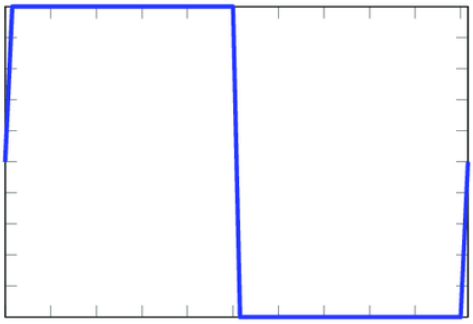
(c) Biorthogonal Wavelet



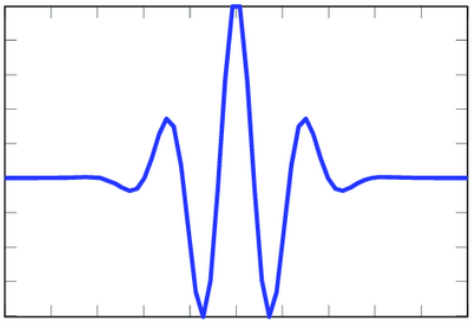
(d) Spline Wavelet



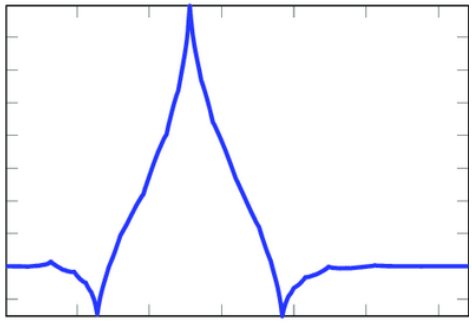
(e) Mexican Hat Wavelet



(f) Haar Wavelet



(g) Gaussian Wavelet



(h) Coiflet Wavelet

Додаток Б

(обов'язковий)

Лістинг коду для нарізки аудіо датасету кол-центру

```
import pandas as pd
import os
import re

directory_path = 'data/NLPproject'

def decideEmotion(string):
  if string[-1] == 'n':
    return 'neutral'
  if string[-1] == 'a':
    return 'negative'
  if string[-1] == 'h':
    return 'positive'
  return 'unknown'
if __name__ == '__main__':
  df = pd.DataFrame([], columns=['Name', 'Emotion'], )
  for filename in os.listdir(directory_path):
    f = os.path.join(directory_path, filename)
    if not os.path.isfile(f):
      for file in os.listdir(f):
        emotion_file = os.path.join(f, file)
        if '.txt' in emotion_file:
          file1 = open(emotion_file, 'r')
          lines = file1.readlines()


          for line in lines:
            numbers = re.findall(r'[\d\.]+', line)
            type = 'b'
            found_type = re.findall(r'(b|[a-z]_[a-z]_[a-z])', line)
            if len(found_type) > 0:
              type = found_type[0]
            if len(type) == 5:
              new_numbers = [number.replace('.','_') for number in numbers]
              emotion = decideEmotion(type)
              if (emotion == 'unknown'):
                continue
              result_file = 'company_dataset/' + file[:-4]+ new_numbers[0] +
new_numbers[1] + '.wav'
              print(result_file)
              os.system('ffmpeg -i {0} -ss {1} -to {2}
{3}'.format(emotion_file[:-3] +'wav', numbers[0], numbers[1],

'data/'+result_file ))
              try:
                open('data/'+result_file, 'r')
                df.loc[len(df)] = {"Name": result_file, "Emotion": emotion}
              except:
                print('errorr',result_file)
  df.to_csv('data/company_dataset.csv')
```

Додаток В

(обов’язковий)

Лістинг коду для нарізки аудіо для датасету студії звукозапису «Clan Kaizoku»

```
import pandas as pd
import os

tables = [
  'data/НП_01_Емоції.csv',
  'data/НП_05_Емоції.csv',
  'data/НП_10_Емоції.csv',
  'data/НП_13_Емоції.csv',
  'data/НП_14_Емоції.csv',
  'data/НП_15_Емоції.csv',
  'data/НП_19_Емоції.csv',
  'data/НП_20_Емоції.csv',
  'data/НП_22_Емоції.csv',
]

if __name__ == '__main__':
  result = pd.read_csv(tables[0], sep=';')
  result['New_name'] = result['Name'] + '_' + result['Start'].astype(str) + '_'
+ result['End'] + '_' + result['Emotion']
  result['New_name'] = result['New_name'].str.replace(r'(\.|:)', '_',
regex=True)
  result['New_name'] = 'my_converted/' + result['New_name'] + '.wav'
  for index, row in result.iterrows():
    os.system('ffmpeg -i data/{0} -ss {1} -to {2} {3}'.format(row['Name'] +
'.wav', row['Start'], row['End'],
                                                              'data/' +
row['New_name']))

  for (i, name) in enumerate(tables):
    if i == 0:
      continue
    df = pd.read_csv(name, sep=';')
    print(df.info())

    df['New_name'] = df['Name'] + '_' + df['Start'].astype(str) + '_' +
df['End'] + '_' + df['Emotion']
    df['New_name'] = df['New_name'].str.replace(r'(\.|:)', '_', regex=True)
    df['New_name'] = 'my_converted/' + df['New_name'] + '.wav'
    for index, row in df.iterrows():
      os.system('ffmpeg -i data/{0} -ss {1} -to {2}
{3}'.format(row['Name']+'.wav', row['Start'],row['End'], 'data/' +
row['New_name']))
    result = pd.concat([result, df], ignore_index=True)
  result.to_csv(f'data/cartoon.csv')
```

Додаток Г

(обов’язковий)

Лістинг коду для прибирання фонового шуму в аудіо

```
import torch
import torchaudio
import soundfile as sf
import os

directory_path = 'data/my_converted'
directory_path_noise = 'data/my_noise_converted'
import noisereduce as nr

if __name__ == '__main__':
  for filename in os.listdir(directory_path):
    data, rate = sf.read(os.path.join(directory_path,filename,))
    reduced_noise = nr.reduce_noise(data, sr=rate)
    reduced_noise = reduced_noise.astype('float32')
    signal = torch.from_numpy(reduced_noise)
    signal = signal.unsqueeze(0)
    torchaudio.save(os.path.join(directory_path_noise, filename), signal, rate)
```

Додаток Ґ

(обов'язковий)

Лістинг коду власної реалізації класу для завантаження даних датасету

```
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self,
                 annotations_file,
                 audio_dir,
                 transformation,
                 target_sample_rate,
                 num_samples,
                 device, is_copying=True):
        self.annotations = pd.read_csv(annotations_file)
        self.audio_dir = audio_dir
        self.device = device
        self.transformation = transformation.to(self.device)
        self.target_sample_rate = target_sample_rate
        self.num_samples = num_samples
        self.is_copying = is_copying

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, index):
        audio_sample_path = self._get_audio_sample_path(index)
        label = self._get_audio_sample_label(index)
        signal, sr = torchaudio.load(audio_sample_path)

        signal = signal.to(self.device)
        signal = self._resample_if_necessary(signal, self.target_sample_rate)
        signal = self._mix_down_if_necessary(signal)
        signal = self._cut_if_necessary(signal)
        if self.is_copying:
            signal = self._copy_if_necessary(signal)
        else:
            signal = self._right_pad_if_necessary(signal)
        signal = self.transformation(signal)
        return signal, label

    def _cut_if_necessary(self, signal):
        if signal.shape[1] > self.num_samples:
            signal = signal[:, :self.num_samples]
        return signal

    def _right_pad_if_necessary(self, signal):
        length_signal = signal.shape[1]
        if length_signal < self.num_samples:
            num_missing_samples = self.num_samples - length_signal
            last_dim_padding = (0, num_missing_samples)
            signal = torch.nn.functional.pad(signal, last_dim_padding)
        return signal

    def _resample_if_necessary(self, signal, sr):
        if sr != self.target_sample_rate:
            resampler = torchaudio.transforms.Resample(sr,
self.target_sample_rate)
            resampler = resampler.to(self.device)
            signal = resampler(signal)
        return signal

    def _mix_down_if_necessary(self, signal):
```

```
        if signal.shape[0] > 1:
            signal = torch.mean(signal, dim=0, keepdim=True)
        return signal

    def _get_audio_sample_path(self, index):
        path = os.path.join(self.audio_dir, self.annotations.iloc[
            index, 2])
        return path

    def _get_audio_sample_label(self, index):
        label = self.annotations.iloc[index, 3]
        label_tensor = torch.tensor(0)
        if label == 'neutral':
            label_tensor= torch.tensor(1)
        if label == 'negative':
            label_tensor= torch.tensor(2)
        return label_tensor

    def _copy_if_necessary(self, signal):
        if signal.shape[1] < self.num_samples:
            num_repeats = int(np.ceil(self.num_samples / signal.shape[1]))
            return signal.repeat(1, num_repeats)[:,:self.num_samples]
        return signal
```

ДОДАТОК Д

(обов’язковий)

Архітектура моделі ResNet50