МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ANDROID ЗАСТОСУНКУ РОЗПІЗНАВАННЯ ЕМОЦІЙ ЗА АУДІО»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.І. Носенко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент професор кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Носенку Миколі Івановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка Android застосунку розпізнавання емоцій за аудіо

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз та обробка аудіо даних.
2. Побудова нейронної мережі.
3. Розробка мобільного застосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2024	
2.	Збір вихідних даних.	22.01.2024	
3.	Обробка методичних та теоретичних	05.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	01.04.2024	
5.	Розробка третього розділу.	13.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	17.06.2024	

Студент _____ М.І. Носенко
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка Android застосунку розпізнавання емоцій за аудіо»: 42 с., 51 рис., 2 табл., 24 джерела, 2 додатки.

АУДІО, КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, МОБІЛЬНИЙ ПРИСТРІЙ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, ANDROID, JETPACK COMPOSE, KERAS, KOTLIN, PYTHON, TENSORFLOW, TENSORFLOW LITE.

Об’єкт дослідження – використання нейронних мереж в класифікації аудіо даних.

Мета роботи: розробка застосунку для класифікації емоцій за вхідним потоком аудіо даних.

Метод дослідження – методи моделювання, проєктування, конструювання та тестування програмного забезпечення, методи проєктування та моделювання нейронних мереж.

У цій роботі досліджується тема використання нейронних мереж для класифікації аудіо даних та їх інтеграція у мобільних застосунках. Під час виконання роботи було проведено аналіз предметної області, обрано архітектуру та реалізовано відповідну нейронну мережу з можливістю подальшої інтеграції у мобільному додатку. У ході реалізації мобільного додатку було створено модулі для обробки вхідного потоку аудіо даних та виокремлення необхідних ознак. В результаті роботи отримано програмне забезпечення для класифікації емоції за аудіо даними.

Нейронну мережу було реалізовано мовою програмування Python з використанням бібліотеки Keras у поєднанні з Tensorflow. Мобільний додаток реалізовано за допомогою нативних засобів Android та додаткових бібліотек.

SUMMARY

Bachelor’s qualifying paper «Development of an Android Application for Speech Emotion Recognition»: 42 pages, 51 figures, 2 tables, 24 references, 2 supplements.

AUDIO, CLASSIFICATION, MOBILE DEVICE, MACHINE LEARNING, NEURON NETWORKS, DATA PROCESSING, ANDROID, JETPACK COMPOSE, KOTLIN, PYTHON, TENSORFLOW, TENSORFLOW LITE

The object of the study is the use of neural networks in the classification of audio data.

The aim of the study is the development of an application for classification of emotions based on the input stream of audio data.

The methods of research are methods of modeling, designing, construction and testing of software, methods of designing and modeling neural networks.

This work explores the topic of using neural networks for the classification of audio data and their integration into mobile applications. During the execution of the work, the subject area was analyzed, the architecture was chosen and the appropriate neural network was implemented with the possibility of further integration in the mobile application. During the implementation of the mobile application, modules were created for processing the incoming stream of audio data and extracting the necessary features. As a result of the work, software was obtained for the classification of emotion based on audio data.

The neural network was implemented in the Python programming language using the Keras library in combination with Tensorflow. The mobile application is implemented using native Android tools and additional libraries.

ЗМІСТ

ВСТУП

Велика увага у сфері машинного навчання приділяється транскрибації натуральної мови. При цьому задачі розпізнавання емоції приділяється менше ресурсів.

Розпізнавання емоцій в аудіо (Speech emotion recognition) – це підполе афективних обчислень. Воно передбачає використання аудіо даних та виокремлення характерних ознак для виявлення людських емоцій. Із розвитком технологій та розширенням взаємодії типу людина-машина все більше областей застосування розпізнавання емоцій покращити досвід користувача. Наприклад, застосування подібних систем у кол-центрах для оцінки якості роботи спеціалістів за допомогою емоції клієнтів. Ще одним варіантом використання є домашні асистенти, що керуються за допомогою голосу. Залучення нейронної мережі, що буде аналізувати емоції користувача, може покращити взаємодію між користувачем та асистентом, що дозволить надати кращій зворотній зв’язок [1].

Важливим завданням розпізнавання емоцій в аудіо є вилучення ознак, що найбільше підходять для класифікації емоцій. Хоча людське мовлення містить велику кількість інформації, саме паралінгвістичні особливості будуть у центрі уваги цієї роботи. Паралінгвістичні особливості кількісно описують варіації у вимові мовних моделей. До них належать висота та інтенсивність мовлення, а також спектральні характеристики, наприклад, частотні кепстральні коефіцієнти мел.

В цій роботі будуть розглянуті такі проблеми:

- робота з аудіо даними;
- проєктування нейронної мережі;
- складність нейронної мережі та мобільні пристрої;
- розробка мобільного додатку та інтеграція в нього нейронного мережі.

1 АНАЛІЗ ТА ОБРОБКА АУДІО ДАНИХ

Для цієї роботи було використано такі набори даних: Toronto Emotional Speech Set (TESS) [2], CREMA-D [3], RAVDESS [4] SAVEE [5], EmoFilm [6], Canadian french emotional speech dataset (CAFE) [7], ASVP-ESD [8], MELD [9], Emotional Speech Dataset (ESD) [10] та Emov-DB [11]. Також було розширено набір даних EmoFilm новими спостереженнями українською мовою.

1.1 Аналіз аудіо даних

Аудіо аналіз – це процес вилучення інформації з аудіо для отримання ознак, що дозволять інтерпретувати ці дані. Аудіо дані – це репрезентація аналогового сигналу у цифровому вигляді. Для аналізу аудіо даних важливими є три ключові характеристики: період часу, амплітуда та частота.

Період часу – це те, скільки триває певний звук або, іншими словами, скільки секунд потрібно для завершення одного циклу коливань. Амплітуда – це інтенсивність звуку, виміряна в децибелах, яку ми сприймаємо як гучність. Частота, виміряна в герцах, показує, скільки звукових коливань відбувається за секунду. Люди сприймають частоту як низький або високий тон [12].

Під час перетворення аналогового сигналу у цифровий відбувається його дискретизація. Кількість вимірів за секунду визначає частоту дискретизації. Для роботи з аудіо даними мовою програмування Python існує бібліотека Librosa [13]. Побудуємо форму хвилі сигналу (рис. 1.1) та відобразимо один з аудіо файлів на графіку (див. рис. 1.2).

```
y, sr = librosa.load(emofilm_uk_examples.iloc[4].path, duration=2)
librosa.display.waveshow(y, sr)
```

Рисунок 1.1 – Код для побудови форми хвилі

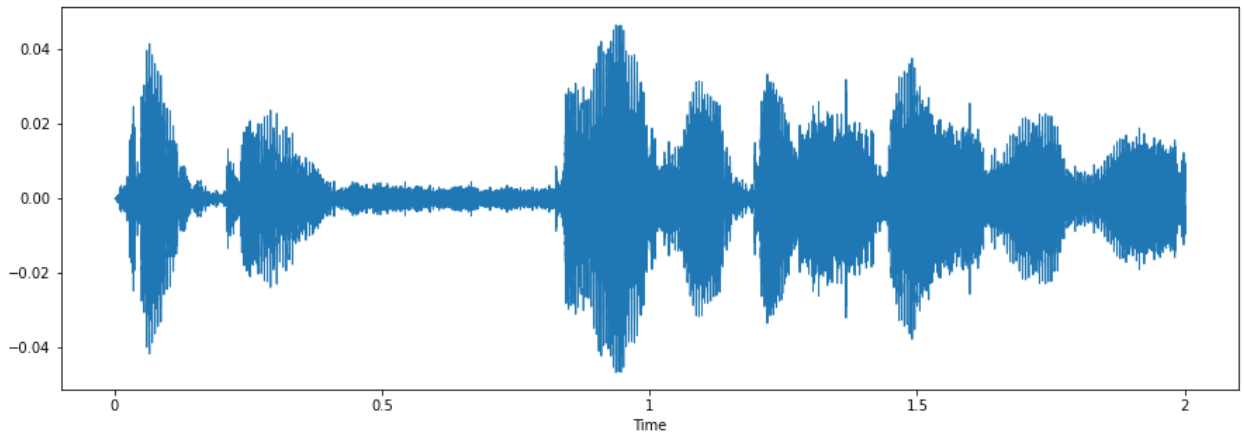

Рисунок 1.2 – Форма хвилі

Форма хвилі відображає час на горизонтальній осі та амплітуду на вертикальній осі, але вона не говорить нам, що відбувається з частотами.

Спектрограма – це детальне зображення сигналу, яке охоплює всі три характеристики звуку.

Розглянемо спектрограми двох емоції: злість та сум (див. рис. 1.3 – 1.8). Для візуалізації візьмемо файли із набору даних CREMA-D.

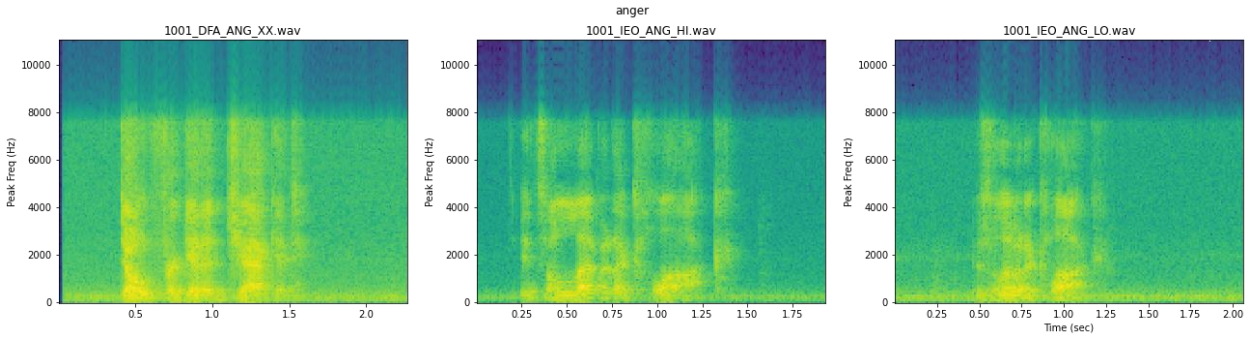

Рисунок 1.3 – Спектрограми емоції «Злість» англійською мовою

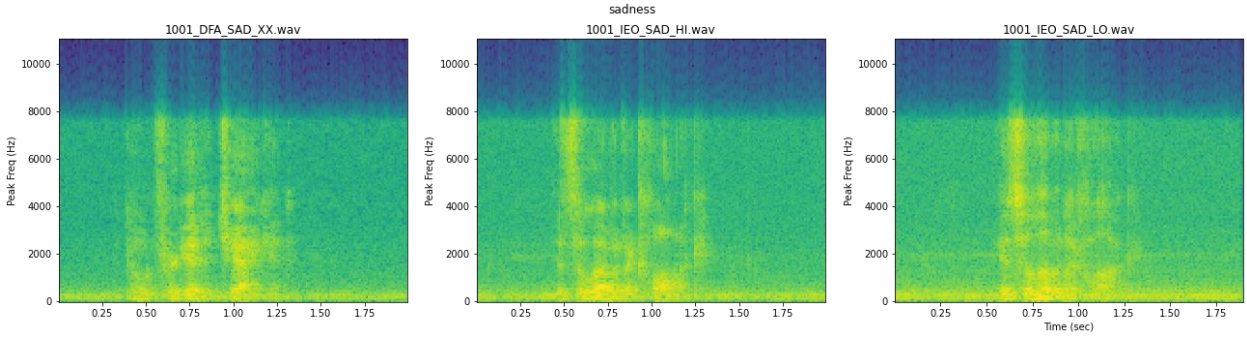

Рисунок 1.4 – Спектрограми емоції «Сум» англійською мовою

На спектрограмах, наведених вище (рис. 1.3, 1.4) бачимо, що файли емоції «злість» гучніші та мають більш виражені частоти у діапазоні 6000-8000 герц, за емоцію «сум». Окрім, цього, вони візуалізують файли, записані англомовними акторами, отже постає питання того, наскільки відрізняються спектрограми для різних мов. Щоб дізнатися це, побудуємо спектрограми відповідних емоції англійсько, італійською, іспанською та українською мовами (рис. 1.5 – 1.8). Для цього використаємо набір даних EmoFilm із доповненнями.

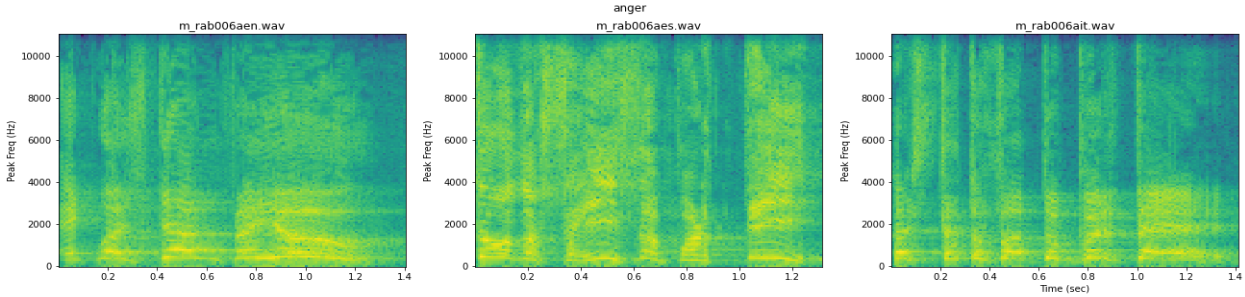

Рисунок 1.5 – Спектрограми емоції «Злість» англійською, італійською, іспанською

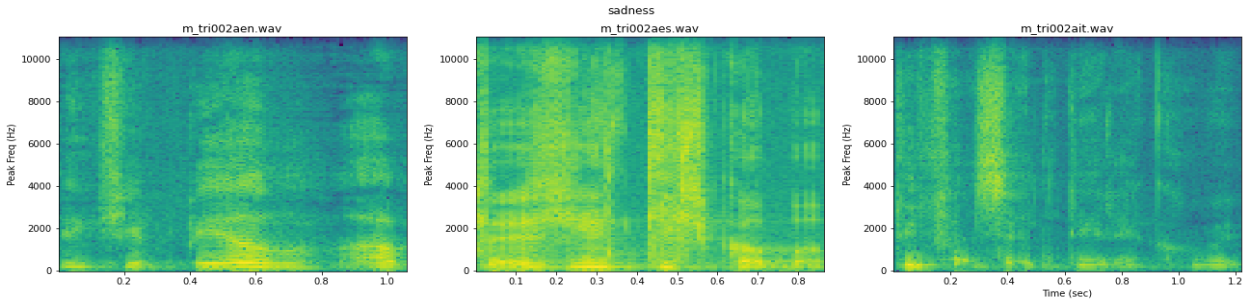

Рисунок 1.6 – Спектрограми емоції «Сум» англійською, італійською, іспанською

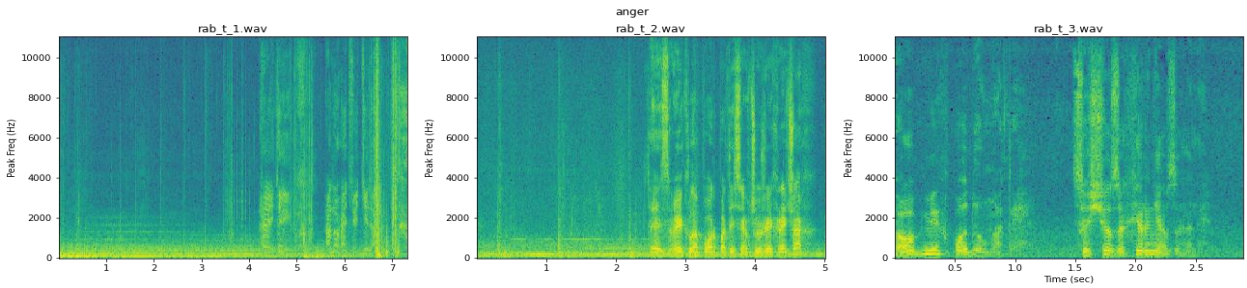

Рисунок 1.7 – Спектрограми емоції «Злість» українською

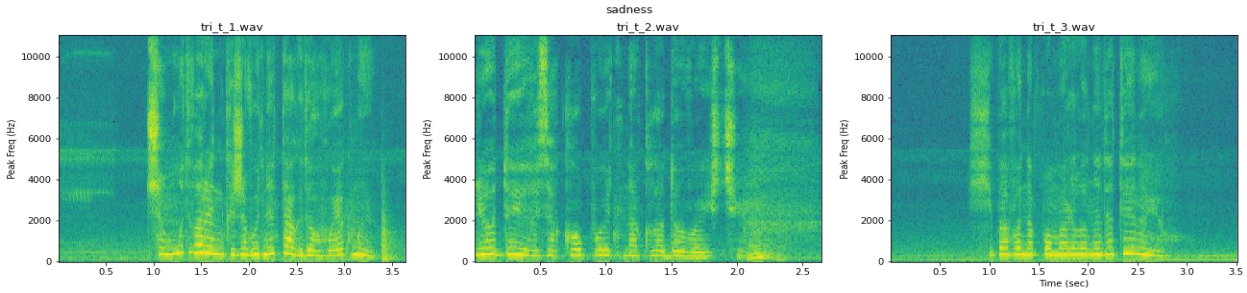

Рисунок 1.8 – Спектрограми емоції «Сум» українською

Бачимо, що немає чіткої відмінності між спектрограмами різних мов.

У методах аналізу аудіо також використовують статистичні характеристики, але в рамках цієї роботи обмежимося спектрограмами.

Також побудуємо гістограму (рис. 1.9) для того, щоб побачити розподіл спостережень за класами (рис. 1.10).

```
fig = plt.figure(figsize=(15, 5))
sns.histplot(data=complete, x='emotion')
plt.show()
```

Рисунок 1.9 – Код генерування гістограми

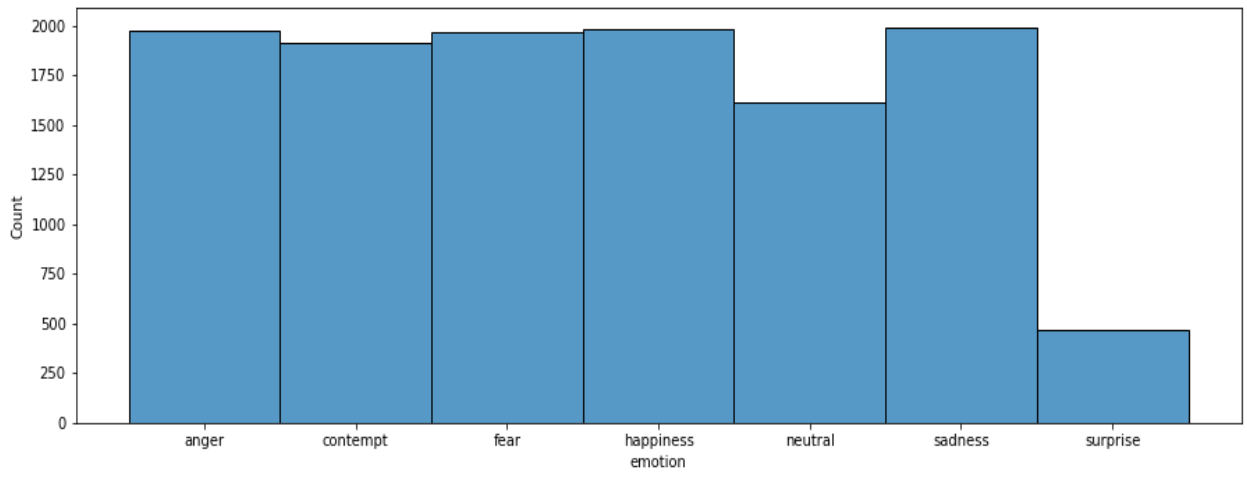

Рисунок 1.10 – Розподіл спостережень за класами

На рисунку вище бачимо, що єдина емоція, що містить значно менше спостережень – це «Здивування».

1.2 Обробка аудіо даних

Для того, щоб збільшити кількість спостережень до аудіо даних можливо застосувати певні техніки аугментації. Під час аугментації важливо розуміти, що саме буде робити нейронна мережа, тому що певні техніки значно змінюють дані та, в результаті, роблять неможливим їх використання.

Наступні функції зазвичай використовуються для аугментації аудіо сигналу [14]:

- розтягування часу;
- зсув часу;
- масштабування висоти звуку;
- ін’єкція шуму;
- інверсія полярності;
- зміщення висоти звуку.

Не кожна з цих функцій є доречною під час класифікації емоцій людини. Зміщення висоти звуку значно змінює його. Наприклад, голос у чоловіка нижчий, ніж у жінки. Якщо збільшити висоту звуку в записі голосу, чоловік може звучати як жінка. Для задачі розпізнавання емоцій така зміна може бути небезпечною.

Інверсія полярності передбачає перемикання значення амплітуди з позитивного на негативне і навпаки. Для більшості людей відтворення після інверсії полярності звучить так само, як оригінальне аудіо [14].

Як бачимо, зміщення висоти звуку є небезпечною технікою для поставленої задачі, у той час, як інверсія полярності не має негативних наслідків. Враховуючи це, для аугментації даних було використано зсув часу, ін’єкцію шуму та інверсію полярності.

Приклад функції аугментації даних наведено на рисунку 1.11.

```
def __get_features_mel(data, sample_rate, n_mels, n_fft, hop_length, augment):
    res1 = __get_mel_flat(data, sample_rate, n_mels, n_fft, hop_length)
    result = np.array(res1)
    if (augment):
        noise_data = __noise(data)
        res1 = __get_mel_flat(noise_data, sample_rate, n_mels, n_fft, hop_length)
        result = np.vstack((result, res1))
        invert_data = __invert_polarity(data)
        res1 = __get_mel_flat(invert_data, sample_rate, n_mels, n_fft, hop_length)
        result = np.vstack((result, res1))
        shifted_data = __shift(data)
        res1 = __get_mel_flat(shifted_data, sample_rate, n_mels, n_fft, hop_length)
        result = np.vstack((result, res1))
        return result
    result = np.reshape(result, (1, len(result)))
    return result
```

Рисунок 1.11 – Функція аугментації даних

Розповсюдженим методом візуалізації аудіо даних є використання мел-спектрограм. Мел-спектрограма – це візуальне представлення звуку, яке враховує те, як люди сприймають звукові частоти. Це різновид спектрограми, яка є більш загальним способом візуалізації звуку, показуючи, як амплітуда (гучність) звуку змінюється на різних частотах з часом. Мел-шкала є емпіричною шкалою, що ґрунтується на людському відчутті частоти звуку [15].

Бібліотека Librosa надає відповідний функціонал для отримання мел-спектрограм (рис. 1.12).

```
def __get_mel_flat(data, sample_rate, n_mels=64, n_fft=1024, hop_length=690):
    mel = librosa.feature.melspectrogram(
        y=data, sr=sample_rate, n_fft=n_fft, hop_length=hop_length, n_mels=n_mels)
    mel = librosa.power_to_db(mel, ref=np.max)
    return mel.flatten()
```

Рисунок 1.12 – Функція генерації мел-спектрограм

Побудуємо (рис. 1.13) зображення створених мел-спектрограм (див. рис. 1.14).

```
uk_samples = uk_scaled[7:10]


fig, axs = plt.subplots(1, len(uk_samples), figsize=(22, 5))
for i in range(len(uk_samples)):
        mfcc_flat = el.__get_mel_flat(uk_samples[i], 22050)
        mfcc = np.reshape(mfcc_flat, (64, 64))
        img = librosa.display.specshow(mfcc, y_axis='hz', ax=axs[i])
        fig.colorbar(img, ax=[axs[i]])


plt.show()
```

Рисунок 1.13 – Код для візуалізації мел-спектрограм

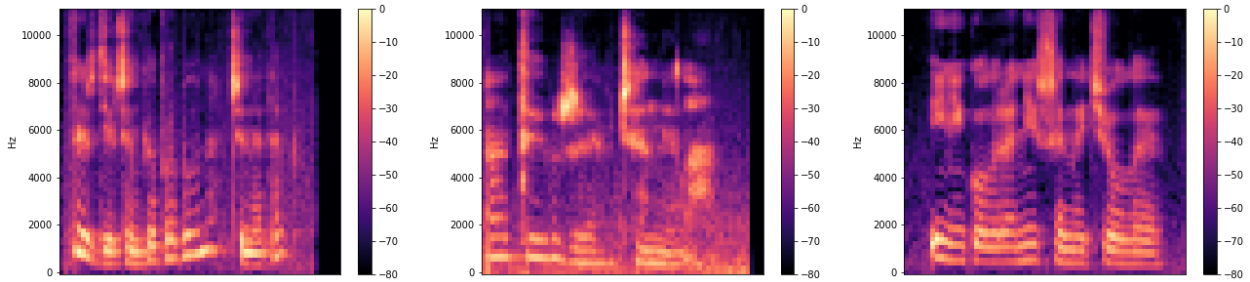

Рисунок 1.14 – Приклад створених мел-спектрограм

Отримані мел-спектрограми мають розмір 64×64 значення. Така розмірність мел-спектрограм обрана з урахуванням часу, що потрібен для їх генерації та об’ємом пам’яті, який вони займають. Зображення розмірністю 64×64 дозволяють використати адекватний розмір пачки для тренування нейронної мережі та не призведуть до помилок пов’язаних із переповненням відео та оперативної пам’яті.

2 ПОБУДОВА НЕЙРОННОЇ МЕРЕЖІ

2.1 Типові архітектури нейронних мереж

Ознаки, що були отримані під час обробки аудіо даних, являють собою зображення, отже логічним рішенням буде використання нейронних мереж, що активно використовуються для роботи з ними, а саме згорткові нейронні мережі. Згорткова нейронна мережа виокремлює ознаки з вхідного зображення, що потім передаються наступним шарам мережі. Ілюстрацію цього процесу можливо побачити на рисунку 2.1.

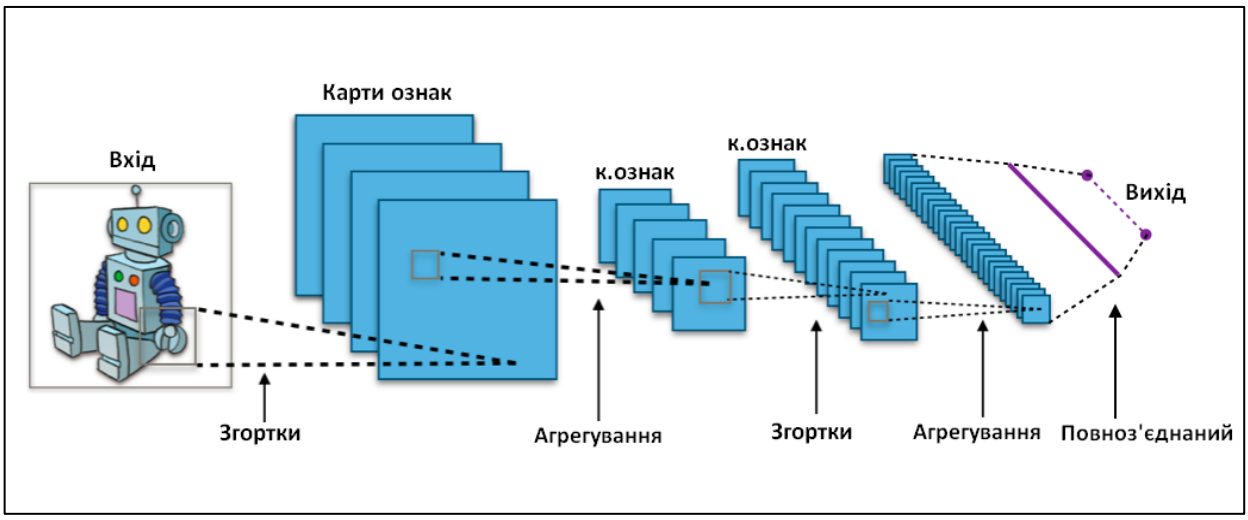

Рисунок 2.1 – Типова згорткова нейронна мережа [16]

Спектрограми зберігають у собі характеристику часу, отже доречним буде використання рекурентних нейронних мереж.

Поєднання цих двох архітектур може бути ефективним у задачі розпізнавання емоцій. У такій архітектурі згорткова нейронна мережа виокремлює просторові ознаки, а рекурентна часові ознаки мовлення [17]. При цьому, рекурентна мережа здатна до обробки довших послідовностей даних та виявлення закономірностей мовлення.

2.2 Створення нейронної мережі

Наразі існують дві популярні бібліотеки для машинного навчання: TensorFlow та PyTorch. TensorFlow – це бібліотека машинного навчання, що була розроблена компанією Google. PyTorch, у свою чергу, була розроблена компанією Meta. Певний час окремо існувала бібліотека Keras, що використовувала різні технології для побудови нейронних мереж, а для користувача надавала простий у використанні API. Зараз Keras та TensorFlow пов’язані між собою та в офіційній документації можливо дізнатися, що Keras має бути технологією за замовчуванням для більшості випадків [18]. Порівняння цих технологій наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння бібліотек мережевого навчання [18, 19, 20]

	Keras	TensorFlow	PyTorch
Мови програмування	Python	Python, C++	Python
Рівень абстракції	Високий	Декілька рівнів абстракції	Декілька рівнів абстракції
Середа розгортання	Хмара, сервер, мобільне середовище	Хмара, сервер, мобільне середовище	Хмара, сервер, мобільне середовище
Підтримка Android	У поєднанні з TensorFlow	Так	Так
Сфера застосування	Розробка	Розробка та дослідження	Дослідження
Налогодження	Обмежене	Широкий спектр можливостей	Широкий спектр можливостей

Із наведеної вище таблиці бачимо, що кожна з бібліотек дозволить реалізувати запропоновану архітектуру та не має явних недоліків. Отже, для

реалізації було використано платформу TensorFlow із API високого рівня Keras. З Keras надає повний доступ до масштабованості та кросплатформних можливостей TensorFlow, можливості використання TPU або GPU, а також експортування моделі для запуску на мобільних пристроях. Архітектура побудованої мережі зображена на рисунку 2.2.

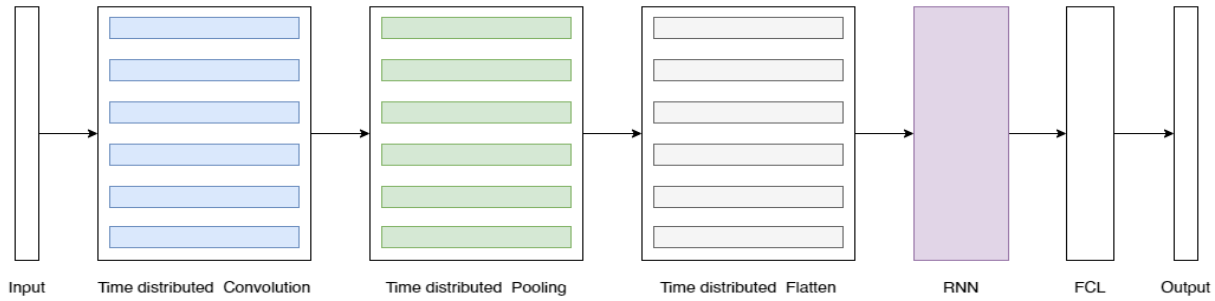

Рисунок 2.2 – Архітектура нейронної мережі

Реалізацію цієї архітектури наведено на рисунку 2.3.

```
def get_model(name = None):
    input_shape = (64, 64, 1)
    emotions_number = train_basic[1].shape[1]
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(TimeDistributed(Conv1D(16, 5, padding="same", activation='relu',
kernel_initializer=initializers.HeUniform(seed=1))))
    model.add(TimeDistributed(AveragePooling1D(3)))
    model.add(TimeDistributed(Flatten()))
    model.add(GRU(64, return_sequences=False))
    model.add(Dense(emotions_number, activation="softmax"))
    if name:
        model._name = name
    return model
```

Рисунок 2.3 – Код створення нейронної мережі

Використання TimeDistributed дозволяє засовувати цільовий шар для кожного часового проміжку вхідних даних При цьому вхідні дані розмірністю

64×64×1 будуть розбиті на 64 часові проміжки такої розмірності: 1×64×1. Кожен з часових проміжків буде окремо оброблений та переданий у рекурентний шар. Розглянемо детальніше інтерпретацію часового проміжку (рис. 2.4).

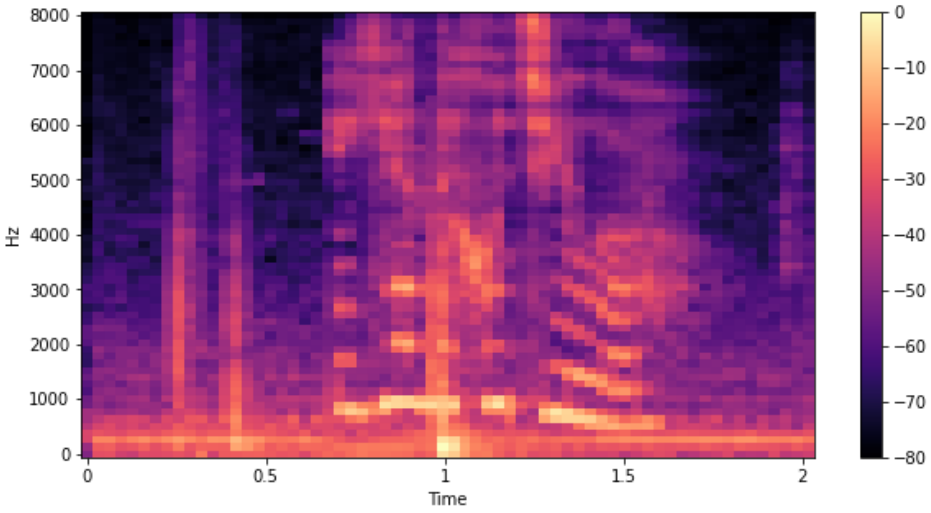

Рисунок 2.4 – Звичайна мел-спектрограма

На рисунку 2.4 бачимо, що змінна часу знаходиться на вісі абсцис. У цьому випадку розбиття даних обгорткою TimeDistributed призведе до того, що у шари згортки потраплятимуть значення, які містять такий сенс – зміни звуку певної частоти з часом. Транспонуємо вхідні дані та знову побудуємо мел-спектрограму (див. рис. 2.5).

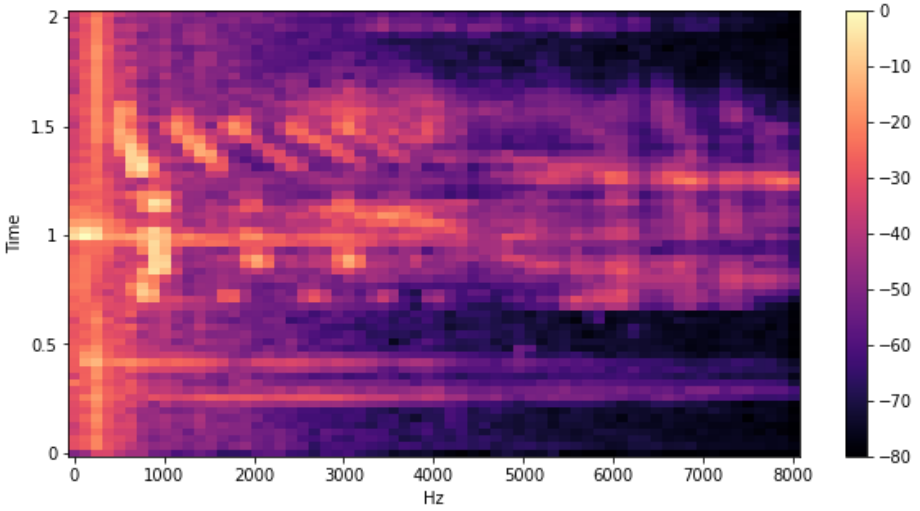

Рисунок 2.5 – Транспонована мел-спектрограма

У цьому випадку бачимо, що змінна часу знаходиться на вісі ординат. Тепер кожен часовий проміжок містить інше значення: зміни у частотах звуку в певний момент часу. Отже, доречним буде транспонувати мел-спектрограми.

2.3 Результати роботи створеної мережі

Важливим етапом роботи є аналіз точності отриманої моделі. Побудуємо графіки навчання (див. рис. 2.6).

На рисунку 2.6 бачимо графіки зміни функції втрат та значення точності протягом навчання. Для тестування точності моделі використовуються дані, що не були використані під час навчання. Це дозволяє дати адекватну оцінку моделі.

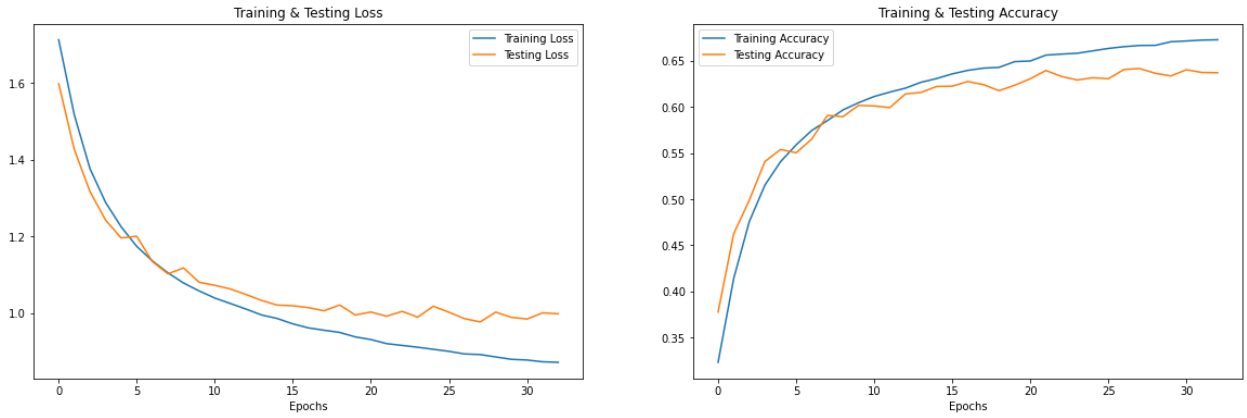

Рисунок 2.6 – Графіки навчання

За допомогою матриці помилок (див. рис. 2.7) можливо зіставити реальні значення із прогнозованими та наочно продемонструвати невідповідність між ними.

Також велику кількість інформації містить звіт класифікації (див. рис. 2.8). У ньому присутні дані про значення повноти (Recall), влучності (Precision), точності (Accuracy) та F-міри (f1-score). Ці метрики дозволяють отримати більш детальну інформацію про результати роботи моделі.

F-міра створеної моделі дорівнює 0.62. F-міра може містити значення від 0 до 1, де 1 – це повна влучність. Такий результат можливо пояснити кількістю різних наборів даних, що були використані для навчання. Велика кількість різноманітних даних дозволяє зробити більш узагальнену модель, що в умовах реального використання працюватиме краще за навчену лише на одному наборі даних.

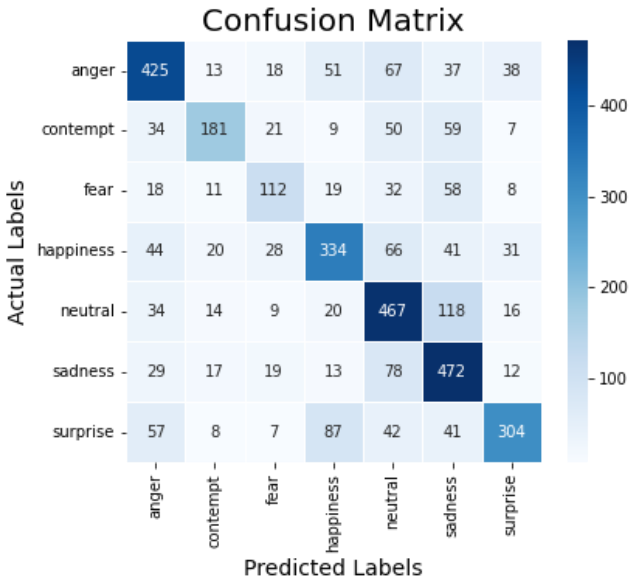

Рисунок 2.7 – Матриця помилок

	precision	recall	f1-score	support
anger	0.66	0.65	0.66	649
contempt	0.69	0.50	0.58	361
fear	0.52	0.43	0.47	258
happiness	0.63	0.59	0.61	564
neutral	0.58	0.69	0.63	678
sadness	0.57	0.74	0.64	640
surprise	0.73	0.56	0.63	546
accuracy			0.62	3696
macro avg	0.63	0.60	0.60	3696
weighted avg	0.63	0.62	0.62	3696

Рисунок 2.8 – Звіт з класифікації

У випадку використання лише набору даних ESD, значення F-міри підіймається до 0.78 (див. рис. 2.9) без буд-яких змін у мережі, що враховували

б особливості цих даних. Але при такому підході реакція моделі на нові дані занадто непередбачувана.

	precision	recall	f1-score	support
anger	0.86	0.83	0.85	343
happiness	0.73	0.69	0.71	365
neutral	0.78	0.78	0.78	364
sadness	0.76	0.90	0.82	348
surprise	0.80	0.72	0.76	330
accuracy			0.78	1750
macro avg	0.79	0.78	0.78	1750
weighted avg	0.78	0.78	0.78	1750

Рисунок 2.9 – Звіт з класифікації для набору даних ESD

У розділі 2.2 була розглянута важливість транспонування даних. Перевіримо це емпірично: проведемо навчання моделі на звичайних, транспонованих та транспонованих даних, що були додатково записані у зворотному порядку. У таблиці 2.2 бачимо реакцію моделі на різні вхідні дані. В назві моделі відображено перетворення даних, які були використані для навчання. Дані, що не були транспоновані, показують гірші результати, що підтверджує важливість цього перетворення.

Таблиця 2.2 – Вплив транспонування даних на точність моделі

Модель	Повнота (Recall)	Влучність (Precision)	Точність (Accuracy)	F-міра (f1-score)
cnn-gru-v9-seq-scale-transpose-16khz	0.627165	0.627165	0.627165	0.627165
cnn-gru-v9-seq-scale-transpose-flip-16khz	0.632846	0.632846	0.632846	0.632846
cnn-gru-v9-seq-scale-16khz	0.555465	0.555465	0.555465	0.555465

Посилання на повну реалізацію нейронної мережі наведено у додатку А.

3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Загальна структура додатку

Для реалізації додатку було використано архітектурний шаблон Model–view–viewmodel (MVVM) для полегшення роботи зі станами. Шаблон MVVM (рис. 3.1) розділяє розробку графічного інтерфейсу (представлення) від бізнес-логіки (моделі), подібно до шаблону MVC. Модель представлення (viewmodel) слугує посередником, перетворюючи дані з моделі у формат, зручний для подальшого використання та відображення. Іншими словами, модель представлення ближча до моделі, ніж до представлення. Вона обробляє більшу частину, якщо не всю, логіку, пов’язану з візуалізацією даних.

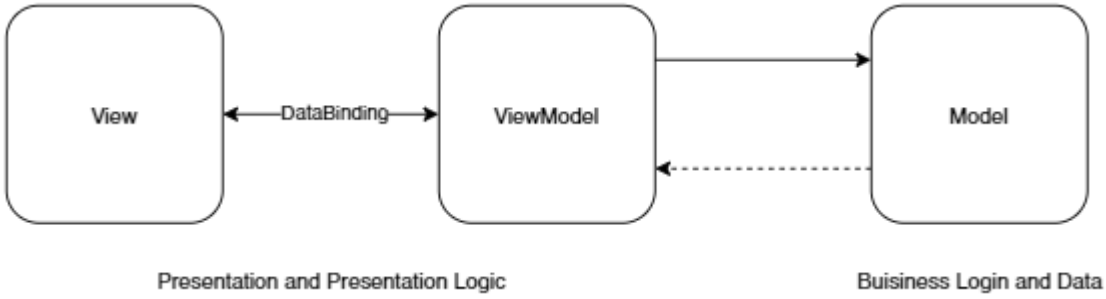

Рисунок 3.1 – Шаблон MVVM

Загальну архітектуру мобільного додатку зображено на рисунку 3.2. Вхідними даними є людське мовлення, записані аудіо дані використовуються для генерування ознак. Ці ознаки передаються у класифікатор. Результатом класифікації є одна з емоції. Можливі прецеденти зображені на рисунку 3.3.

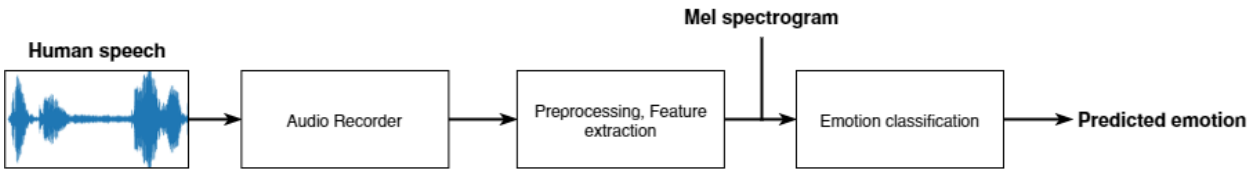

Рисунок 3.2 – Архітектура мобільного додатку

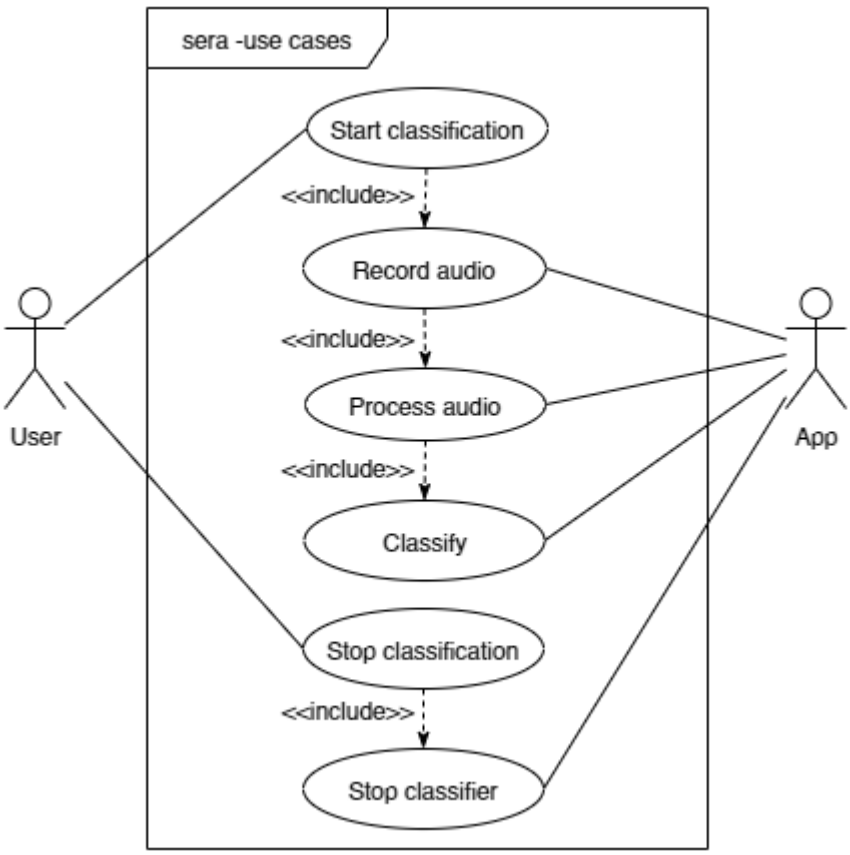

Рисунок 3.3 – Діаграма прецедентів

Для того, щоб інтегрувати створену нейронну мережу у мобільний додаток, використаємо Tensorflow Lite. Tensorflow Lite – це набір інструментів, що дозволяє запускати створені моделі на мобільних, вбудованих і периферійних пристроях [21]. Для цього необхідно перетворити попередньо створену нейронну мережу перетворити на tflite модель. Для полегшення роботи з моделлю доречно додати до неї метадані – це дозволить автоматично згенерувати код оболонки TensorFlow Lite.

Файли моделей TensorFlow – це програмний код, а бібліотеки, що дають можливість використовувати їх, надають лише інтерпретатор, але не генерують код моделі, тобто кожна модель є окремим, незалежним від інших, класом. З цього випливає питання узагальнення моделей у спільний клас або використання спільного інтерфейсу. Побудуємо такий інтерфейс навколо вхідних даних моделі (див. рис. 3.4).

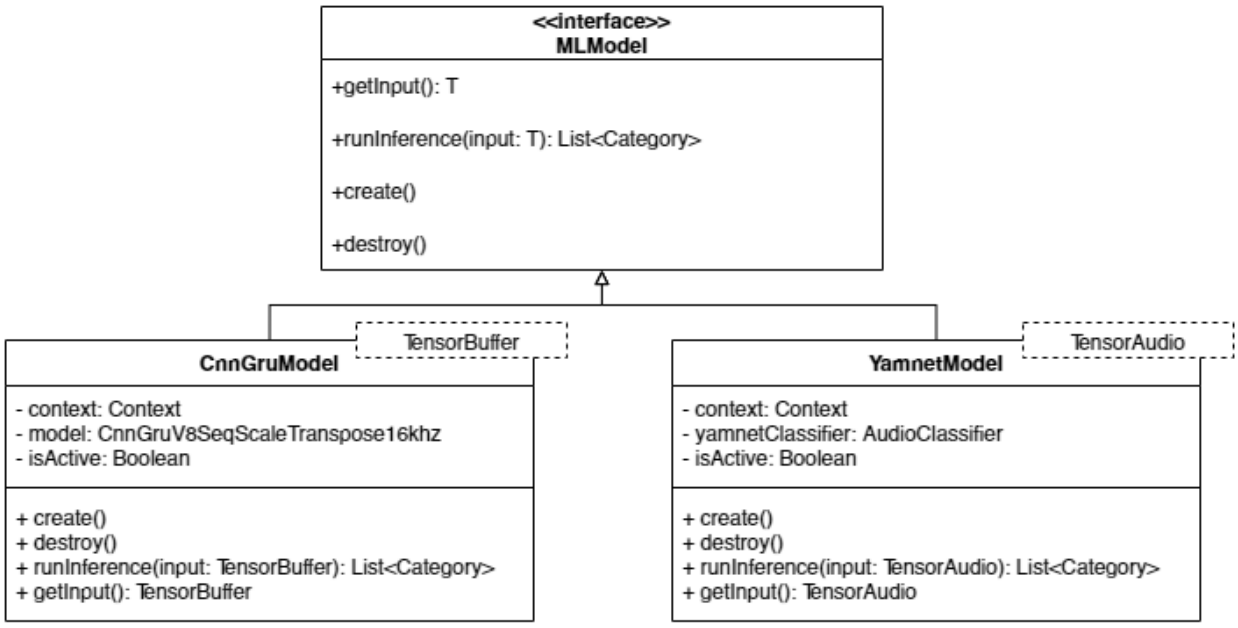

Рисунок 3.4 – Інтерфейс MLModel та його імплементації

На рисунку 3.4 бачимо, як за допомогою узагальнення можливо оперувати різними типами даних. Цей підхід дозволяє об’єднати різні моделі в певну ієрархію та використовувати спільний інтерфейс а не залежати від конкретної реалізації. Завдяки цьому та наявності метаданих у моделі, її запуск стає тривіальним (рис. 3.5). Структура додатку наведена на рисунку 3.6.

```
private lateinit var model: CnnGruV9SeqScaleTranspose16khz
override fun create() {
    …
    model = CnnGruV9SeqScaleTranspose16khz.newInstance(context)
}
override fun runInference(input: TensorBuffer): List<Category> {
    ...
    return model.process(input).probabilityAsCategoryList
}
```

Рисунок 3.5 – Код виклику tflite моделі

Інтерфейс користувача було створено за допомогою Jetpack Compose. Jetpack Compose – це сучасний фреймворк від Google для створення

інтерфейсів користувача Android-додатків на мові Kotlin. Він пропонує декларативний підхід до UI-розробки, що робить код більш лаконічним, читабельним та зручним для підтримки. Створимо макет додатку з урахуванням варіантів його використання (див. рис. 3.7).

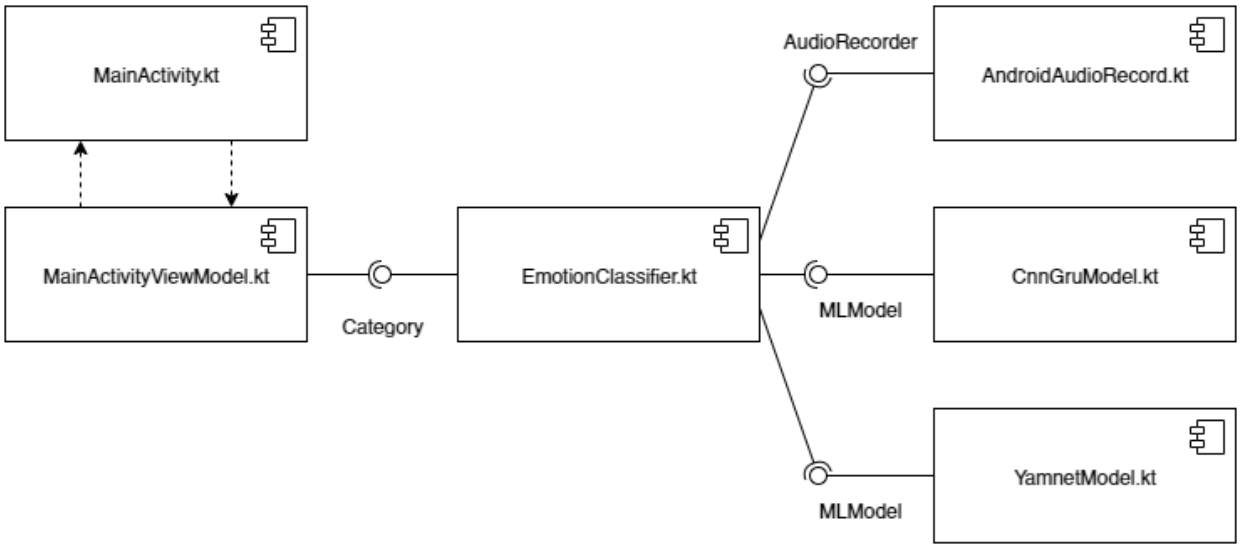

Рисунок 3.6 – Діаграма компонентів мобільного додатку

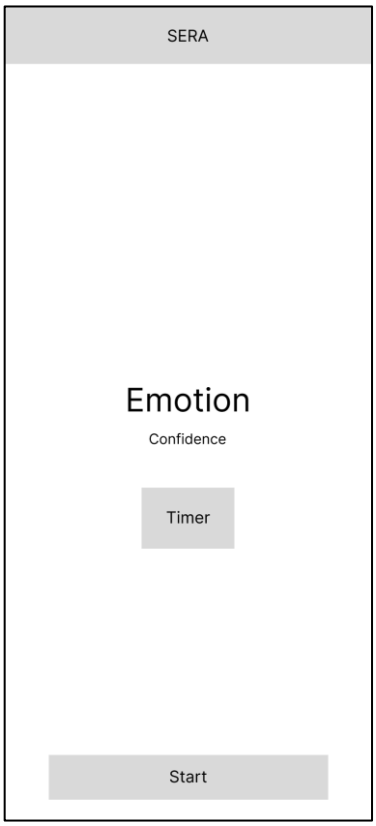

Рисунок 3.7 – Макет мобільного додатку

Для запису аудіо було використано AudioRecord. AudioRecord – це платформа для запису аудіоданих на пристрої Android. AudioRecord усуває шар абстракції між програмою та апаратним забезпеченням пристрою, записуючи нестиснений звук без можливості запису безпосередньо у файл. Це API є аудіосистемою найнижчого рівня для Android, яку все ще можна використовувати на рівні Java та Kotlin [22].

3.2 Підготовка аудіо даних в Android

Під час розробки даної нейронної мережі була використана бібліотека Librosa. Для того, щоб забезпечити сумісність результатів виокремлення ознак, у мобільному застосунку використаємо бібліотеку JLibrosa [23]. Ця бібліотека була створена для використання у Java машині, отже є цілком сумісною з операційною системою Android.

Для навчання моделі було використано мел-спектрограми з розмірністю 64×64 значення, отже у мобільному застосунку необхідно обробити вхідний аудіо сигнал таким чином, щоб отримати аналогічні дані. Для цього скористаємося методом generateMelSpectroGram (рис. 3.8) із бібліотеки JLibrosa.

```
fun extractMelToBuffer(yValues: FloatArray, sr: Int, nFFT: Int = 1024, nMels: Int = 64,
hopLength: Int = 690, scale: Boolean = false, transpose: Boolean = false): ByteBuffer {
    val data = jLibrosa.generateMelSpectroGram(yValues, sr, nFFT, nMels, hopLength)
    var dataDb = powerToDB(data, ref = data.maxValue())
    if (scale) { dataDb = dataDb.minMaxScaleRespectColumns(-1, 1) }
    if (transpose) { dataDb = dataDb.transpose() }
    return dataToByteBuffer(dataDb)
}
```

Рисунок 3.8 – Метод для створення мел-спектрограм

На рисунку 3.8 бачимо реалізацію методу, що генерує мел-спектрограми з масиву вхідних даних, трансформує силу сигналу у децибели, нормалізує та транспонує дані. В результаті мел-спектрограма записується у буфер та використовується для класифікації.

3.3 Запис аудіо та класифікація

Як вже було зазначено у розділі 3.1, для запису аудіо було використано AudioRecord, але для того, щоб не залежати саме від цього класу, було створено інтерфейс AudioRecorder (рис. 3.9).

```
interface AudioRecorder {
    fun start()
    fun stop()
    fun readData(): FloatArray
    fun getState(): Int
}
```

Рисунок 3.9 – Інтерфейс AudioRecorder

AndroidAudioRecorder – клас, що реалізує інтерфейс AudioRecorder. Для отримання аудіо з об’єкта AudioRecord необхідно періодично читати з нього дані. Кожен раз, після отримання даних, буфер оновлюється. Зчитування даних за останні дві секунди запису аудіо значно знижує можливість моделі реагувати на зміни емоційного забарвлення мовлення, отже наповнюватимемо змінну, що зберігає зчитані дані, послідовно (див. рис. 3.10). Для цього використаємо структуру даних черга. Це дозволить легко видаляти старі дані та додавати щойно зчитані.

В мові програмування Kotlin черга реалізована за допомогою стандартного класу ArrayDeque.

```
override fun readData(): FloatArray {
    …
    val newData = FloatArray(recorder!!.channelCount * recorder!!.bufferSizeInFrames)
    val loadedValues = recorder!!.read(newData, 0, newData.size,
AudioRecord.READ_NON_BLOCKING)
    if (loadedValues <= 0) return floatArrayOf()
    if (queue.isEmpty()){
        queue.addAll(newData.toTypedArray())
        while (queue.size < newData.size){ queue.add(0f) }
    } else{
        queue.addAll(newData.toTypedArray())
        while (queue.size > newData.size){ queue.removeFirst()}
    }
    return queue.toFloatArray()
}
```

Рисунок 3.10 – Реалізація методу читання даних з AudioRecord

Класифікація відбувається в класі EmotionClassifier, що імплементує інтерфейс Classifier (рис. 3.11). Для користувачів класів, що реалізують цей інтерфейс необхідними є лише можливості почати та закінчити класифікацію. При цьому в процесі класифікації потрібно постійно передавати результати роботи класифікатора.

```
interface Classifier {
    fun start(): Flow<ClassificationResult<List<Category>>>
    fun stop()
}
```

Рисунок 3.11 – Інтерфейс Classifier

Розглянемо фрагмент реалізації методу start (див. рис. 3.12). Важливим компонентом цього методу є асинхронне читання даних та їх класифікація. Мова програмування Kotlin надає таку можливість за допомогою корутін

(Coroutine). Корутіни дозволяють виконувати програмний код асинхронно, а потік даних Flow робить можливим віддавати проміжні дані колектору.

```
override fun start(): Flow<ClassificationResult<List<Category>>> {
    audioRecorder.start()
    return flow {
        …
        delay(warmUpIntervalMs)
        while (currentCoroutineContext().isActive) {
            val newValues: FloatArray = audioRecorder.readData()
            …
            val processedData = featureExtractor.extractMelToBuffer(newValues,
audioRecorder.getSampleRate(), hopLength = 502, scale = true, transpose = true)
            mel.loadBuffer(processedData)
            val probabilities = serModel.runInference(mel)
            emit(ClassificationResult.Success<List<Category>>(probabilities))
            delay(classificationIntervalMs)
        }
    }.flowOn(Dispatchers.IO)
}
```

Рисунок 3.12 – Фрагмент реалізації методу start

На рисунку 3.12 бачимо приклад використання екземплярів класів AndroidAudioRecorder та JLibrosaExtractor. Також бачимо, наскільки простим є використання збереженої моделі.

Отримані, що передаються з потоку будуть оброблені в MainActivityViewModel, а саме в методі startClassification (див. рис. 3.14). Дані із потоку передаються в обгортці Classification Result (рис. 3.13). Ця обгортка є класом типу sealed. Цей тип класів є специфічним для мови програмування Kotlin. Ці мають особливість: усі нащадки цього класу відомі під час компіляції. Це дозволяє безпечно обробити дані, що надходять із потоку. У даному випадку, результатом класифікації може бути Success або Error.

```
sealed class ClassificationResult<T>(val data: T? = null, val message: String? = null) {
    class Success<T>(data: T) : ClassificationResult<T>(data)
    class Error<T>(message: String, data: T? = null) : ClassificationResult<T>(data, message)
}
```

Рисунок 3.13 – Можливі результати класифікації

Приклад обробки даних у методі startClassification на рисунку 3.14.

```
fun startClassification() {
    …
    classifier.start()
        .onEach { result ->
            when (result) {
                is ClassificationResult.Error -> {
                    _error.value = result.message ?: "Oops... Something went wrong!"
                }
                is ClassificationResult.Success -> {
                    _categories.value = result.data?.map { it.toEmotion() }.orEmpty()
                    filterCategories()
                }
            }
        }
        .launchIn(viewModelScope)
}
```

Рисунок 3.14 – Фрагмент обробки результатів роботи моделі

Посилання на повну реалізацію мобільного застосунку наведено у додатку А. Приклади роботи застосунку зображені у додатку Б.

3.4 Перевірка вхідних даних

Розбіжність під час навчання та роботи – це різниця між продуктивністю під час тренування та продуктивністю під час реального використання

нейронної мережі. Причиною цього може бути невідповідність між тим, як обробляються дані в умовах навчання та у середі розгортання [24]. У даному випадку використовується дві бібліотеки для отриманна мел-спектрограм: Jlibrosa та Librosa. Створимо набір мел-спектрограм за допомогою JLibrosaExtractor (рис. 3.15) та Librosa (рис. 3.16). Після цього перевіримо, наскільки вони подібні.

```
fun main() {
    val extractor = JlibrosaExtractor()
    File("audio").walkTopDown().forEach {
        val audioData = extractor.readFile(it.path, 22050, 2)
        val spectrogram = extractor.extractMel(audioData, 22050, 1024, 64, 690)
        val arr = mk.ndarray(spectrogram)
        mk.writeCSV(path = "D:\\Programming\\jlibrosa-mel-audio-test\\features-
jlibrosa\\${it.name.dropLast(4)}.csv", ndarray = arr)
    }
}
```

Рисунок 3.15 – Код генерації мел-спектрограм за допомогою JlibrosaExtractor

```
def __get_mel_flat(data, sample_rate, n_mels=64, n_fft=1024, hop_length=690):
    mel = librosa.feature.melspectrogram(
        y=data, sr=sample_rate, n_fft=n_fft, hop_length=hop_length, n_mels=n_mels)
    mel = librosa.power_to_db(mel, ref=np.max)
    return mel


mels_l = []
for a in audio_data:
    mel = __get_mel_flat(a, 22050)
    mels_l.append(mel)
```

Рисунок 3.16 – Код генерації мел-спектрограм за допомогою Librosa

Схожість двох мел-спектрограм описує, наскільки вони подібні одна до одної. Її можна обчислити за допомогою формули, яка враховує різницю між

спектрограмами та нормалізує значення від 0 до 1. Після цього отримана значення переводяться у відсотки. 100% схожість означає, що спектрограми ідентичні, а 0% – що вони абсолютно різні.

$$S = \left(1 - \frac{1}{n}\sum_{i=1}^{n}\frac{|A_t - A_p|}{D}\right) * 100,$$

де A_t – реальне значення, A_t – прогнозоване значення,

D – максимально можлива абсолютна різниця.

Реалізацію цієї функції наведено на рисунку 3.17.

```
def compare_ndarrays(arr1, arr2):
    if not np.array_equal(arr1.shape, arr2.shape):
        raise ValueError("Arrays must have the same shape.")
    abs_diff = np.abs(arr1 - arr2)
    mean_abs_diff = np.mean(abs_diff)
    max_abs_diff = np.max(arr1) - np.min(arr1)
    similarity = 1 - (mean_abs_diff / max_abs_diff)
    similarity_percent = similarity * 100
    return similarity_percent
```

Рисунок 3.17 – Код порівняння двох мел-спектрограм

Результат порівняння створених спектрограм наведено на рисунку 3.18.

```
Similarity between arrays: 99.93%
```

Рисунок 3.18 – Результат двох мел-спектрограм

Додатково візуалізуємо мел-спектрограми, створені за допомогою JLibrosaExtractor та Librosa (див. рис. 3.19 та 3.20).

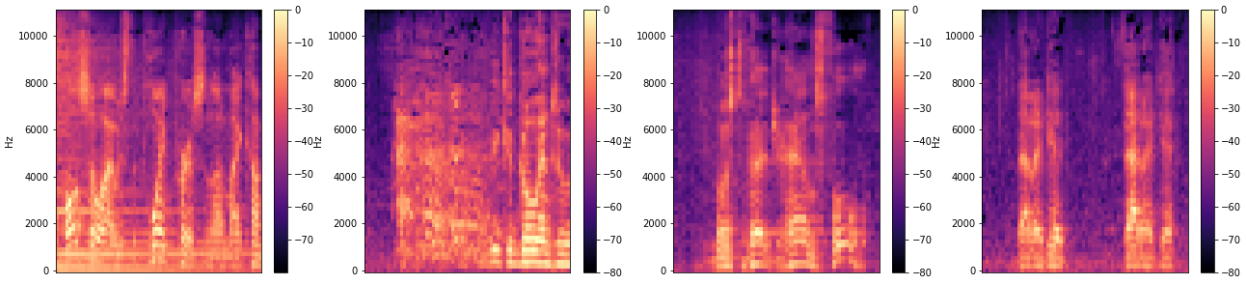

Рисунок 3.19 – Мел-спектрограми, створені за допомогою JLibrosaExtractor

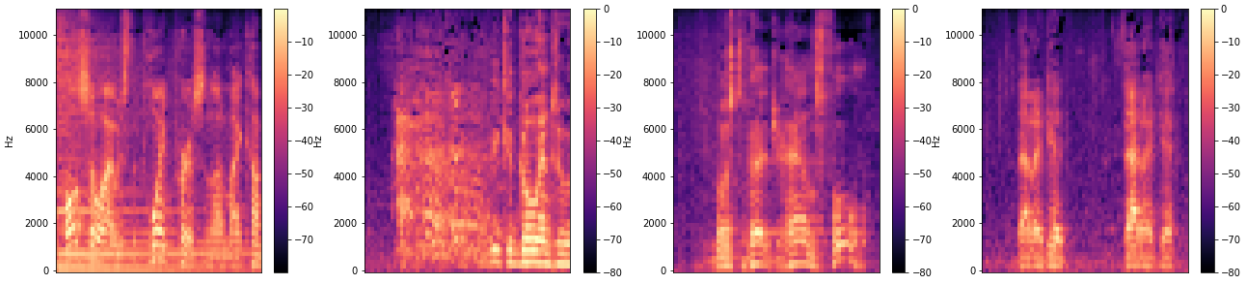

Рисунок 3.20 – Мел-спектрограми, створені за допомогою Librosa

З наведених прикладів бачимо, що за допомогою бібліотеки JLibrosa дійсно можливо згенерувати мел-спектрограми, що відповідають даним для навчання.

Окрім виокремлення ознак, важливим етапом обробки даних є перетворення отриманих даних у децибели, нормалізація та транспонування матриці значень. Для цього було створено та перевірено відповідні утиліти (рис. 3.21).

Test class ArrayUtilsTest		72 ms
min max scaling with respect to column should scale data by column	passed	42 ms
testMaxValueByRow	passed	15 ms
max value by column should return maximum value	passed	1 ms
minValue	passed	1 ms
min max scaling with respect to column should scale in range between -1 to 1	passed	3 ms
min value by column should return minimum value	passed	1 ms
min max scaling with respect to column should not return NaN values	passed	1 ms
min max scaling should scale data globally	passed	0 ms
testMinValueByColumn	passed	0 ms
testMaxValueByColumn	passed	1 ms
testMinValueByRow	passed	2 ms
test transposed matrix	passed	3 ms
maxValue	passed	2 ms

Рисунок 3.21 – Результат тестування ArrayUtils

ВИСНОВКИ

Таким чином, під час виконання кваліфікаційної роботи було досліджено методи та технології для роботи з аудіо даними та розроблено мобільний додаток для розпізнавання емоцій за допомогою аудіо. Процес розпізнавання включає в себе запис вхідних аудіо даних, виокремлення ознак з потоку аудіо та передача їх нейронній мережі. При цьому використані технології дозволяють розгорнути створену нейронну мережу не лише у мобільному застосунку.

У першому розділі було досліджено поняття аудіо сигналу, методи його практичного дослідження та способи обробки аудіо даних та виокремлення ознак з них. Із цих досліджень було виявлено відмінності в аудіо сигналі різних емоцій, схожість вираження емоцій різними мовами та методики, що дозволяють безпечно розширити наявні дані новими синтетичними спостереженнями.

У другому розділі було розглянуто архітектури нейронних мереж, що застосовуються для подібних задач, порівняно сучасні технології для створення нейронних мереж та розглянуто процес побудови мережі за заданою архітектурою. Також було оцінено точність створеної нейронної мережу.

У третьому розділі було описано архітектуру мобільного додатку та наведено деталі його реалізації. Важливим етапом у цьому було створення додаткового рівню абстракції для узагальнення моделей. Велику увагу було приділено можливій різниці між тренуванням та роботою моделі. Щоб запобігти цьому було перевірено ознаки, що генеруються у середі навчання та середі роботи. Також було перевірено додаткову утиліту для обробки ознак на мобільному пристрої.

В результаті роботи отримано мобільний додаток, що відповідає поставленій меті.

ПЕРЕЛІК ПОСИЛАНЬ

1. Real-Time speech emotion analysis for smart home assistants / R. Chatterjee et al. *IEEE transactions on consumer electronics*. 2021. Vol. 67, no. 1. P. 68–76. DOI: 10.1109/tce.2021.3056421.
2. Kathleen P.-F. M., Kate D. Toronto emotional speech set (TESS). Borealis. URL: https://borealisdata.ca/dataset.xhtml?persistentId=doi:10.5683/SP2/E8H2MF (дата звернення: 24.01.2024).
3. CREMA-D: crowd-sourced emotional multimodal actors dataset / H. Cao et al. *IEEE transactions on affective computing*. 2014. Vol. 5, no. 4. P. 377–390. DOI: 10.1109/taffc.2014.2336244.
4. Livingstone S. R., Russo F. A. 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*. 2018. Vol. 13, no. 5. P. e0196391. DOI: 10.1371/journal.pone.0196391.
5. Jackson P., Ul Haq S. Surrey audio-visual expressed emotion (SAVEE) database. University of Surrey – Centre for Vision, Speech and Signal Processing. URL: http://kahlan.eps.surrey.ac.uk/savee/ (дата звернення: 24.01.2024).
6. EmoFilm – A multilingual emotional speech corpus / E. Parada-Cabaleiro et al. Zenodo. URL: https://zenodo.org/records/1326428 (дата звернення: 24.01.2024).
7. Gournay P., Lahaie O., Lefebvre R. A canadian french emotional speech dataset. *MMSys '18*: 9th ACM multimedia systems conference, Amsterdam Netherlands. New York, NY, USA, 2018. DOI: 10.1145/3204949.3208121.
8. ASVP-ESD: a dataset and its benchmark for emotion recognition using both speech and non-speech utterances / T. T. L. Dejoli et al. *Gloabal scientific journals*. Vol. 8, no. 5. e1793.
9. MELD: a multimodal multi-party dataset for emotion recognition in

conversations / S. Poria et al. Proceedings of the 57th annual meeting of the association for computational linguistics, Florence, Italy. Stroudsburg, PA, USA, 2019. DOI: 10.18653/v1/p19-1050.

10. Emotional voice conversion: Theory, databases and ESD / K. Zhou et al. *Speech communication*. 2022. Vol. 137. P. 1–18. DOI: 10.1016/j.specom.2021.11.006.
11. The emotional voices database: towards controlling the emotion dimension in voice generation systems / A. Adigwe et al. arXiv.org. URL: https://arxiv.org/abs/1806.09514 (дата звернення: 24.01.2024).
12. Audio analysis with machine learning: building ai-fueled so. AltexSoft. URL: https://www.altexsoft.com/blog/audio-analysis/ (дата звернення: 14.05.2024).
13. Librosa/librosa: 0.10.1 / M. Brian et al. Zenodo. URL: https://doi.org/10.5281/zenodo.8252662 (дата звернення: 07.02.2024).
14. Haba D. Data Augmentation with Python: Enhance deep learning accuracy with data augmentation methods for image, text, audio, and tabular data. Birmingham : Packt Publishing, 2023. 394 с.
15. Пилипенко В. О., Слюсарь І. І., Слюсар В. І., Маруженко В. М. Варіант використання нейронної мережі в системі «Smart Home». *Інтеграція інформаційних систем і інтелектуальних технологій в умовах трансформації інформаційного суспільства*: тези доп. IV Міжнародної науково-практичної конференції, що присвячена 50-ій річниці кафедри інформаційних систем та технологій. Полтава : ПДАУ, 2021. С. 93–96. DOI: 10.32782/978-966-289-562-9.
16. Typical cnn. Wikimedia Commons. URL: https://commons.wikimedia.org/w/index.php?curid=46476856 (дата звернення: 14.05.2024).
17. Damacharla P., Rajabalipanah H., Fakheri M. H. LSTM-CNN network for audio signature analysis in noisy environments. 10th annual conf. on computational science & computational intelligence, Las Vegas, 13–15 December 2023. DOI: 10.48550/arXiv.2312.07059.
18. Keras: The high-level API for TensorFlow. TensorFlow. URL:

https://www.tensorflow.org/guide/keras (дата звернення: 25.02.2024).

19. Keras vs TensorFlow vs PyTorch : All You Need To Know. LinkedIn. URL: https://www.linkedin.com/pulse/keras-vs-tensorflow-pytorch-all-you-need-know-topd-alliance-pvt-ltd/ (дата звернення: 25.02.2024).
20. Home. PyTorch. URL: https://pytorch.org/mobile/home/ (дата звернення: 25.02.2024).
21. TensorFlow Lite. TensorFlow. URL: https://www.tensorflow.org/lite/guide (дата звернення: 25.02.2024).
22. Android audio recording guide part 2 (audiorecord). Medium. URL: https://medium.com/@anuandriesz/android-audio-recording-guide-part-2-audiorecord-f98625ec4588 (дата звернення: 28.02.2024).
23. GitHub – subtitle-synchronizer/jlibrosa: librosa equivalent java library to process audio file adn extract features from it. GitHub. URL: https://github.com/Subtitle-Synchronizer/jlibrosa (дата звернення: 28.02.2024).
24. Rules of machine learning. Google for Developers. URL: https://developers.google.com/machine-learning/guides/rules-of-ml#training-serving_skew (дата звернення: 01.02.2024).

ДОДАТОК А

Посилання на Git

1. nosenko-mi/ser_diploma. GitHub. URL: https://github.com/nosenko-mi/ser_diploma.
2. nosenko-mi/EmotionClassification. GitHub. URL: https://github.com/nosenko-mi/EmotionClassification.

ДОДАТОК Б

Приклади роботи мобільного додатку

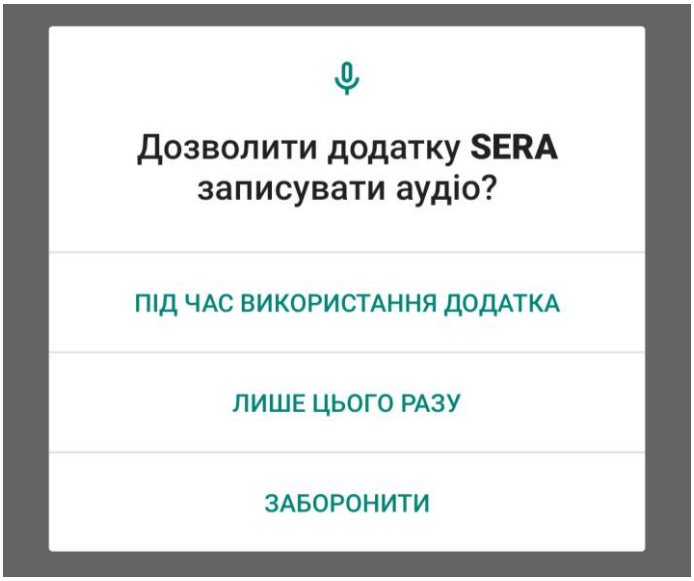

Рисунок Б.1 – Повідомлення надання дозволу

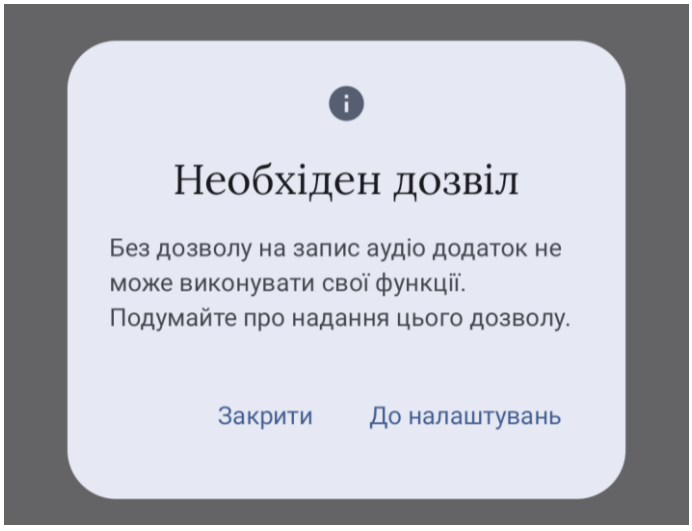

Рисунок Б.2 – Повідомлення про важливість дозволу

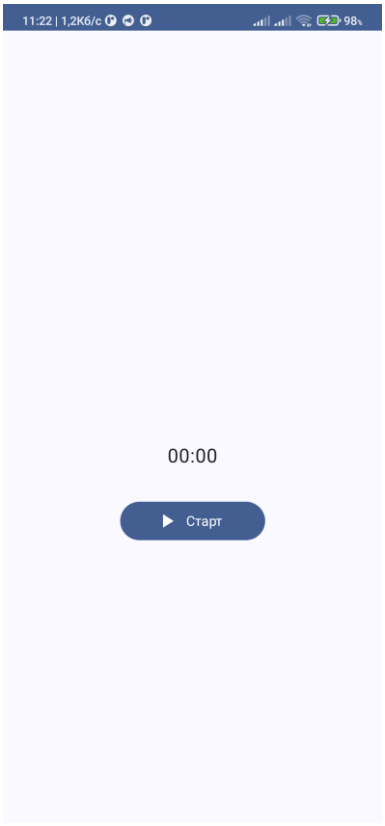

Рисунок Б.3 – Головний екран

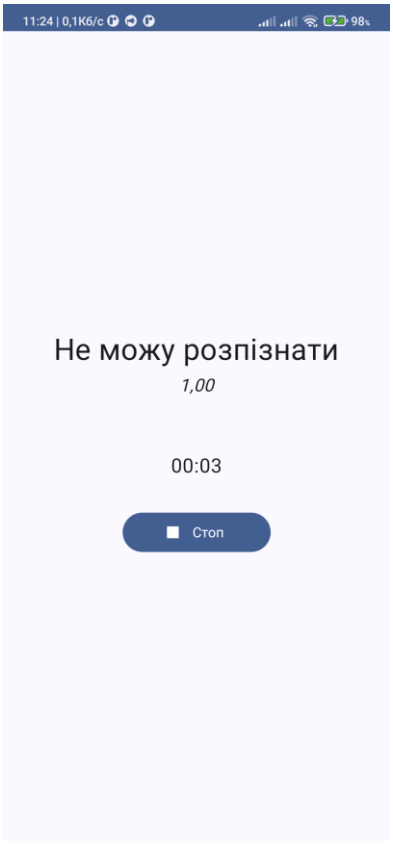

Рисунок Б.4 – Нерозпізнана емоція

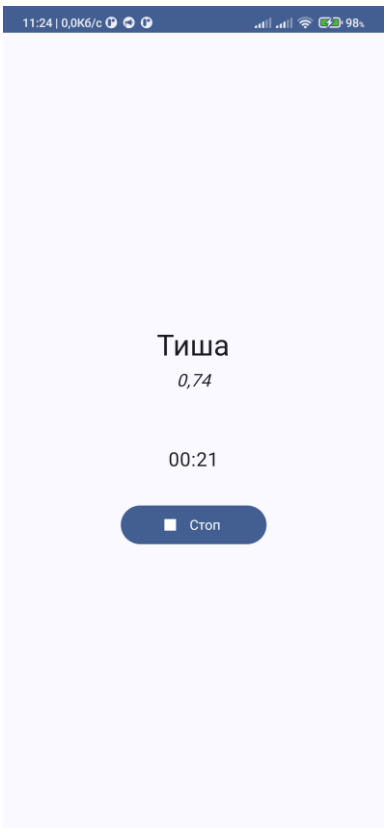

Рисунок Б.5 – Розпізнана тиша

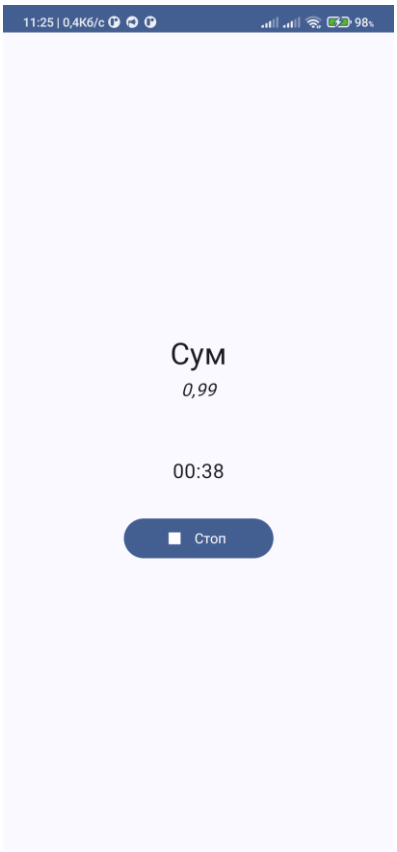

Рисунок Б.6 – Розпізнана емоція

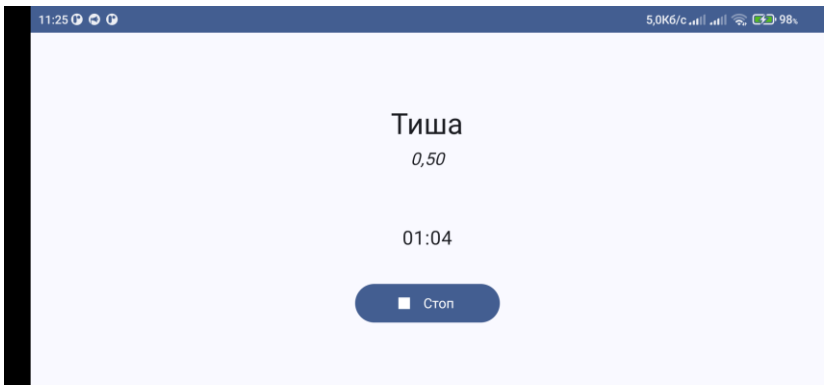

Рисунок Б.7 – Альбомна орієнтація додатку