МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

КУРСОВАЯ РАБОТА

РАСПОЗНАВАНИЕ МУЗЫКАЛЬНЫХ ЖАНРОВ С ПОМОЩЬЮ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

Работу выполнил _____ А.А. Хачатрян
(подпись)

Направление подготовки 01.04.02 Прикладная математика и информатика

Направленность (профиль) Математическое и информационное обеспечение экономической деятельности

Научный руководитель
канд. физ.-мат. наук, доц. _____ А.В. Письменский
(подпись)

Нормоконтролёр
канд. физ.-мат. наук, доц. _____ Г.В. Калайдина
(подпись)

Краснодар
2021

РЕФЕРАТ

Курсовая работа, 30 с., 21 рис., 1 табл., 10 источников.

РАСПОЗНАВАНИЕ, КЛАССИФИКАЦИЯ, МУЗЫКАЛЬНЫЕ ЖАНРЫ, НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ, НЕЙРОННЫЕ СЕТИ, АНАЛИЗ АУДИОДАНЫХ, МЕЛ-ЧАСТОТНЫЕ КЕПСТРАЛЬНЫЕ КОЭФФИЦИЕНТЫ, ПЕРСЕПТРОН, СВЁРТОЧНАЯ СЕТЬ, РЕКУРРЕНТНАЯ СЕТЬ, ДОЛГАЯ КРАТКОСРОЧНАЯ ПАМЯТЬ

Целью данной работы является разработка нейронной сети для распознавания жанров музыкальных композиций.

Для достижения поставленной цели необходимо знать язык программирования Python, уметь разрабатывать нейронные сети, а также решить следующие задачи:

– рассмотреть методы решения поставленной задачи;

– найти данные для обучающей выборки;

– разработать архитектуру нейронной сети для распознавания музыкальных жанров;

– написать программу, распознающую жанры музыкальных композиций.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мы живём во время молниеносно развивающихся технологий и растущей численностью разнообразных сервисов, предоставляющих пользователям некоторую информацию. В частности, появляется всё больше музыкальных сервисов. Для того, чтобы достичь успеха на рынке, подобные средства должны предоставлять пользователю возможность не только прослушивать музыкальные композиции, но и обеспечить свои приложения дополнительными функциями. Например, интересной и полезной была бы возможность анализа жанровых предпочтений пользователя, исходя из прослушанных им композиций, для последующего составления списка рекомендаций. Также, в нашей жизни может возникнуть ситуация, когда мы услышали какую-либо музыкальную композицию, но не можем точно определить её жанр.

Одним из вариантов решения вышеприведённых вопросов является разработка нейронной сети для распознавания жанров музыкальных композиций. Именно эта задача и является целью данной работы.

Таким образом, необходимо реализовать модель нейронной сети, которая способна принимать на вход музыкальный аудиофайл и распознавать его жанр. Данная задача является примером задачи классификации, где классы – это жанры музыки. Также необходимо найти данные для обучающей выборки.

1 Актуальность темы

В настоящее время стала весьма актуальной задача распознавания и поиска музыкальных композиций в связи с увеличением числа музыкальных веб-сервисов и ростом их популярности. Данные сервисы предоставляют немалое количество полезных функций: прослушивание песни, распознавание музыкальных композиций, предоставление информации об исполнителях и композициях, рекомендации и т. д.. Но эти сервисы не предоставляют пользователю возможности определить жанр музыкальной композиции, которую он прослушивает. Либо просто у пользователя есть какая-либо песня в смартфоне или компьютере, и он хочет определить её жанр, но нет сервиса, который позволил бы ему это сделать.

Ни одна система распознавания музыкальных композиций не достигла высоких показателей для массового применения. Например, самый высокий процент успешной классификации на MIREX 2013 (это ежегодное соревнование по разным областям музыкального информационного поиска, включая классификацию произведений по жанрам) составил 76% при классификации по 10 жанрам [2]. По данным MIREX за 2020 точность классификации держится примерно на том же уровне, что и на MIREX 2013 [7].

Таким образом, задача создания нейросети для распознавания музыкальной композиции, очевидно, является актуальной. Данную нейросеть в дальнейшем можно будет интегрировать в какой-нибудь музыкальный сервис либо в самостоятельное приложение для распознавания жанров, и у пользователей наконец появится и данная желаемая функция.

2 Технология аудиоанализа

Одним из наиболее рациональных представлений аудиозаписи для анализа является метод мел-частотных кепстральных коэффициентов (MFCC) (Рисунок 1). Он широко применяется для составления характеристик речевых сигналов и получил широкое распространение в задачах распознавания речи.

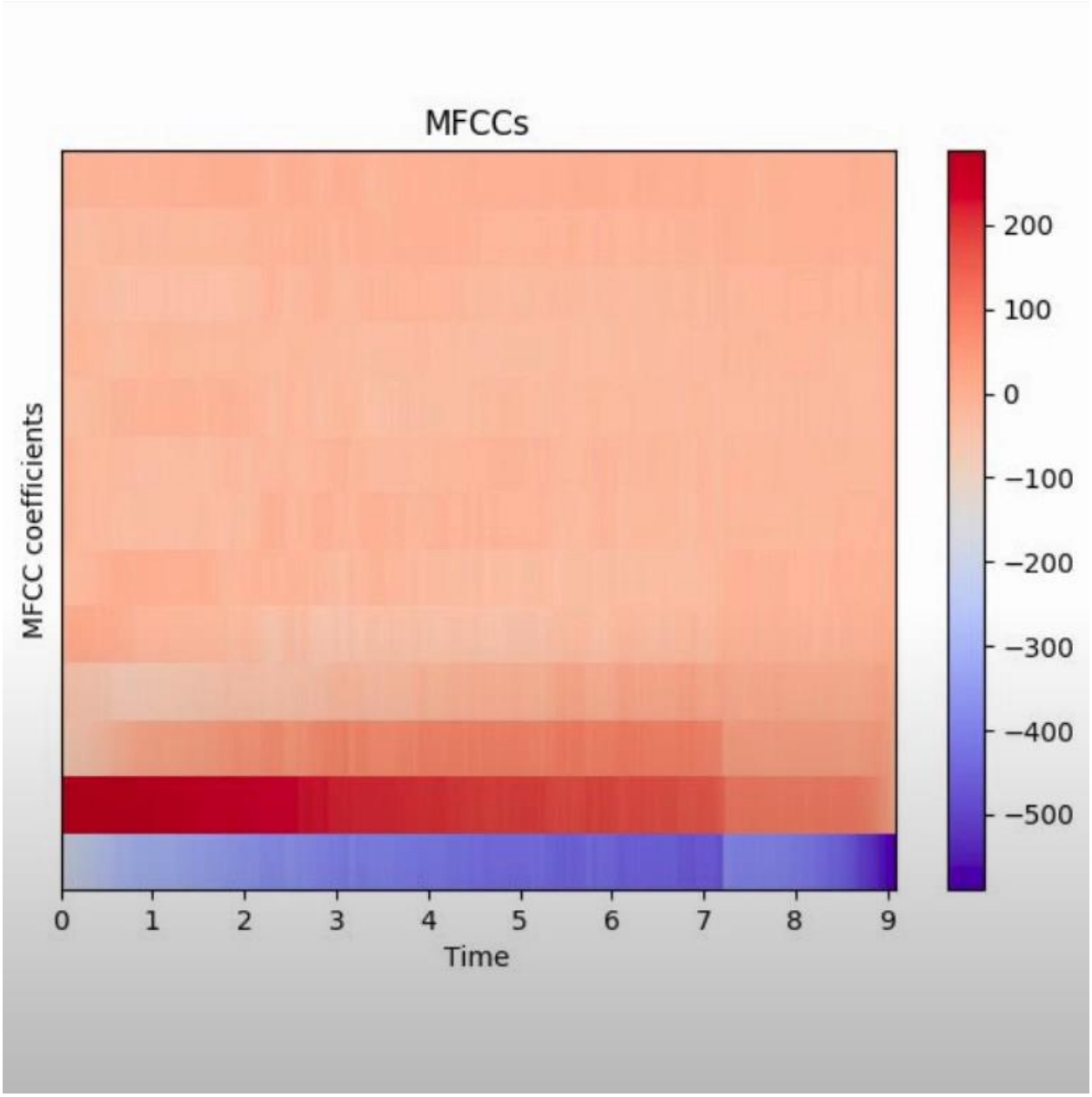

Рисунок 1 – Мел-частотные кепстральные коэффициенты

Данная метрика больше приближена к тому, как человек анализирует музыкальную композицию на слух, так как человек воспринимает высоту звука в определённый момент времени, а не данные о частотах, которые являются основой спектрограмм (Рисунок 2). Кроме того, данные, представленные в этой метрике, в среднем в 4 раза меньше по объёму, чем данные спектрограмм [1].

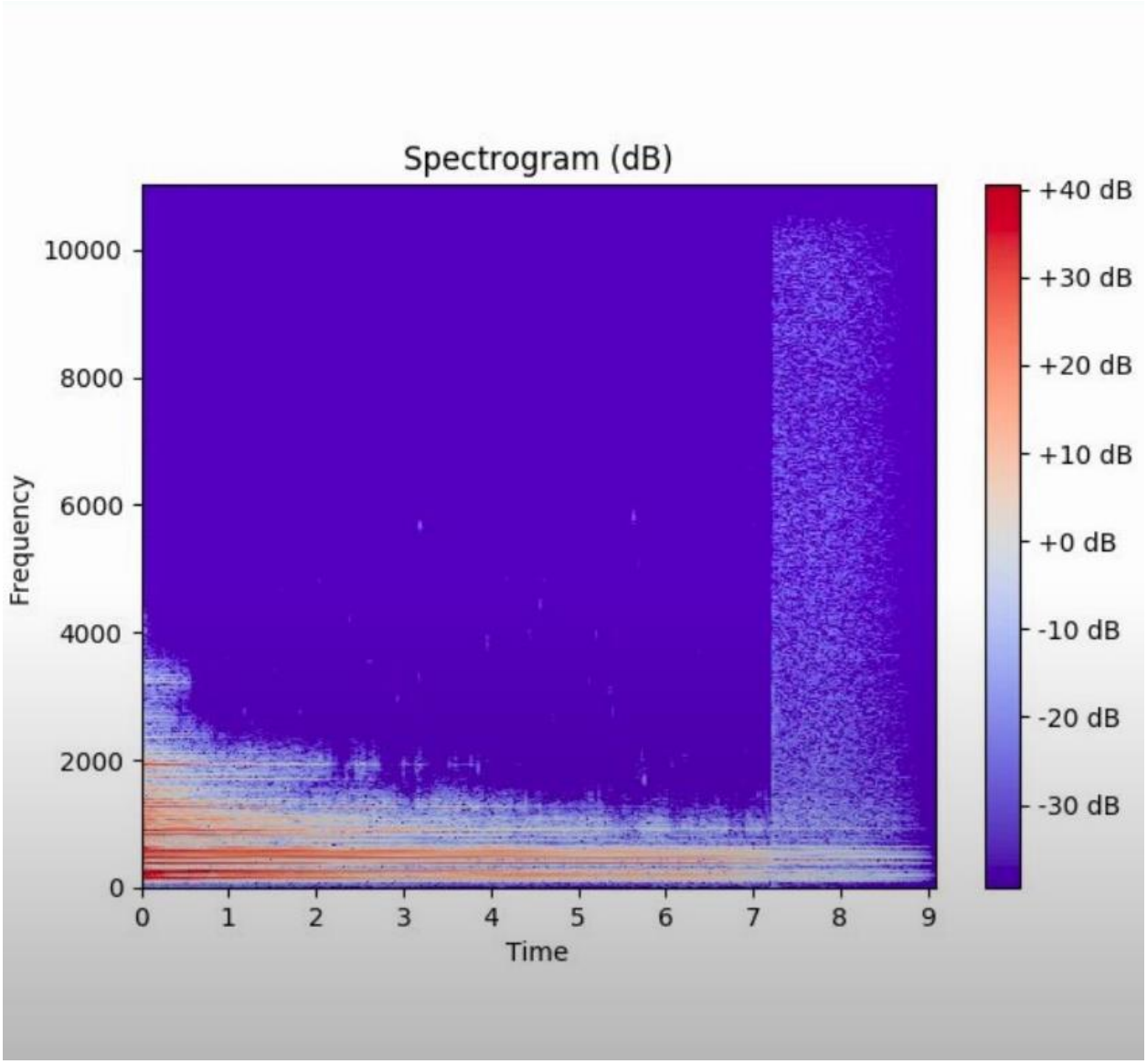

Рисунок 2 – Спектрограмма

Спектрограмма (сонограмма) – это способ визуализации уровня или «громкости» сигнала во времени на различных частотах, присутствующих в

форме волны. Обычно спектрограмма представляется в виде тепловой карты). Это изображение, изображающее зависимость спектральной плотности мощности сигнала от времени. Они применяются для анализа звуков животных, идентификации речи, сейсмологии, в различных областях музыки, радиолокации и гидролокации, обработке речи, и др. [10].

Распространённым представлением спектрограммы является 2D-диаграмма: на горизонтальной оси представлено время, по вертикальной оси представлена частота.

Спектрограмма обычно формируется одним из 1-х способов: аппроксимируется, как набор фильтров, полученных из серии полосовых фильтров, либо рассчитывается по сигналу времени, используя оконное преобразование Фурье (Рисунок 3).

Формирование спектрограммы при помощи преобразования Фурье выполняется методами цифровой обработки данных. Происходит цифровая выборка данных во временной области. Аудиосигнал разбивается на части, которые, как правило, перекрываются, и после производится преобразование Фурье, чтобы рассчитать величину частотного спектра для каждой из частей. Каждая часть соответствует вертикальной линии на изображении – значение амплитуды в зависимости от частоты в каждый момент времени. Спектры на изображении располагаются рядом.

Спектральные характеристики многомерные, из-за чего занимают существенную часть памяти при работе с огромными объёмами данных. Поэтому необходимо найти более эффективное представление спектра, которое можно использовать на практике. А поиски наиболее рационального представления приводят к мел-частотным кепстральным коэффициентам, которые, как говорилось ранее, используются в качестве характеристики речевых сигналов.

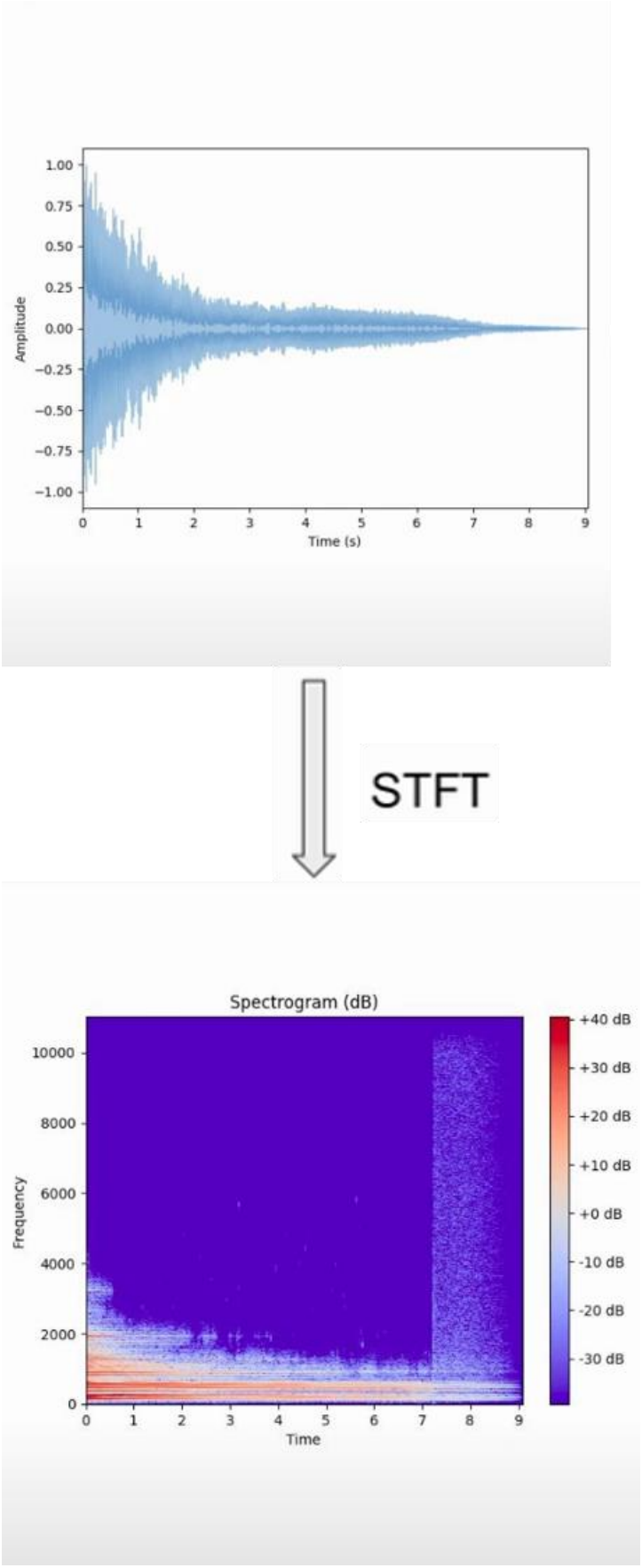

Рисунок 3 – Получение спектрограммы

Мел-частотные кепстральные коэффициенты (MFCC) описывают мощность огибающей спектра, которая характеризует модель речевого тракта. Они получаются путём преобразования Фурье исходного сигнала, отображения значений спектра на мел-шкалу и последующего дискретного косинусного преобразования значений на мел-шкале. Полученные значения амплитуд спектра и будут являться целевыми коэффициентами.

Данные имеют повышенную помехоустойчивость и позволяют принимать точные решения на относительно небольших интервалах анализа аудиоданных. Главной идеей данного метода является наибольшее приближение информации, поступающей на слуховой анализатор мозга человека. Признаки, построенные на основе мел-кепстральных коэффициентов, учитывают психоакустические принципы восприятия речи, так как используют мел-шкалу, которая связана с критическими полосами слуха [4].

Приведём значения понятий мела и кепстра. Мел – это единица высоты звука, которая основана на восприятии этого звука органами слуха человека или, иначе говоря, своеобразное представление энергии спектра сигнала, которое обычно является вектором из тринадцати вещественных значений. А кепстр – это результат дискретного косинусного преобразования от логарифма амплитудного спектра сигнала.

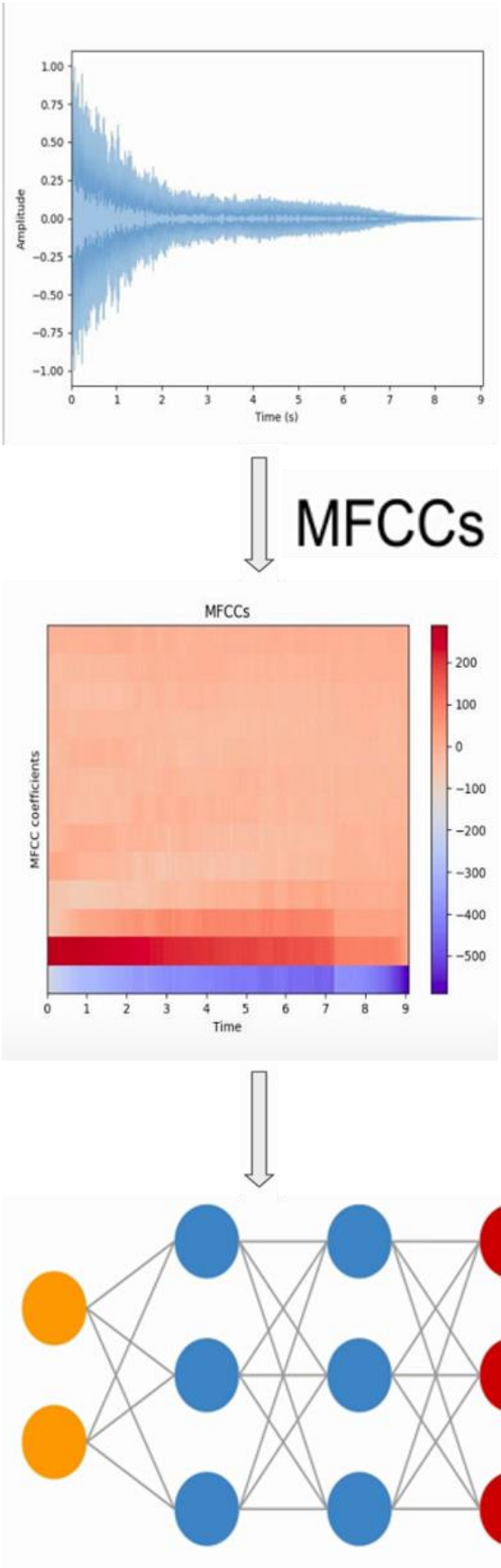

Рисунок 4 – Процесс предобработки данных

На рисунке выше приведён процесс предобработки аудиоданных, применяя метод мел-частотных кепстральных коэффициентов, для дальнейшего обучения нейронной сети (Рисунок 4). Сначала из аудиофайла извлекаются эти коэффициенты, а потом они подаются на вход нейросети.

3 Набор обучающих данных

В качестве обучающей выборки была использована база аудиозаписей GTZAN. В данном датасете 1000 аудиофайлов по 30 секунд. Всего 10 музыкальных жанров, по 100 звуковых дорожек на жанр. В наборе данных представлены следующие жанры: блюз, классическая, кантри, диско, хип-хоп, джаз, металл, поп, регги, рок. Все дорожки представляют собой моно 16-битные аудиофайлы 22050 Гц в формате .wav

Т. к. каждый жанр представлен всего 100 звуковыми дорожками (что не много), эти аудиофайлы были разбиты на 10 частей, и в итоге получены данные по 1000 звуковых дорожек в каждом жанре по 3 секунды.

4 Проектирование нейросети

Для анализа аудиоданных и извлечения мел-частотных кепстральных коэффициентов была использована библиотека librosa.

Librosa – это модуль Python для анализа звуковых сигналов, который предназначен для работы с аудио (в частности, с музыкой). Он включает все необходимое для создания системы MIR (поиск музыкальной информации) и имеет подробную документацию вместе со большим количеством примеров и руководств [5].

Для создания нейронной сети использовался пакет Keras.

Tensorflow – самая известная библиотека, используемая в создании моделей глубокого обучения. У него очень большое сообщество, и оно даёт большую гибкость в работе. Тем не менее, Tensorflow не так удобен для пользователя. Чтобы решить эту проблему, существует высокоуровневый API Keras от Tensorflow, который предоставляет строительные блоки для более лёгкого создания и обучения моделей глубокого обучения.

Keras – это высокоуровневое API для построения и тренировки моделей глубокого обучения. Оно используется для быстрого прототипирования, углублённого исследования.

Архитектура нейросети и параметры представлены ниже:

– 3 скрытых полносвязных слоя по 512, 256 и 64 нейрона;

– в выходном слое 10 нейронов – по количеству жанров;

– функция активации на скрытых слоях – ReLU, на выходном SoftMax.;

– оптимизатор сети – Adam со скоростью обучения 0.0001;

– количество эпох обучения – 50.

5 Программирование

Для разработки нейросети был использован язык программирования Python. Разработка велась в IDE PyCharm.

Опишем основные моменты в разработке данной программы.

5.1 Извлечение мел-частотных кепстральных коэффициентов

Ниже на рисунке (Рисунок 1) приведена структура JSON-файла, который формируется для хранения информации об обучающей выборке [3]. В файле хранится следующая информация:

– mapping – названия жанров ("blues", "classical" и т. д.);

– mfcc – мел-частотные кепстральные коэффициенты;

– labels – коды жанров (значения от 0 до 9).

```
17      # dictionary to store data
18      data = {
19          "mapping": [],
20          "mfcc": [],
21          "labels": []
22      }
```

Рисунок 5 – структура JSON-файла с информацией о датасете

Ниже на рисунке показан код, извлекающий мел-частотных кепстральных коэффициентов (Рисунок 6). Алгоритм извлечения необходимых характеристик таков:

– с помощью метода *load()* библиотеки *librosa* загружаем аудиофайл;

– при помощи метода *feature.mfcc()* этой же библиотеки извлекаем мел-частотные кепстральные коэффициенты, передав методу все необходимые параметры.

```
# load audio file
file_path = os.path.join(dirpath, f)
signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)

# process segments extracting mfcc and storing data
for s in range(num_segments):
    start_sample = num_sample_per_segment * s              # s = 0 -> 0
    finish_sample = start_sample + num_sample_per_segment  # s = 0 -> num_sample_per_segment

    mfcc = librosa.feature.mfcc(signal[start_sample:finish_sample],
                                sr=sr,
                                n_fft=n_fft,
                                n_mfcc=n_mfcc,
                                hop_length=hop_length)
```

Рисунок 6 – Извлечение мел-частотных кепстральных коэффициентов

5.2 Построение нейронной сети. Персептрон

Многослойный персептрон – класс ИНС прямого распространения, состоящих минимум из 3-х слоёв: входного, скрытого и выходного. За исключением входных нейронов все остальные нейроны используют нелинейную функцию активации [8].

В настоящее время многослойные персептроны всё ещё являются распространённым инструментом анализа данных и входят в большинство платформ бизнес-аналитики.

Делим входные данные на обучающую и тестовую выборки, задав размер тестовой выборки 30 % от входных данных (Рисунок 7).

```
# split the data into train and test sets
inputs_train, inputs_test, targets_train, targets_test = train_test_split(inputs,
                                                                          targets,
                                                                          test_size=0.3)
```

Рисунок 7 – Деление на обучающую и тестовую выборки

На рисунке ниже (Рисунок 8) приведён процесс создания архитектуры нейросети при помощи библиотеки *Keras*:

– в выходном слое 10 нейронов;

– количество скрытых слоёв равно 3 (512, 256, 64 – количество нейронов в каждом скрытом слое соответственно). Скрытые слои полносвязные;

– функция активации на скрытых слоях – ReLU, на выходном SoftMax.

```
53      # build the network architecture
54      model = keras.Sequential([
55          # input layer
56          keras.layers.Flatten(input_shape=(inputs.shape[1], inputs.shape[2])),
57
58          # 1st hidden layer
59          keras.layers.Dense(512, activation="relu"),
60
61          # 2st hidden layer
62          keras.layers.Dense(256, activation="relu"),
63
64          # 3st hidden layer
65          keras.layers.Dense(64, activation="relu"),
66
67          # output layer
68          keras.layers.Dense(10, activation="softmax")
69      ])
```

Рисунок 8 – Архитектура нейросети

Задаём конфигурацию нейронной сети для обучения (Рисунок 9):

– оптимизатор сети – *Adam* со скоростью обучения 0.0001;

– функцией потерь является *sparse_categorical_crossentropy*, т. к. в данном случае решается задача классификации с множеством классов;

– метрика *accuracy* – метрика, которую мы хотим получить. Она отображает точность обучения нейронной сети.

```
71      # compile network
72      optimizer = keras.optimizers.Adam(learning_rate = 0.0001)
73      model.compile(optimizer = optimizer,
74                    loss = "sparse_categorical_crossentropy",
75                    metrics = ["accuracy"])
```

Рисунок 9 – Конфигурация нейронной сети

Далее обучаем нейросеть при помощи метода *fit()* (Рисунок 10). Для обучения заданы следующие параметры:

– количество эпох равно 50;

– параметр *batch_size* = 32 – это количество элементов выборки, с которыми идёт работа в пределах одной итерации до изменения весов.

```
78      # train network
79      history = model.fit(inputs_train,
80                  targets_train,
81                  validation_data = (inputs_test, targets_test),
82                  epochs = 50,
83                  batch_size = 32)
84
```

Рисунок 10 – Обучение нейросети

Нейросеть после 50 эпох обучения выдаёт точность:

– 91.44 % на обучающей выборке (accuracy);

– 57.45 % на тестовой выборке (val_accuracy).

Эти данные представлены на рисунке ниже (Рисунок 11):

```
Epoch 50/50
219/219 [==============================] - 2s 9ms/step - loss: 0.2466 - accuracy: 0.9144 - val_loss: 2.3049 - val_accuracy: 0.5745
```

Рисунок 11 – Результат обучения нейросети

На рисунке ниже представлены графики точности и ошибок на тренировочных и валидационных данных (Рисунок 12):

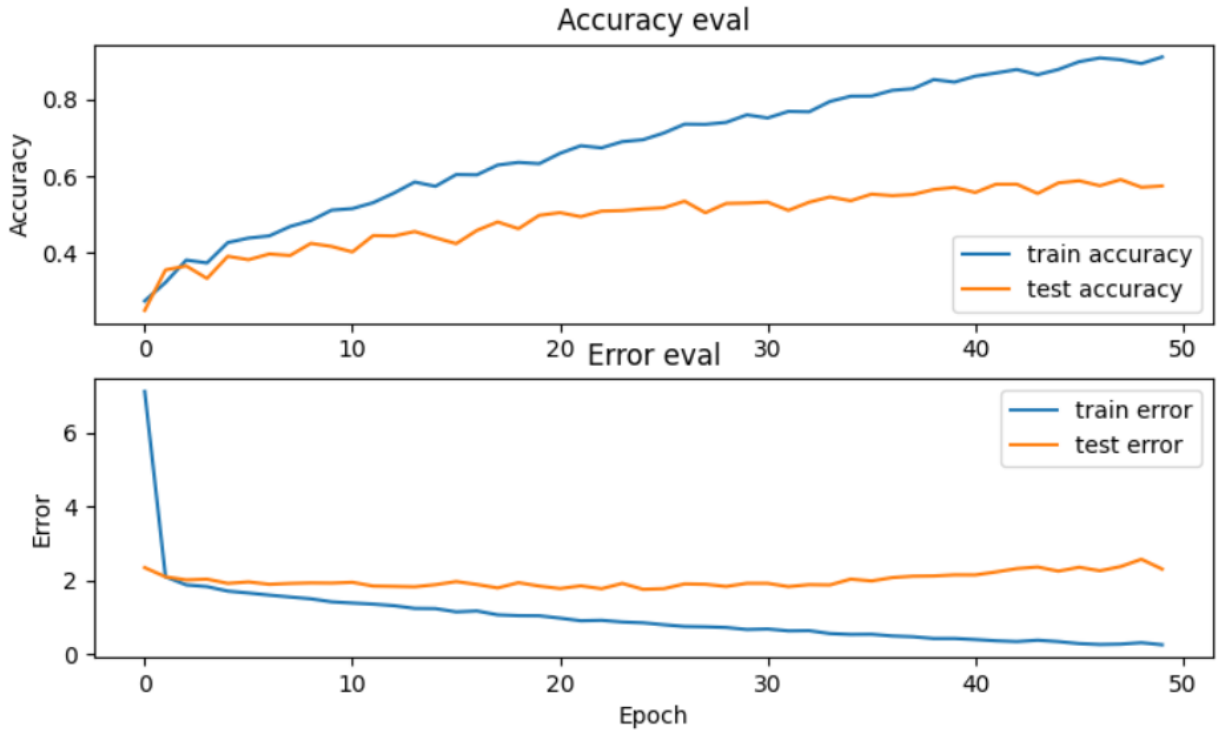

Рисунок 12 – Графики точности и ошибок

Результаты, конечно, не впечатляющие, что значит есть над чем поработать. Из графиков видно, что есть проблема переобучения. Модель хорошо объясняет примеры из обучающей выборки, но относительно плохо работает на примерах, которые не участвовали в обучении (тестовая выборка).

5.3 Предотвращение переобучения

К счастью, существуют методы решения данной проблемы и одним из этих решений является *Dropout* – исключение из сети (dropping out) некоторого процента нейронов.

Также есть метод *регуляризации* – метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу или предотвратить переобучение. Чаще всего эта информация имеет вид штрафа за сложность модели.

Применим данные метод, а также повысим количество эпох обучения (Рисунок 13):

```
model = keras.Sequential([
    # input layer
    keras.layers.Flatten(input_shape=(inputs.shape[1], inputs.shape[2])),

    # 1st hidden layer
    keras.layers.Dense(512, activation="relu", kernel_regularizer=keras.regularizers.l2(0.001)),
    keras.layers.Dropout(0.3),

    # 2st hidden layer
    keras.layers.Dense(256, activation="relu", kernel_regularizer=keras.regularizers.l2(0.001)),
    keras.layers.Dropout(0.3),

    # 3st hidden layer
    keras.layers.Dense(64, activation="relu", kernel_regularizer=keras.regularizers.l2(0.001)),
    keras.layers.Dropout(0.3),

    # output layer
    keras.layers.Dense(10, activation="softmax")
])
```

Рисунок 13 – Добавление Dropout и регуляризации

Посмотрим на результаты модификаций (Рисунок 14).

Нейросеть после 100 эпох обучения выдаёт точность:

– 63.15 % на обучающей выборке (accuracy);

– 58.62 % на тестовой выборке (val_accuracy).

```
Epoch 100/100
219/219 [==============================] - 3s 15ms/step - loss: 1.2898 - accuracy: 0.6315 - val_loss: 1.5979 - val_accuracy: 0.5862
```

Рисунок 14 – Результат обучения нейросети

На рисунке ниже представлены графики точности и ошибок на тренировочных и валидационных данных:

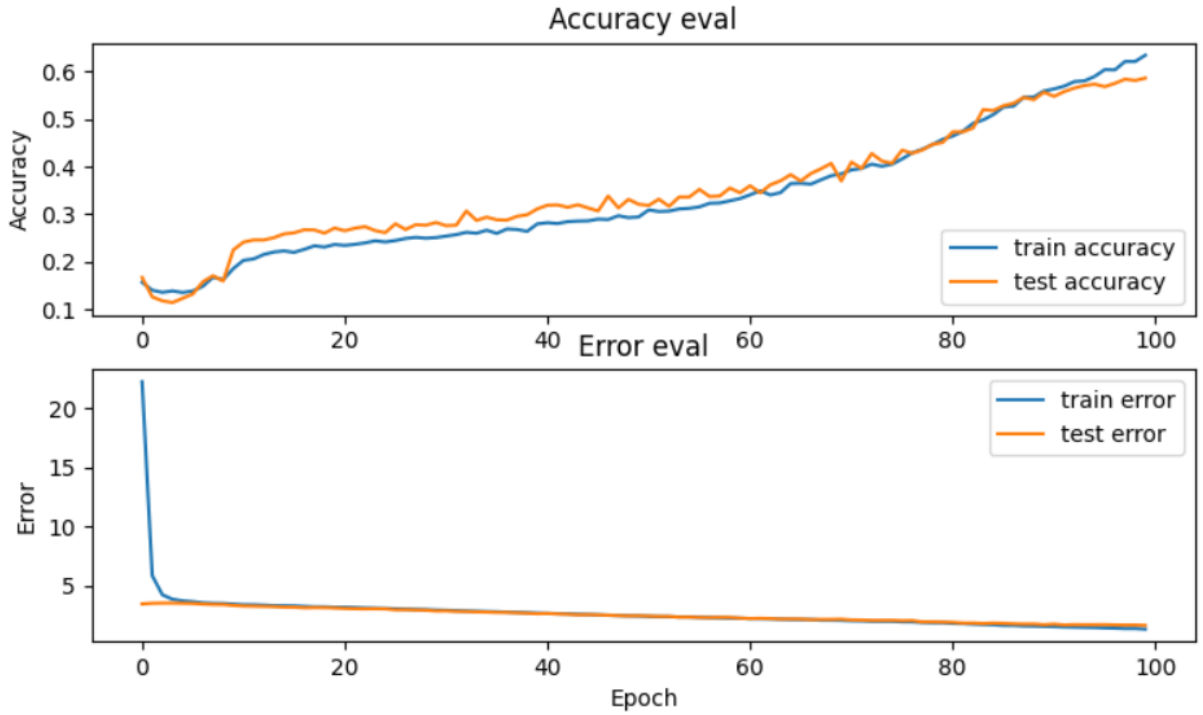

Рисунок 15 – Графики точности и ошибок

Как видим из результатов обучения, нейросеть всё ещё слабо обучена, но, как видно графиков выше, теперь решена проблема переобучения.

5.4 Построение нейронной сети. Свёрточная сеть

Для улучшения результатов изменим архитектуру нейронной сети. Построим свёрточную нейронную сеть, архитектура которой приведена на рисунке ниже (Рисунок 16).

Свёрточные нейронные сети (CNN) похожи на обычные нейронные сети: они также состоят из нейронов с весами и сдвигами. Каждый нейрон получает входные данные, выполняет скалярное произведение и, если нужно, добавляет нелинейную оптимизацию. Они также обладают функцией потерь (SVM/Softmax) на последнем, полносвязном, слое [6].

Различие архитектуры CNN в том, что они предполагают, что входные данные являются изображениями, благодаря чему определенные свойства могут быть зашифрованы в архитектуру.

Сеть имеет следующие слои:

– 3 свёрточных слоя с функцией активации *ReLU*, *MaxPool2D* и слои пакетной нормализации *BatchNormalization*;

– выравнивающий слой *Flatten*;

– полносвязный слой с 64 нейронами и функцией активации *ReLU*;

– слой *Dropout*;

– в выходном слое 10 нейронов – по количеству жанров. Функция активации на выходном слое – *SoftMax*.

```
    # create model
    model = keras.Sequential()

    # 1st conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 2nd conv layer
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPool2D((3, 3), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # 3nd conv layer
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu', input_shape=input_shape))
    model.add(keras.layers.MaxPool2D((2, 2), strides=(2, 2), padding='same'))
    model.add(keras.layers.BatchNormalization())

    # flatten the output and feed it into dense layer
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.3))

    # output layer
    model.add(keras.layers.Dense(10, activation='softmax'))
```

Рисунок 16 – Архитектура свёрточной нейросети

Проведём обучение нейросети продолжительностью 30 эпох. Результаты представлены на рисунке ниже (Рисунок 17):

```
Epoch 30/30
188/188 [==============================] - 10s 51ms/step - loss: 0.7463 - accuracy: 0.7366 - val_loss: 0.8338 - val_accuracy: 0.7073
```

Рисунок 17 – Результат обучения свёрточной нейросети

Нейросеть после 30 эпох обучения выдаёт точность:

– 73.66 % на обучающей выборке (accuracy);

– 70.73 % на тестовой выборке (val_accuracy).

На рисунке ниже представлены графики точности и ошибок на тренировочных и валидационных данных (Рисунок 18):

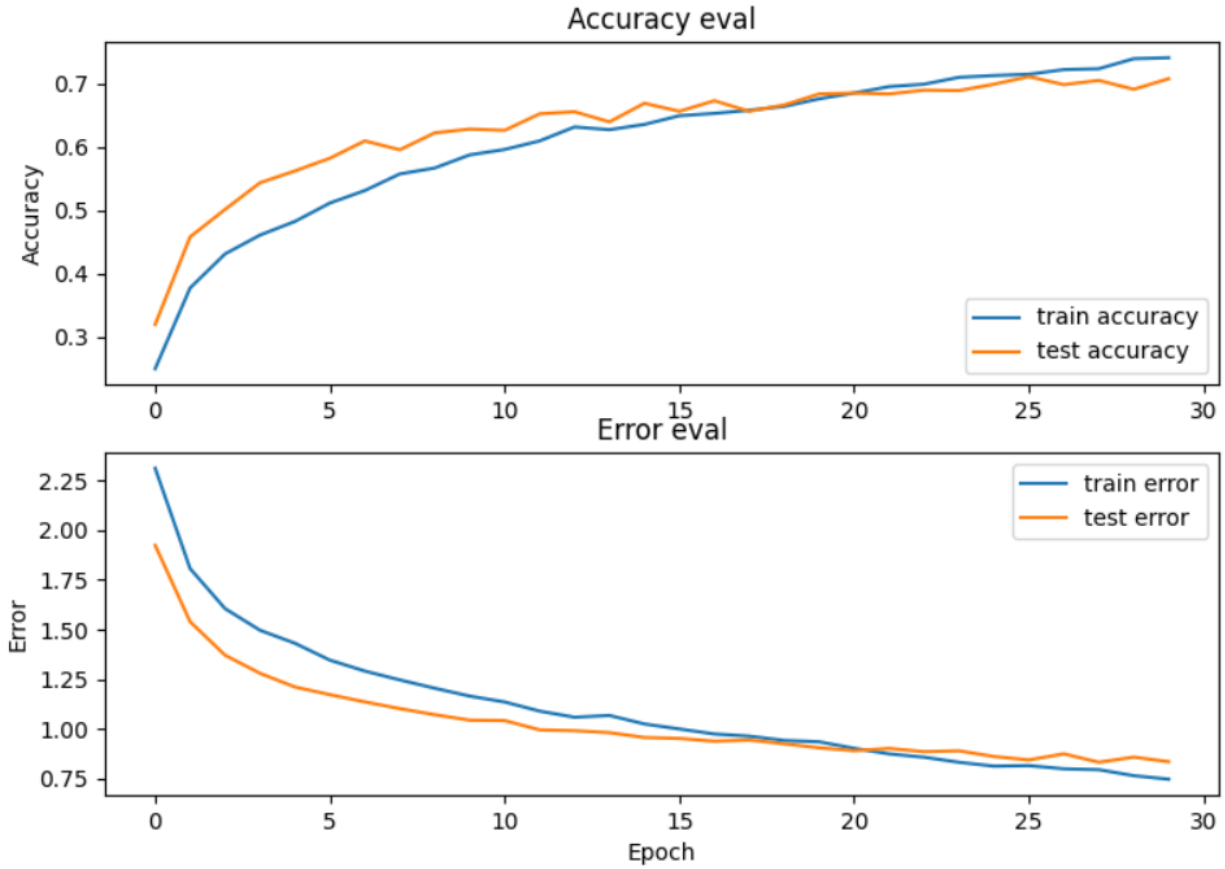

Рисунок 18 – Графики точности и ошибок

Как видим из результатов обучения, нейросеть обучена намного лучше, чем в предыдущем случае, а также, как видно графиков выше, отсутствует проблема переобучения.

5.5 Построение нейронной сети. Рекуррентная сеть. Долгая краткосрочная память

Рекуррентные нейронные сети (РНС) – вид искусственных нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому есть возможность обрабатывать последовательные пространственные цепочки или серии событий во времени.

В отличие от многослойных персептронов, РНС могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети рекуррентные нейросети применимы в таких задачах, где что-то целостное разбито на некоторые части, например: распознавание речи или рукописного текста.

Было предложено немало разных архитектур для рекуррентных сетей от простых до более сложных. В последнее время наибольшее распространение получили LSTM (сеть с долговременной и кратковременной памятью) и GRU (управляемый рекуррентный блок).

Долгая краткосрочная память (LSTM) – разновидность архитектуры рекуррентных нейронных сетей, предложенная в 1997 году Зеппом Хохрайтером и Юргеном Шмидхубером [9].

Как и большинство рекуррентных нейросетей, LSTM универсальна в том смысле, что при достаточном количестве элементов нейросети она может выполнить любое вычисление, которое может выполнить обычный компьютер. В отличие от традиционных RNN, LSTM хорошо приспособлена к обучению на задачах обработки, классификации и прогнозирования временных рядов в тех случаях, в которых значимые события разделены временными лагами с неопределённой продолжительностью и границами.

Для улучшения результатов изменим архитектуру нейронной сети. Построим сеть LSTM – долгая краткосрочная память – разновидность рекуррентных нейронных сете, архитектура которой приведена на рисунке ниже (Рисунок 19).

Сеть имеет следующие слои:

– 2 слоя LSTM;

– полносвязный слой с 64 нейронами и функцией активации *ReLU*;

– слой *Dropout*;

– в выходном полносвязном слое 10 нейронов – по количеству жанров. Функция активации на выходном слое – *SoftMax*.

```
47      # create model
48      model = keras.Sequential()
49
50      # 2 LSTM layers
51      model.add(keras.layers.LSTM(64, input_shape=input_shape, return_sequences=True))
52      model.add(keras.layers.LSTM(64))
53
54      # dense layer
55      model.add(keras.layers.Dense(64, activation='relu'))
56      model.add(keras.layers.Dropout(0.3))
57
58      # output layer
59      model.add(keras.layers.Dense(10, activation='softmax'))
```

Рисунок 19 – Архитектура RNN-LSTM нейросети

Проведём обучение нейросети продолжительностью 50 эпох. Результаты представлены на рисунке ниже (Рисунок 20):

```
Epoch 50/50
188/188 [==============================] - 30s 157ms/step - loss: 0.6919 - accuracy: 0.7711 - val_loss: 0.9863 - val_accuracy: 0.6800
```

Рисунок 20 – Результат обучения LSTM нейросети

Нейросеть после 30 эпох обучения выдаёт точность:

– 77.11 % на обучающей выборке (accuracy);

– 68 % на тестовой выборке (val_accuracy).

На рисунке ниже представлены графики точности и ошибок на тренировочных и валидационных данных (Рисунок 21):

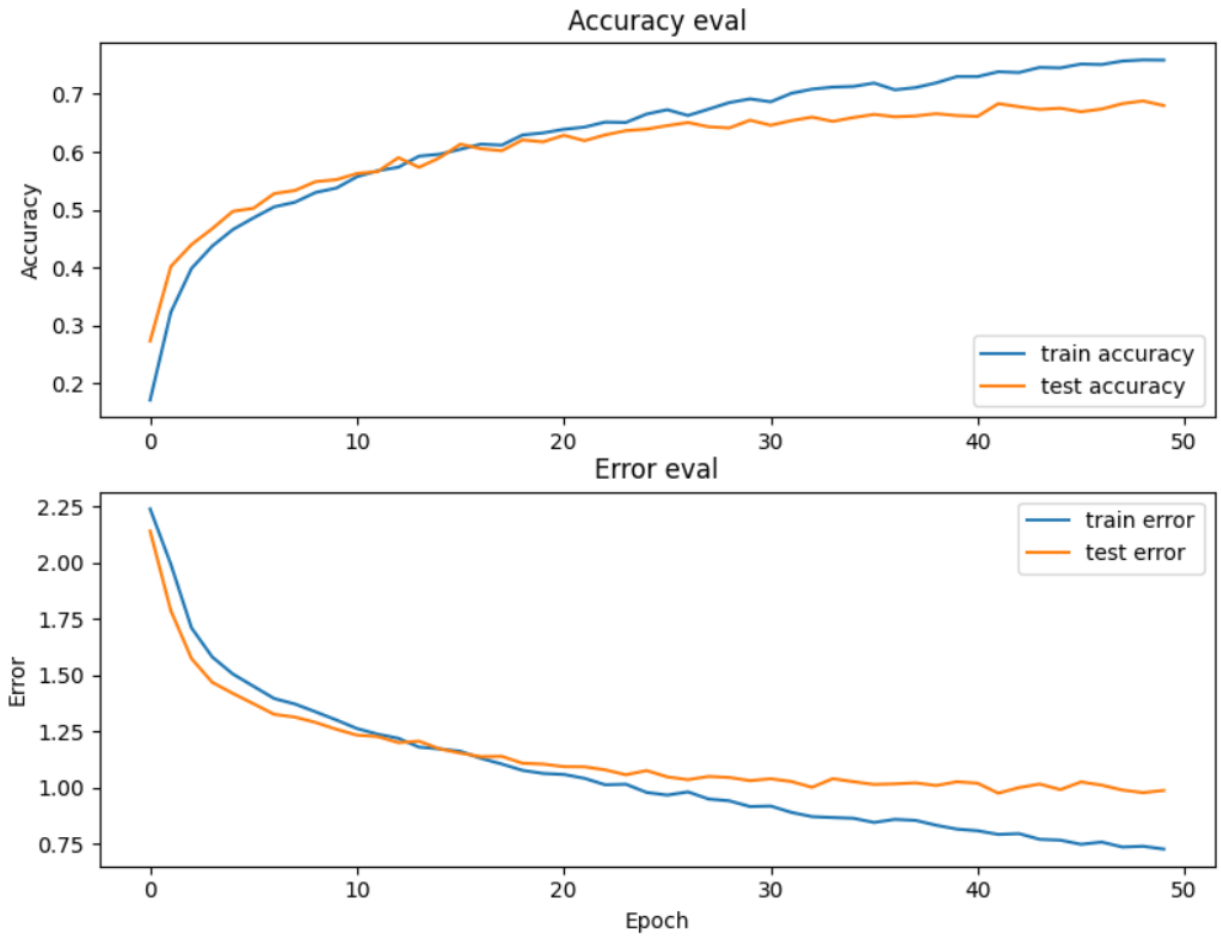

Рисунок 21 – Графики точности и ошибок

Как видим из результатов обучения, нейросеть обучена чуть хуже, чем в предыдущем случае, а также, как видно графиков выше, отсутствует проблема переобучения.

6. Сравнение построенных архитектур

В таблице ниже приведено сравнение построенных архитектур нейронной сети (Таблица 1).

Таблица 1 – Сравнение архитектур нейросети

Архитектура	Точность на обучающей выборке (%)	Точность на тестовой выборке (%)
Персептрон	91.44	57.45
Персептрон с Dropout	63.15	58.62
Свёрточная сеть	73.66	70.73
Рекуррентная сеть. Долгая краткосрочная память	77.11	68

Как видно из таблицы, наиболее подходящей архитектурой нейронной сети является свёрточная сеть. Она лучше остальных архитектур проявила себя в точности на тестовой выборке (70.73 %). Рекуррентная сеть также показывает относительно неплохие результаты. У неё 68 % точности на тестовой выборке.

Хуже всех проявил себя персептрон – присутствовала проблема переобучения. Однако, добавление слоёв Dropout и регуляризация решили эту проблему.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы разработана нейронная сеть для распознавания жанров музыкальных композиций. Все задачи, поставленные для достижения этой цели, решены, а именно:

– рассмотрены методы решения поставленной задачи;

– спроектировано 4 архитектуры нейросети;

– написана нейронная сеть, распознающая жанры музыкальных композиций;

– проведено сравнение 4-х разработанных архитектур нейронной сети.

Данная тема актуальна, т. к. нет сервиса, которых предоставлял бы пользователям возможность напрямую определить жанр какой-либо музыкальной композиции.

И в конце концов, всегда можно сделать лучше, чем уже есть. Ведь совершенных программных продуктов не существует.

Лучшим образом в данной задаче проявила себя свёрточная нейронная сеть (CNN). CNN – это хорошая альтернатива для извлечения признаков, что подтверждает предположение о том, что внутренние характеристики вариаций аудиоданных аналогичны характеристикам данных изображений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Нейросетевое распознание жанра музыкальных произведений // (Рус.). – URL: https://hub.exponenta.ru/post/neyrosetevoe-raspoznavanie-zhanra-muzykalnykh-proizvedeniy-music-genre-recognition-with-neural-networks576 (дата обращения: 06.06.2021).

2 Солдатова О. П. Использование многослойного персептрона для распознавания жанров музыкальных композиций // Известия Самарского научного центра Российской академии наук – 2016. (Рус.). – URL: https://cyberleninka.ru/article/n/ispolzovanie-mnogosloynogo-perseptrona-dlya-raspoznavaniya-zhanrov-muzykalnyh-kompozitsiy (дата обращения: 06.06.2021).

3 Deep Learning (for audio) with Python // (Engl.) – URL: https://www.youtube.com/playlist?list=PL-wATfeyAMNrtbkCNsLcpoAyBBRJZVlnf (дата обращения: 13.06.2021).

4 Маркина Ю. Ю., Белов Ю. С. Кепстральные коэффициенты как необходимая характеристика процесса создания системы имитации голоса человека с помощью методов глубокого обучения // Международный студенческий научный вестник. – № 1, 2018. (Рус.). – URL: https://eduherald.ru/ru/article/view?id=18125 (дата обращения: 07.06.2021).

5 Анализ аудиоданных с помощью глубокого обучения и Python (часть 1) // (Рус.). – URL: https://nuancesprog-ru.turbopages.org/nuancesprog.ru/s/p/6713/ (дата обращения: 14.06.2021).

6 Анализ аудиоданных с помощью глубокого обучения и Python (часть 2) // (Рус.). – URL: https://nuancesprog.ru/p/6758/ (дата обращения: 14.06.2021).

7 MIREX 2020: Audio Train/Test: Genre Classification (Mixed) - MIREX08 Dataset // (Engl.). – URL: https://www.music-ir.org/nema_out/mirex2020/results/act/mixed_report/summary.html (дата обращения: 14.06.2021).

8 Многослойный персептрон // (Рус.). – URL: https://wiki.loginom.ru/articles/multilayered-perceptron.html (дата обращения: 16.06.2021).

9 Рекуррентная нейронная сеть // (Рус.). – URL: https://ru.wikipedia.org/wiki/Рекуррентная_нейронная_сеть (дата обращения: 16.06.2021).

10 Спектрограмма // (Рус.). – URL: https://ru.wikipedia.org/wiki/Спектрограмма (дата обращения: 16.06.2021).