ШИФР: Ukrainian Detection

РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ У СУЧАСНИХ ПІСНЯХ ЗА ДОПОМОГОЮ НЕЙРОННОЇ МЕРЕЖІ

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 4

1.1 Аналіз існуючих методів для визначення мови в аудіо 4

1.2 Порівняння з існуючим аналогом 5

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМИ РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ ТА СТВОРЕННЯ ДАТАСЕТУ 7

2.1 Математична модель 7

2.2 Створення та обробка датасету 8

2.3 Виділення пісенних ознак 9

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМИ ІЗ РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 11

3.1 Вибір мови програмування та необхідних бібліотек 11

3.2 Реалізація обраної архітектури нейронної мережі 11

3.3 Тестування розробленої мережі 13

ВИСНОВКИ 16

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 17

ВСТУП

Важливість розпізнавання мови в сучасному світі є важливою задачею, оскільки завдяки технологіям, які дозволяють визначити мову, можна налагодити взаємодію між людьми з різних країн, вивчати інші мови, перевіряти себе на правильність вимови тощо.

Штучні нейронні мережі застосовуються у багатьох сферах нашого життя, вирішуючи різноманітні задачі класифікації, розпізнавання образів, фільтрації і т. ін. Також вони знаходять широке застосування у виявленні та розпізнаванні мови.

Мета дослідження – створити і навчити нейронну мережу розпізнаванню мови (української чи іншої) у сучасних піснях, які збережено в аудіо-файлах.

Результати даного дослідження можуть бути корисними для тих, хто займається обробкою звуків, лінгвістам та іншим дослідникам, чия галузь так чи інакше пов’язана із мовою.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз існуючих методів для визначення мови в аудіо

На даний момент виділяють такі найпопулярніші методи для розпізнавання мови в аудіо:

1) моделі машинного навчання;
2) API для розпізнавання мови;
3) спеціальне програмне забезпечення для ідентифікації мови.

Також для розпізнавання мови використовують різні типи нейронних мереж, такі як згорткові, рекурентні, звичайні тощо.

Звичайна нейронна мережа (рис. 1.1) складається з декількох шарів, першим з яких є вхідний шар, у який подаються дані для класифікації, тобто ознаки. Далі йдуть приховані шари, у яких власне відбуваються всі обчислення, тобто тренування моделі, яке полягає в тому, щоб якнайкраще налаштувати ваги між нейронами для конкретної задачі. Цих шарів може бути декілька. Останній шар – вихідний, у якому мережа видає результат.

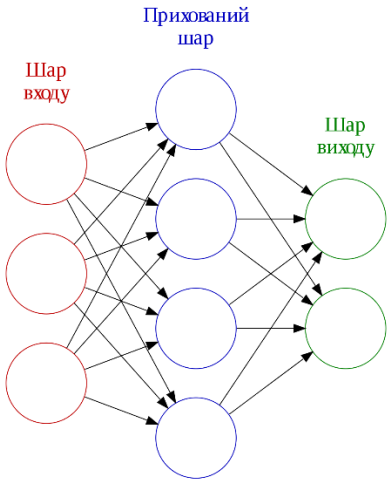

Рисунок 1.1 – Звичайна нейронна мережа (приклад одношарового перцептрона)

Згорткові нейронні мережі (ЗНМ) – це архітектура нейронних мереж (рис. 1.2), що містять спеціальні згорткові шари для роботи з зображеннями. ЗНМ використовують спільні ваги в згорткових шарах. Це означає, що для кожного рецептивного поля шару використовується той самий фільтр (банк ваг). У

результаті зменшується обсяг необхідної пам’яті та поліпшується продуктивність [7].

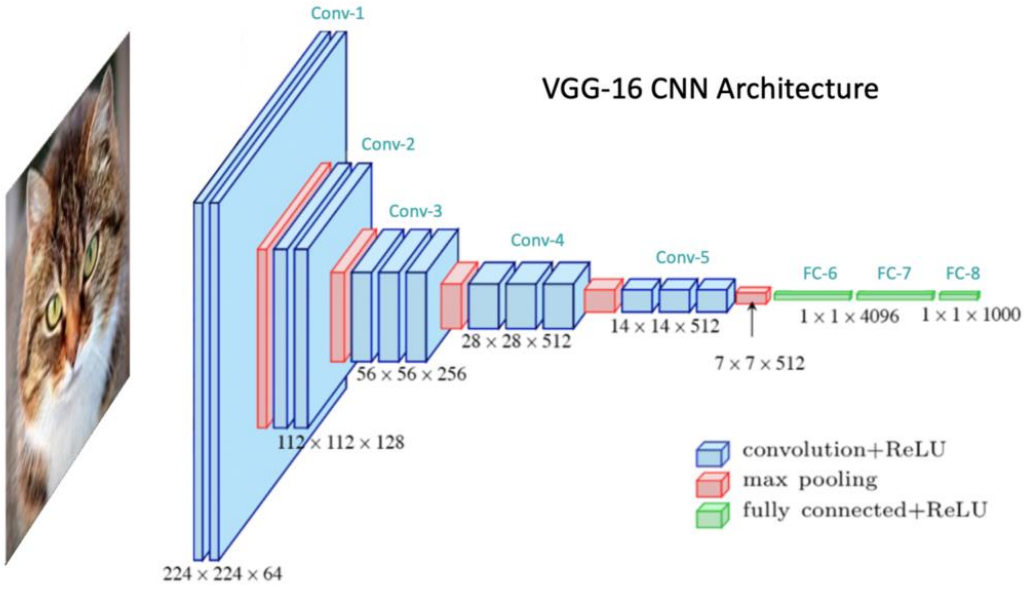

Рисунок 1.2 – Архітектура згорткової нейронної мережі

Нейронні мережі називають рекурентними (recurrent neural networks, RNN) (рис. 1.3), оскільки вони виконують одне й те саме завдання для кожного елемента послідовності, при цьому результат залежить від попередніх обчислень. Інший спосіб думати про RNN полягає в тому, що вони мають «пам’ять», яка фіксує інформацію про те, що було обчислено до цього часу [11; 12].

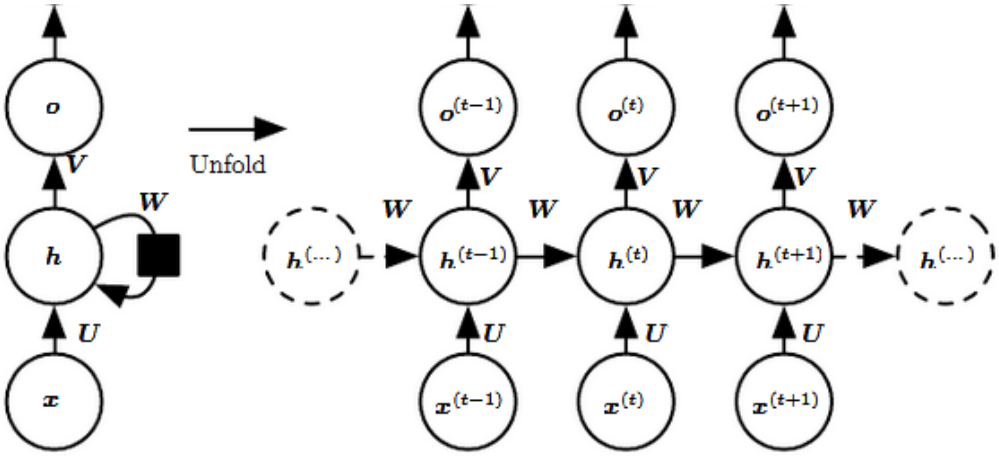

Рисунок 1.3 – Архітектура рекурентної нейронної мережі

1.2 Порівняння з існуючим аналогом

Розпізнавання мови застосовується у різних системах голосового керування, перекладачах на неявному рівні. Наприклад, при введенні тексту за

допомогою голосу перекладач автоматично визначає мову по певних закономірностях і потім переводить голос у текст.

Однак є й системи, які дозволяють дізнатись, власне який це голос без переведення у текст, а лише визначивши її за певними даними. Для прикладу можна навести розпізнавач мови від Translated Labs.

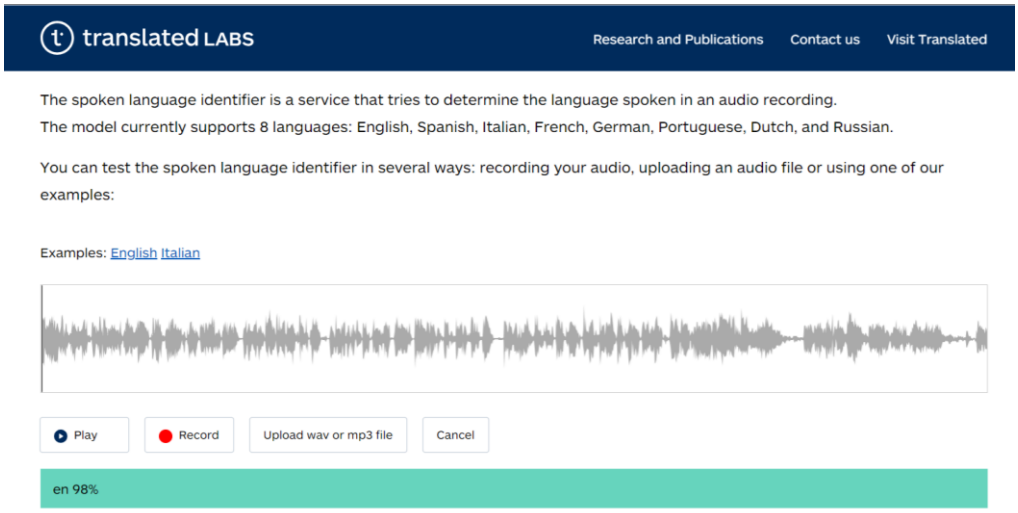

Рисунок 1.4 – Розпізнавач мови від Translated Labs

Розробники запевняють, що чим довшим є запис, то тим краще модель буде розпізнавати закономірності. Модель працює з тим, що виділяє певні характеристики, однак не звертає увагу на фонетичні й мовленнєві складники [13], що іноді призводить до плутанини між мовами.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМИ РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ ТА СТВОРЕННЯ ДАТАСЕТУ

2.1 Математична модель

Як відомо, звук – це невидима механічна хвиля, яка виникає внаслідок коливань частинок у певному середовищі, наприклад, повітрі.

При аналізі звуку виділяють декілька характеристик, як-от частота, довжина хвилі, амплітуда та швидкість. Також виділяють й інші характеристики, якщо говорити про сприйняття людиною, наприклад тембр, тон та темп.

Однак якщо підходити з іншої точки зору і сприймати звук як певний сигнал, то ситуація стає цікавішою. Обробкою сигналів займається відповідна галузь [5], яка досліджує теорію перетворення як цифрових, так і аналогових сигналів, що є змінними в часі або просторі фізичними величинами. Наприклад, за допомогою перетворення Фур'є можна перейти від часової (time) до частотної (frequency) області (domain) та виявити найбільш вживані частоти з певної звукової хвилі, наприклад, як на рисунку 2.1.

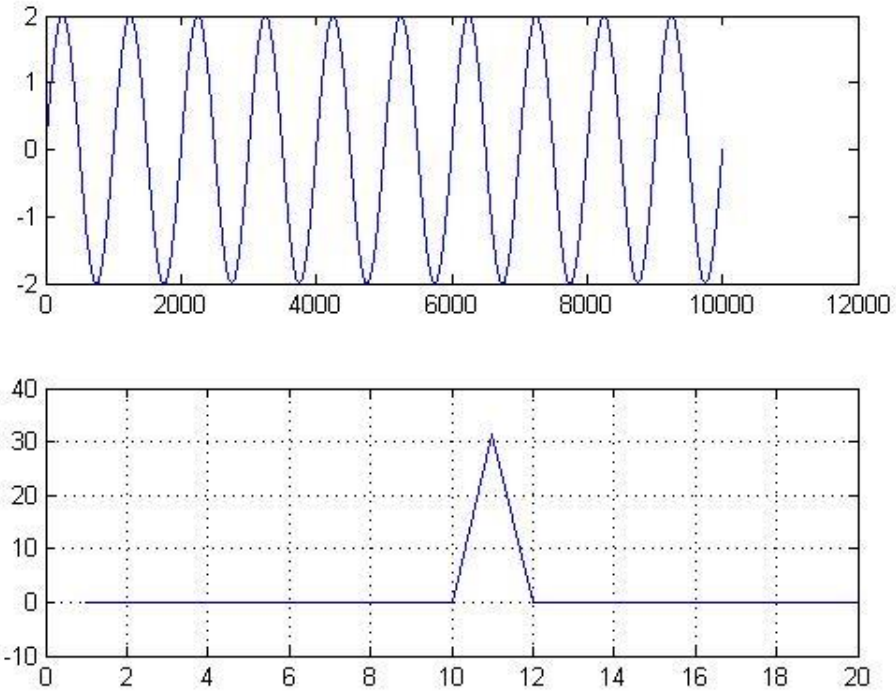

Рисунок 2.1 – Представлення звуку в часовій області (графік зверху) та у області частот (графік знизу)

2.2 Створення та обробка датасету

На початку планувалось створити датасет, який міститиме необроблені українські та англійські пісні, тобто звичайні пісні на платформі Spotify, але сучасна українська музика є змішаною із багатьма різними жанрами, то через це можуть з’являтись різні шуми, які будуть зменшувати точність майбутньої моделі. Тому було прийнято рішення виділити із пісень лише голос й працювати саме з ним. Для цього використовувався застосунок Ultimate Vocal Remover V5 (рис 2.2) [1], який дозволяє відокремити голос з пісні, так і навпаки вилучити його.

Для пришвидшення навчання було взято 30-секундні відрізки пісень, оскільки повна обробка аудіо-файлу може тривати досить довгий час, і до того ж не завжди є ймовірність, що обробка повної пісні повністю дасть точний результат.

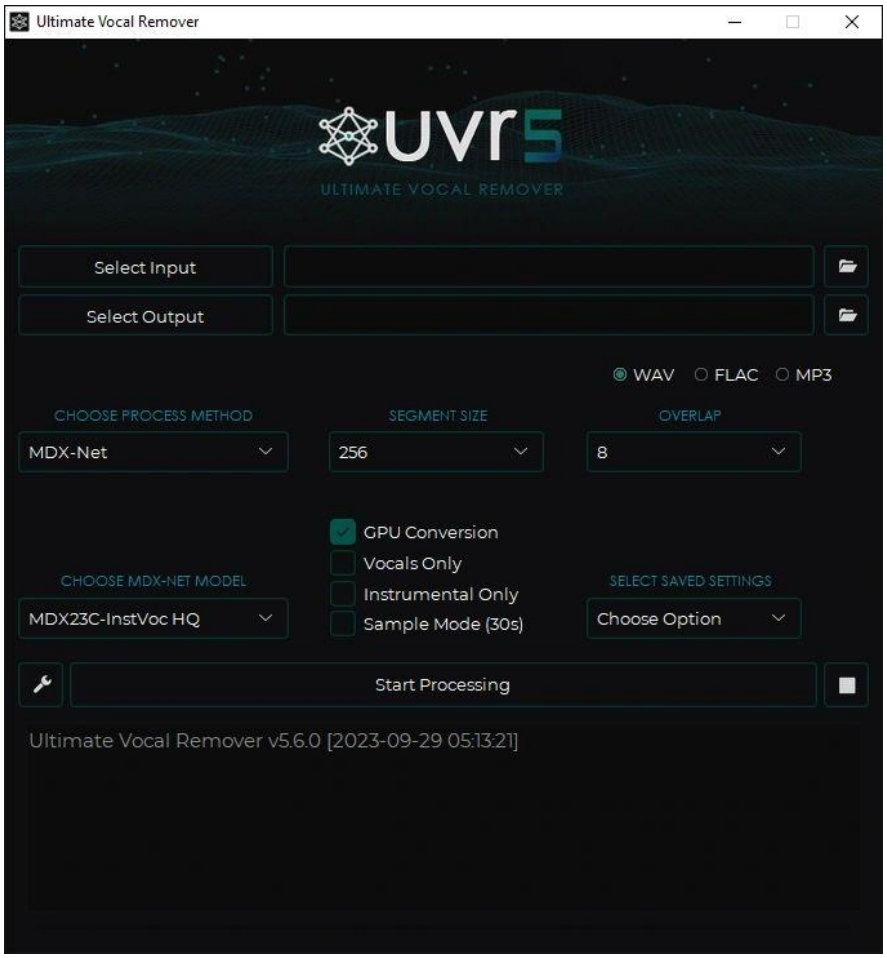

Рисунок 2.2 – Інтерфейс UVR5

Порівняємо хвильовий графік необробленої пісні “Пообіцяй мені” (рис. 2.3) та графік після відокремлення голосів за допомогою UVR5 (рис. 2.4).

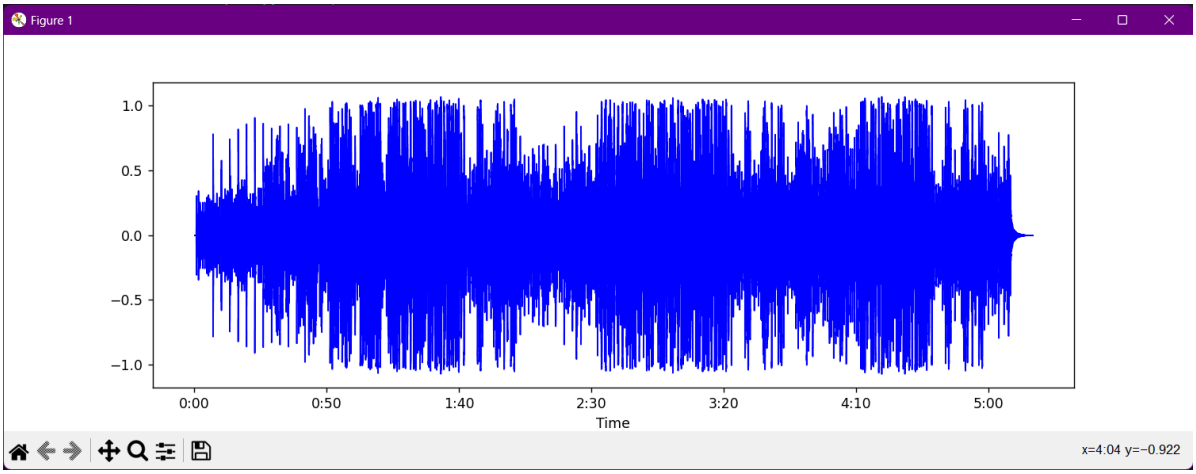

Рисунок 2.3 – Хвильовий графік необробленої пісні

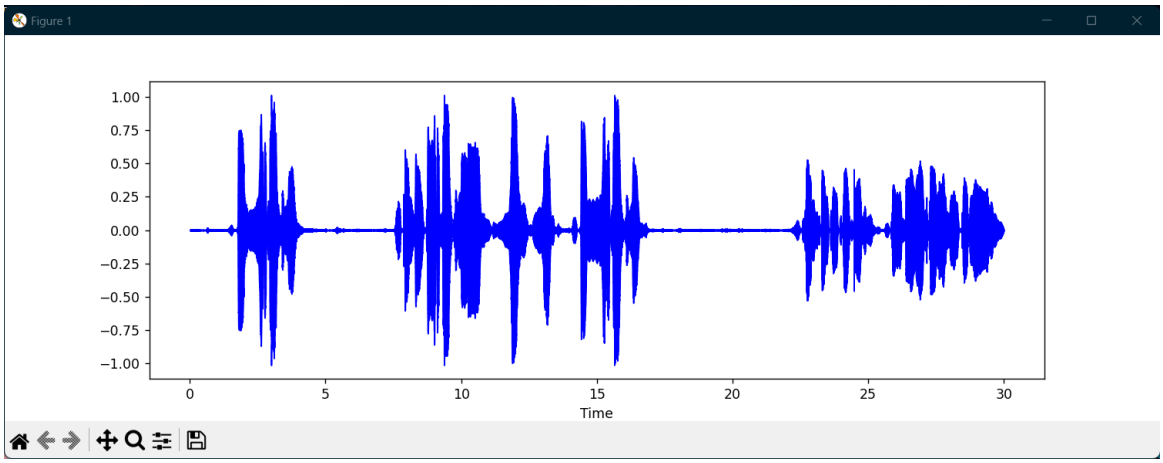

Рисунок 2.4 – Хвильовий графік обробленої пісні

Очевидно, що графік, який зображено на рисунку 2.4, краще підходить для аналізу та подальшого використання, оскільки завдяки ньому можливо точніше сказати де відбулось говоріння, а де ні.

Отже, датасет містить 520 голосів із пісень, 260 – українські, 260 – англійські.

2.3 Виділення пісенних ознак

Для того, щоб почати тренування нейронної мережі, необхідно зрозуміти, як саме обробити отриманий аудіо-сигнал (рис. 2.4). Для представлення сигналу в цифровому вигляді використовують такі техніки:

1. Спектрограми Мела – різновид спектрограми, де частота виражена не в герцах, а в мелах, які представляють собою одиницю звуку, що базується на сприйнятті звуку нашими органами чуття;

2. Кодування сигналу, використовуючи конвертори, наприклад ADC;

3. Виділення основних характеристик сигналу.

У даному дослідженні використовується третя техніка. За допомогою Python-бібліотеки librosa виділимо основні характеристики сигналу, як-от:

1. Спектральний центроїд – міра, яка характеризує спектр та показує, де розташований центр мас спектру [3];
2. Функція кольоровості – представляє тональний вміст сигналу [4];
3. Середньоквадратичне відхилення;
4. Ширина спектральної лінії – кількісна характеристика розмитості лінії в спектрі;
5. Спектральний спад – це частота, нижче якої заданий відсоток загальної спектральної енергії;
6. Швидкість переходу через нуль – швидкість з якою сигнал змінюється від позитивного до негативного та навпаки;
7. Мелові коефіцієнти [14] – невеликий набір ознак, які описують загальну форму спектральної оболонки.

Після виділення ознак маємо наступний csv-файл (рис. 2.5), який і будемо у подальшому використовувати для навчання нейронної мережі.

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11
filename	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4
ukrainian(1).wav	0.3650365173816681	0.11084567755460739	2411.5113603460327	2226.62446004139	4402.28171333809	0.13397273485874614	-198.67794799804688	87.05519866943336	-33.57948684692383	-2.16569209
ukrainian(10).wav	0.3823751807212829	0.04930853098630905	2838.124640257724	2939.525021998371	6440.894386126161	0.1085174843991873	-341.1250305175781	54.981742858886672	3.8035919666290283	2.2762432095
ukrainian(100).wav	0.3823751807212829	0.05485282093286514	2372.8150428576364	2507.1205981232883	4851.195665483504	0.10113090774961 3	-309.4673156738281	83.03164728551562	4.206775665283203	6.517251491
ukrainian(101).wav	0.3473291397094726 6	0.06432347744703293	2082.4269379096695	2326.3589649697255	4127.350042478957	0.08379919093459752	-317.8353576660156	85.97201538085938	2.921785593032837	24.176454546
ukrainian(102).wav	0.2645838558673858 6	0.1437988579273224	3166.5624606849688	2695.025576161987	6372.394800333785	0.16018005937983745	-120.49153137207031	40.285186767578125	-21.462467193603516	-13.308751 1
ukrainian(103).wav	0.2945546805858612	0.115116648375988	2440.6456680924994	2599.3979861770295	5299.467954842299	0.09033807807662539	-255.1018829345703	67.39384460449219	-3.6774954795837402	3.862513780
ukrainian(104).wav	0.2711164057254791	0.1442258656024933	2969.6938821869553	2623.51059339349	6121.9879811762285	0.13616357573045665	-141.09774780273438	50.60841369628906	-17.85483169555664	1.804503798
ukrainian(105).wav	0.3508458733558655	0.1267181783914566	3031.3331680895635	2575.413887903597	5859.181236999323	0.16827638230456657	-144.8926544189453	63.615318298339844	-8.237070083618164	-9.205594060
ukrainian(106).wav	0.2839991748332977 3	0.08751464653133743	2571.674178384629	2449.271915439526	4959.603343290441	0.1357395420133514	-193.90797424316406	81.11906433105469	-11.08075237227417	-9.870759010
ukrainian(107).wav	0.2293531149625778 2	0.13110259175300598	2257.2290213294295	2379.784893066744	4495.089485475522	0.09553910482778638	-209.49501037597656	73.25717163085938	-15.768799781799316	0.849335730
ukrainian(108).wav	0.2557601928710937 5	0.14070121943950653	2515.534882991652	2539.3836679435394	5078.002627346169	0.10921097982778638	-172.50938415527344	63.76025390625	-9.01506900787353 5	16.144018170
ukrainian(109).wav	0.2836002111434936 5	0.14989252388477325	2721.443167783276	2407.5407326892987	5072.369328076626	0.13816998839009287	-180.78842163085938	53.366462707519 53	-20.763992309570312	9.725029945
ukrainian(11).wav	0.2307938635349273 7	0.1098780557513237	2648.9683386033935	2491.648065309397	5220.485099107488	0.13631474639609134	-220.45648193359375	59.33058547973633	-30.745594024658203	-22.149713 5
ukrainian(110).wav	0.2383117079734802 2	0.08842588216606636	3104.463812145513	2665.192529432309	6349.636604616147	0.15010302280863003	-172.31361389160156	35.464981079101 56	-19.884485244750977	-4.735864630
ukrainian(111).wav	0.3726308047771454	0.11456147581338882	2818.1957707529696	2542.8353064610255	5628.107634269785	0.13754829902767027	-239.3123779296875	44.043292999267 58	1.2811638116836548	16.954107280
ukrainian(112).wav	0.2262484133243560 8	0.10801931470632553	2565.8786065015056	2328.018457822199	4750.896271980215	0.1316885461010062	-182.5834197998047	64.64790344238281	-48.509246826171875	-21.698169 7
ukrainian(113).wav	0.3310583829879761	0.08536235243082047	2808.140355695766	2585.666884289166	5568.974658505466	0.12955099288893188	-278.7171630859375	39.997688293457031	-18.898874282836914	0.928501546
ukrainian(114).wav	0.2948177456855774	0.15171054005622864	2977.645796445219	2861.9438347666173	6545.527086759868	0.12003366570723684	-122.3187255859375	60.862773895263 67	13.785533905029297	11.979657 17
ukrainian(115).wav	0.2648043334484100 3	0.13054148852825165	2790.071160503824	2138.6430032471067	4876.128848049294	0.16710594342589008	-126.7261734008789	30.126602172851562	-75.21357727050781	0.171103894
ukrainian(116).wav	0.2562455832958221 4	0.06981729716062546	2451.748427776334	2599.2685911560357	5077.985960780283	0.10993848865615326	-246.49066162109375	83.783630371093 75	-4.373466491699219	-7.445315360
ukrainian(117).wav	0.3419036567211151	0.0987020879983902	2427.109836750772	2549.1167913262393	5225.051738160313	0.09298583228521672	-300.4235534667969	57.89344406127930	-0.42635682225227356	3.745113849
ukrainian(118).wav	0.2521195411682129	0.14921888709068298	2386.698966810446	2311.6463544603102	4961.5283316503	0.10373784587848298	-156.87744140625	63.909679412841 8	-22.423646926879883	2.834033012
ukrainian(119).wav	0.2964622974395752	0.04062988609075546	2164.9878906379686	2357.328017730493	4428.439888496517	0.09071600474071208	-377.47686767578125	78.938682556152 34	-6.948600292205810 5	4.401718139
ukrainian(12).wav	0.2394750416278839	0.08609294891357422	2684.431046808526	2498.6345734134734	5375.009164721604	0.13522367211687306	-224.6827850341797	54.786449432373 05	-28.15173339843 75	-13.526649 4
ukrainian(120).wav	0.2640826404094696	0.1359698623418808	2456.498420000152	2413.1629832444264	5205.685188600522	0.1169237071884675	-140.02691650390625	80.144226074218 75	-6.275240898132324	0.926023006

Рисунок 2.5 – Вміст csv-файлу з ознаками

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОГРАМИ ІЗ РОЗПІЗНАВАННЯ УКРАЇНСЬКОЇ МОВИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Вибір мови програмування та необхідних бібліотек

Для реалізації нейронної мережі було обрано мову програмування Python, оскільки вона має велику підтримку бібліотек із машинного навчання, а також простий та лаконічний синтаксис.

Було використано наступні бібліотеки:

— librosa – для обробки звукових файлів та виділення з них ознак;

— pandas – для зчитування даних з csv-файлу під час тренування;

— numpy – для обробки масивів великої розмірності;

— keras – для навчання нейронної мережі;

— sklearn – для розподілу тренувальних та тестувальних даних, а також для кодування міток;

— joblib – для збереження файлів моделі.

3.2 Реалізація обраної архітектури нейронної мережі

Під час реалізації нейронної мережі необхідно враховувати низку факторів та критеріїв, оскільки вони всі тісно між собою пов’язані, й зміна одного критерію може призвести до зміни інших аби підтримувати систему в балансі.

Також необхідно орієнтуватись на кількість та якість даних. Достатня кількість даних і висока їх якість дозволяє моделі краще виявляти закономірності та менше помилятись на даних, які вона раніше не зустрічала.

У нашому випадку кількість даних відносно невелика. Для їх обробки створимо нейронну мережу – багатошаровий перцептрон (рис. 3.1), який матиме таку структуру:

- вхідний шар, який приймає характеристики звуку;
- прихований шар, який має 1024 нейрони;
- шар, який нормалізує дані;
- прихований шар, який має 256 нейронів;
- шар, який нормалізує дані;
- прихований шар, який має 32 нейрони;

– flatten шар;
– dropout шар, який зменшує ймовірність перенавчання;
– вихідний шар, який має 2 нейрони, відповідно до двох класів.

Усі шари, окрім останнього, матимуть ReLu в якості активаційної функції. Вихідний шар у свою чергу матиме активаційну функцію Softmax.

```
def create_model(X_train):
    model = Sequential()

    model.add(keras.layers.Dense(27,activation='relu',input_shape=X_train[1].shape))
    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Dense(1024, activation='relu'))
    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Dense(256,activation='relu'))
    model.add(keras.layers.BatchNormalization())

    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(32,activation='relu'))
    model.add(keras.layers.Dropout(0.3))

    model.add(keras.layers.Dense(2,activation='softmax'))
    return model
```

Рисунок 3.1 – Реалізація нейронної мережі

Зробимо попередню обробку даних, аби нейронна мережа змогла з ними працювати. Спочатку видалимо непотрібні стовпці з csv-файлу (рис. 3.2), а саме стовпець “filename”, який містить назву файлу.

Після цього закодуємо мітки та підлаштуємо дані до конкретного шаблону. Після цього розподілимо дані на тренувальні та тестові із розподілом 8:2 (рис. 3.2).

```
data = pd.read_csv('data.csv', encoding='latin-1', on_bad_lines='skip')
data = data.drop(['filename'], axis=1)

encoder = LabelEncoder()
y = encoder.fit_transform(data['label'])
scaler = StandardScaler()

X = scaler.fit_transform(np.array(data.iloc[:, :-1], dtype=float))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = create_model(X_train)
```

Рисунок 3.2 – Попередня обробка даних

Виконаємо останні налаштування мережі, в якості оптимізаційної функції оберемо RMSprop, а в якості функції втрат SparseCategoricalCrossentropy. Для того, аби фіксувати момент, коли модель почне перенавчатись, додамо параметр

ранньої зупинки (рис 3.3), який буде виконуватись тоді, коли втрати нейронної мережі не покращуватимуться протягом 10 епох.

Також добавимо параметр зменшення швидкості навчання (рис. 3.3), щоб зменшити ймовірність того, що модель попаде у plateau [9].

```
model.compile(
              optimizer=keras.optimizers.RMSprop(),
               loss=keras.losses.SparseCategoricalCrossentropy(),
               metrics=['accuracy'],
             )

early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.2,
                                              patience=4, min_lr=0.0000000001)
```

Рисунок 3.3 – Налаштування нейронної мережі

Після всіх цих налаштувань та визначення оптимальних значень гіперпараметрів моделі [10] виконаємо тренування.

```
history = model.fit(X_train,
                    y_train,
                    epochs=500,
                    validation_data=(X_test, y_test),
                    batch_size=32,
                    callbacks=[early_stopping, reduce_lr]
                    )
```

Рисунок 3.4 – Тренування нейронної мережі

3.3 Тестування розробленої мережі

У якості оцінки точності моделі використовувалась функція втрат SparseCategoricalCrossentropy та метрика точності. Наведемо відповідні графіки точності (рис. 3.5) та функції втрат (рис. 3.6).

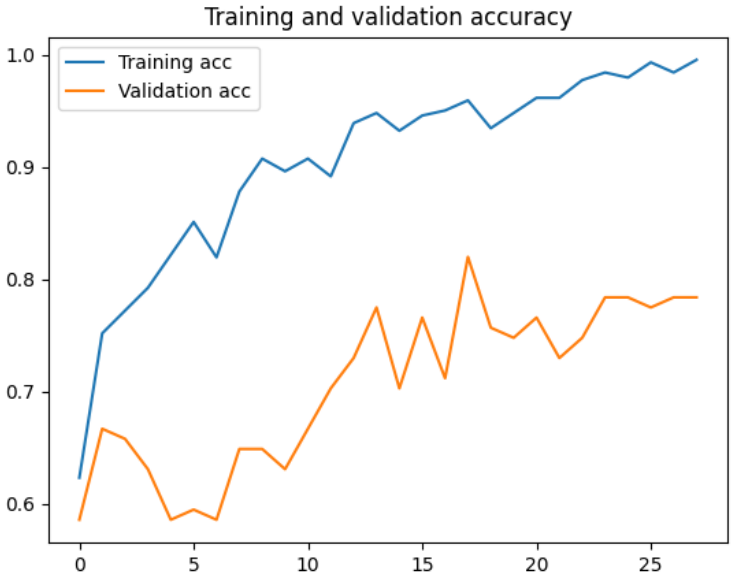

Рисунок 3.5 – Тренувальна та валідаційна точність на різних епохах

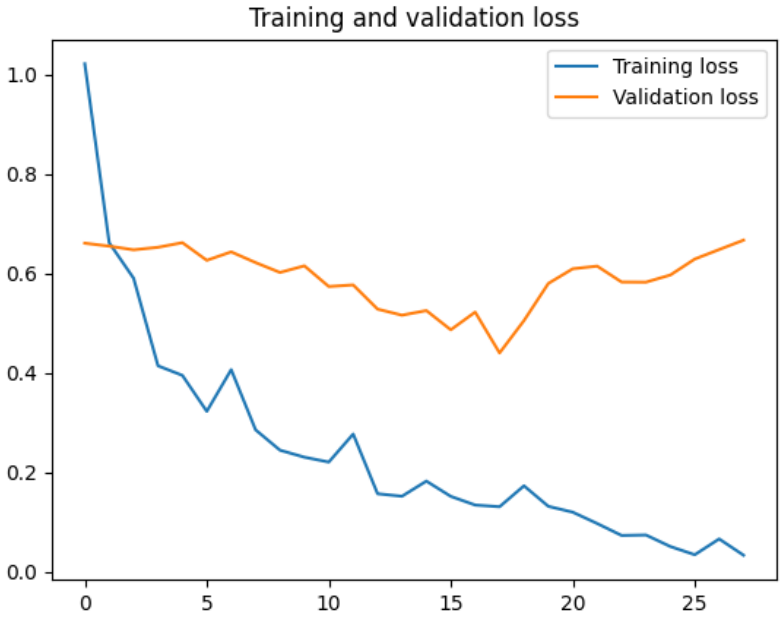

Рисунок 3.6 – Значення функції втрат на різних епохах

Обрахуємо тестову точність на тестовій вибірці (рис. 3.7).

```
test_loss, test_acc = model.evaluate(X_test, y_test)
print('test_acc: ', test_acc)
print('test_loss: ', test_loss)
```

Рисунок 3.7 – Обрахування тестової точності

На тестовій вибірці точність склала приблизно 81%. Тепер перевіримо мережу на тих піснях, які вона ще не бачила.

Для перевірки візьмемо два плейлісти з платформи Spotify: перший містить 36 пісень іноземної групи Twenty One Pilots, другий – 26 пісень групи української групи Один в каное.

Під час перевірки вияснили, що пісні групи Twenty One Pilots були правильно визначенні у 28 випадках, а пісні групи Один в каное – у 19 випадках.

ВИСНОВКИ

Під час виконання роботи було розглянуто основні типи нейронних мереж, програму-аналог та принципи їхньої роботи.

Далі було побудовано математичну модель та досліджено основні характеристики звуку з точки зору сигналів. Було виділено звукові характеристики та створено власний датасет, який налічує 520 пісень.

Розроблена нейронна мережа була навчена і протестована. На тестовій вибірці точність склала приблизно 81%. В результаті тестування моделі на піснях, які вона ще не бачила, з’ясувалося, що у більшості випадків вона розпізнає мови коректно, однак є випадки, коли модель не може правильно визначити мову. Це може бути пов’язано як й з малим розміром датасету, так і самою піснею, адже у деяких піснях неможливо чітко вловити особливості мови лише за допомогою спектральних характеристик, що і призводить до плутанини. Також було наведено графіки навчання та функції втрат.

Перспективами подальшого розвитку розробки може бути застосування і дослідження кількох типів нейронних мереж для розпізнавання мови, зокрема, з урахуванням аналізу та виділення характеристик звуку з морфологічної та фонетичної точки зору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ultimate Vocal Remover 5. URL: https://ultimatevocalremover.com/ (дата звернення 07.04.2024).
2. Müller, Meinard & Ellis, Daniel & Klapuri, Anssi & Richard, Gaël. (2011). Signal Processing for Music Analysis. Selected Topics in Signal Processing, IEEE Journal of. 5. 1088 - 1110. 10.1109/JSTSP.2011.2112333 (дата звернення 07.04.2024).
3. A case study of scaling multiple parameters by the violin - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Spectral-centroid-shows-the-center-of-mass-of-a-spectrum-and-is-connected-with-the_fig6_280770543 (дата звернення 07.04.2024).
4. Shah, Ayush & Kattel, Manasi & Nepal, Araju & Shrestha, D.. (2019). Chroma Feature Extraction (дата звернення 07.04.2024).
5. Теорія сигналів [Електронний ресурс] : навч. посіб. для студ. спеціальності 153 «Мікро- та наносистемна техніка» / КПІ ім. Ігоря Сікорського ; уклад.: А.О. Попов. – Електронні текстові данні (1 файл: 7399 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 268 с. (дата звернення 07.04.2024).
6. K, Shashank & M, Rahul & Munavalli, Jyoti & Anand, Prajwal. (2022). Dual-Language Detection using Machine Learning. Advances in Intelligent Systems and Technologies. 177-180. 10.53759/aist/978-9914-9946-1-2_32 (дата звернення 07.04.2024).
7. Згорткові нейронні мережі. URL: https://conf.ztu.edu.ua/wp-content/uploads/2019/02/45-1.pdf (дата звернення 07.04.2024).
8. Mcdonald, Susan. (2012). Perception: A Concept Analysis. International journal of nursing knowledge. 23. 2-9. 10.1111/j.2047-3095.2011.01198.x (дата звернення 07.04.2024).
9. Yoshida, Yuki & Okada, Masato. (2020). Data-dependence of plateau phenomenon in learning with neural network—statistical mechanical analysis. Journal of Statistical Mechanics: Theory and Experiment. 2020. 124013. 10.1088/1742-5468/abc62f (дата звернення 07.04.2024).

10. Yang, Li & Shami, Abdallah. (2020). On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice (дата звернення 07.04.2024).
11. Du, Ke-Lin & Swamy, M.N.s. (2014). Recurrent Neural Networks. 10.1007/978-1-4471-5571-3_11 (дата звернення 07.04.2024).
12. Reccurent Neural Networks (RNNs). URL: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85 (дата звернення 07.04.2024).
13. Spoken language identifier. URL: https://translatedlabs.com/spoken-language-identifier (дата звернення 07.04.2024).
14. Koolagudi, S.G., Rastogi, D., Rao, K.S. (2012). Spoken Language Identification Using Spectral Features. In: Parashar, M., Kaushik, D., Rana, O.F., Samtaney, R., Yang, Y., Zomaya, A. (eds) Contemporary Computing. IC3 2012. Communications in Computer and Information Science, vol 306. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32129-0_52 (дата звернення 07.04.2024).