УДК 004

Классификация звуков с помощью библиотек TensorFlow и Librosa в Python

Эрдман Александр Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрен процесс создания программы на языке программирования Python. Данная программа является алгоритмом машинного обучения и использует ключевые библиотеки TensorFlow и Librosa. Алгоритм обрабатывает звуковой файл, производит сравнение и определение звука на основе тестовых данных.
Ключевые слова: Python, TensorFlow, машинное обучение, Librosa

Classification of sounds using TensorFlow and Librosa libraries in Python

Erdman Alexander Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article describes the process of creating a program in the Python programming language. This program is a machine learning algorithm and uses the TensorFlow and Librosa libraries. The algorithm processes the audio file, compares and determines the sound based on the test data.
Keywords: python, TensorFlow, machine learning, Librosa

1 Введение

1.1 Актуальность

В настоящие время машинное обучение и обработка данных являются существенным инструментом во многих сферах деятельности людей, начиная от научных исследования и заканчивая анализом статистики бизнес проектов. Практически все сферы, где задействованы данные, используется машинное обучение и анализ на основе искусственного интеллекта. Часто ярким результатом активного использования машинного обучения можно считать предсказание цен на биржах или изменения спроса потребителей на основе конкретных признаков в бизнесе. В научных же целях машинное обучение используются для предсказания тех или иных погодных явлений, результатов экспериментов или предсказание срока эксплуатации того или иного материала. Помимо масштабных проектов в науке и бизнесе, машинное обучение активно используют энтузиасты в создание, например изображений, что на сегодняшний день считается крайне популярным

занятием. Другим примером можно считать анализ и обработку звуковых файлов, выявление их явных признаков, на основе которых можно отнести звуковой файл к конкретному классу звука. Определение типа звука будет рассмотрено в данной статье, в качестве возможности машинного обучения обрабатывать и классифицировать звуковые данные.

1.2 Обзор исследований

Р.Л. Астапов и Р.М. Мухамадеева рассмотрели вопросы машинного обучения, а именно задача автоматизации подбора параметров машинного обучения и обучение модели с помощью кросс-валидации [1]. П.М. Горбунов, Ю.А. Мацкевич и А.В. Чубарь привели объяснение различных видов функционирования машинного обучения [2]. С.Ж. Дуалет реализовал разделения информацию на части в социальной сети Twitter в среде Python [3]. В.А. Мельникова и Д.А. Медведев провели сравнительный анализ библиотек языка программирования Python для решения задач анализа данных [4]. Е.А. Григорьев и Н.С. Климонов рассмотрели и описали разведочный анализ данных с помощью Python [5]. О.А. Ермаков и Н.П. Брозгунова рассмотрели язык программирования возможности языка Python для анализа данных [6].

1.3 Цель исследования

Целью исследования является создание программы для определения звуков и его классификацию на основе dataset.

2 Материалы и методы

Для создания программы используется IDE Google Colab. Подключённая библиотека TensorFlow и Librosa. DataSet с классификацией звуков.

3 Результаты и обсуждения

Перед тем, как приступить к реализации классификации и определения звука, нужно звук сначала преобразовать. Программа не может напрямую обрабатывать звук и тем более его определять. Звук нужно представить в другом виде. Звук – это колебания, которые называются акустической волной. Волну можно измерить. Для этого используется частота, скорость, амплитуда и направление волны. Конкретно в задаче машинного обучения, существенную роль играет только частота и амплитуда. В создаваемой программе будут использована библиотека Librosa для анализа музыки. Благодаря данной библиотеки можно получить данные музыкального файла и его частоту дискретизации, на основе которой и будет происходить машинное обучение.

Запускаем web-сервис Google Colab, в данном сервисе будет происходить разработка программы. Далее необходимо импортировать библиотеку Librosa, а также дополнительные необходимые модули для работы:

```
import pandas as pd
import os
import librosa
import librosa.display
import numpy as np
import IPython.display as ipd
import matplotlib.pyplot as plt
%matplotlib inline
```

Данные берутся с открытых источников интернета. Далее файл загружается в папку проекта Google Colab. В конечном итоге получены два файла: первый содержит только аудиоданные, которые представлены звуками города, а второй представлен .csv и содержит необходимые данные о каждом звуке. После загрузки файлов и их подключения, можно взять произвольный звуковой файл. Выберем аудиофайл с играющими детьми. С помощью Librosa извлечём данные audio_data и sampling_rate (рис. 1).

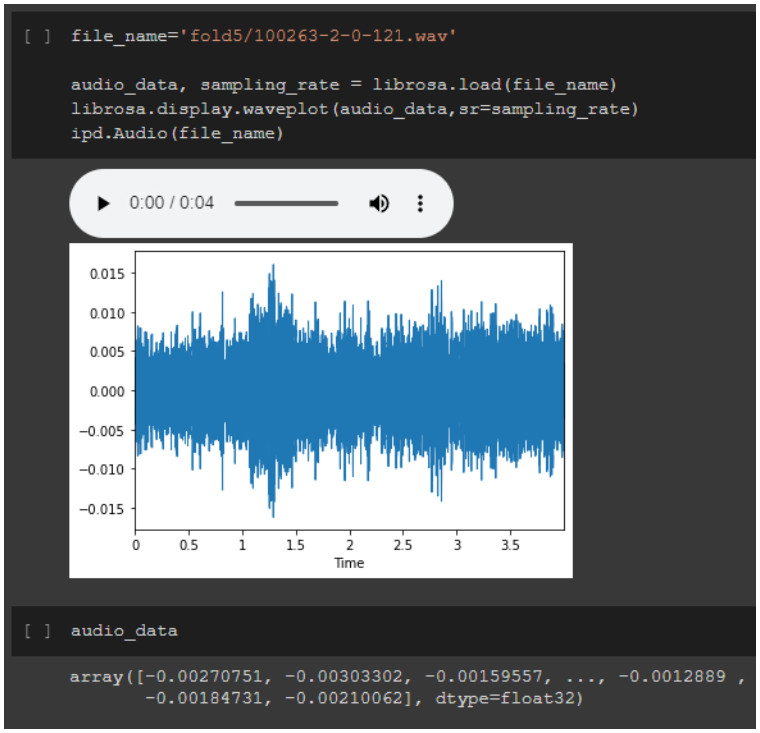
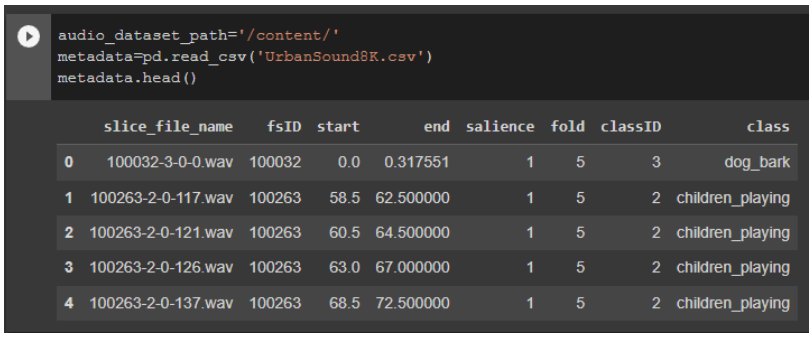


Рисунок 1. Вывод данных о звуке с помощью библиотеки Librosa

Далее выведем данные о аудиофайле. Можно видеть, что Librosa преобразовала файл в моносигнал. Всё верно, так как звуки из dataset являются монозвуками. Моно звук используется Librosa для упрощения дальнейшей обработки. С помощью команды sampling_rate можно узнать частоту дискретизации. По умолчанию библиотека использует дискретизацию 22050. С помощью библиотеки pandas открываем для чтения .csv файл:

```
audio_dataset_path='/content/'
metadata=pd.read_csv('UrbanSound8K.csv')
metadata.head()
```

После открытия dataset загружаем UrbanSound8K.csv файл, доступный в нашей папке загруженного набора данных. Затем идёт сохранение его в переменной, известной как metadata. Далее можно использовать метод head() для просмотра таблицы данных (рис. 2).



```
audio_dataset_path='/content/'
metadata=pd.read_csv('UrbanSound8K.csv')
metadata.head()
```

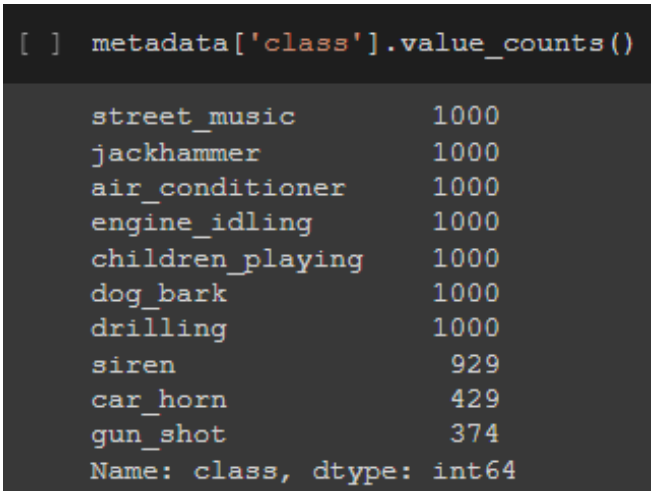
	slice_file_name	fsID	start	end	salience	fold	classID	class
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing

Рисунок 2. Открытие и считывание данных с dataset

Данные также организованы в соответствующие классы файлов. Набор данных не должен быть несбалансированным. Для просмотра сбалансированности данных используем следующую команду:

```
metadata['class'].value_counts()
```

после выполнения программы можно увидеть результат (рис. 3).



```
[ ] metadata['class'].value_counts()

street_music        1000
jackhammer          1000
air_conditioner     1000
engine_idling       1000
children_playing    1000
dog_bark            1000
drilling            1000
siren                929
car_horn             429
gun_shot             374
Name: class, dtype: int64
```

Рисунок 3. Проверка сбалансированности данных

Результаты показывают, что большинство классов в наборе данных сбалансированы. Таким образом, это будет хороший набор данных для анализа. Данные предварительно обрабатываются, чтобы извлечь значимые функции. Затем извлеченные функции используются для обучения вместо использования данных в необработанном виде.

Чтобы извлечь функции, будет использоваться алгоритм кепстральных коэффициентов Mel-Frequency (MFCC). Алгоритм MFCC суммирует частотное распределение по размеру окна. Это позволяет анализировать как частотные, так и временные характеристики предоставленного звука. Это позволит определить признаки для классификации:

mfccs = librosa.feature.mfcc(y=audio_data, sr=sampling_rate, n_mfcc=40)

После выполнения команды mfccs будут получены шаблоны из одного аудиофайла на основе частотных и временных характеристик (рис. 4).

```
[ ]  mfccs

     array([[-4.6613168e+02, -4.6417816e+02, -4.7455182e+02, ...,
             -4.4540848e+02, -4.5221939e+02, -4.5637799e+02],
            [ 1.0846554e+02,  1.1128984e+02,  1.0955853e+02, ...,
              1.1160173e+02,  1.1063791e+02,  1.1319142e+02],
            [-2.5252140e+01, -2.7399439e+01, -3.2546665e+01, ...,
             -3.8440331e+01, -3.4312595e+01, -3.5521683e+01],
            ...,
            [ 2.3573508e+00,  1.6371250e+00,  3.2692363e+00, ...,
              7.8856702e+00,  1.0755114e+01,  1.1197763e+01],
            [-3.2311397e+00, -2.6380532e+00,  4.6177328e-01, ...,
              1.0223865e+01,  1.1984882e+01,  1.3385002e+01],
            [-1.3852274e+01, -1.0576165e+01, -2.1510942e+00, ...,
              2.9695926e+00,  2.1894133e+00,  6.6635776e-01]], dtype=float32)
```

Рисунок 5. Шаблоны, которые были извлечены из аудио на основе частотных и временных характеристик.

Чтобы извлечь функции из всех аудиофайлов в наборе данных, создаем список для хранения всех извлеченных функций (рис. 6).

```
[ ]  def features_extractor(file):
         audio, sample_rate = librosa.load(file_name, res_type='kaiser_fast')
         mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
         mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)

         return mfccs_scaled_features
```

Рисунок 6. Создание списка для хранения функций

Затем перебирается каждый аудиофайл и извлекается функция, используя кепстральные коэффициенты Mel-Frequency (рис. 7).

```
[ ]  extracted_features=[]
     for index_num,row in tqdm(metadata.iterrows()):
         file_name = os.path.join(os.path.abspath(audio_dataset_path),'fold'+str(row["fold"])+'/',str(row["slice_file_name"]))
         final_class_labels=row["class"]
         data=features_extractor(file_name)
         extracted_features.append([data,final_class_labels])
```

Рисунок 7. Извлечение функций

Далее следует преобразовать весь список во фрейм данных с помощью библиотеки Pandas. Данный процесс представляет результаты в таблицы для более простого анализа (рис. 8).

```
[ ] extracted_features_df=pd.DataFrame(extracted_features,columns=['feature','class'])
    extracted_features_df.head(10)
```

	feature	class
0	[-215.79301, 71.66612, -131.81377, -52.09133, ...	dog_bark
1	[-424.68677, 110.56227, -54.148235, 62.01074, ...	children_playing
2	[-459.56467, 122.800354, -47.92471, 53.265705,...	children_playing
3	[-414.55377, 102.896904, -36.66495, 54.18041, ...	children_playing
4	[-447.397, 115.0954, -53.809113, 61.60859, 1.6...	children_playing
5	[-447.70856, 118.409454, -35.24866, 56.73993, ...	children_playing
6	[-477.1972, 120.63773, -29.692501, 57.051914, ...	children_playing
7	[-464.84656, 117.71454, -30.163269, 50.72254, ...	children_playing
8	[-472.1215, 126.76601, -38.36653, 58.748646, -...	children_playing
9	[-196.18527, 114.94506, -14.661183, 1.2298629,...	car_horn

Рисунок 8. Извлечённые функции и их соответсвуюшие классы

Следующая команда (функция extract() возвращает элементы массива) разбивает набор данных на независимые и зависимые наборы данных, X и y:

```
X=np.array(extracted_features_df['feature'].tolist())
y=np.array(extracted_features_df['class'].tolist())
```

X используется заглавная, так как принято обоозначать двумерные массивы с заглавной буквы, а одномерные с маленьких, как пример y.

Затем идёт импорт оба метода из TensorFlow to_categorical и LabelEncoder Sklearn:

```
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
y=to_categorical(labelencoder.fit_transform(y))
```

LabelEncoder — служебный класс, помогающий нормализовать метки таким образом, чтобы они содержали только значения от 0 до n_classes-1. Используется для преобразования классов в числовые значения.

Этот шаг включает в себя использование метода sklearn_train_test_split для разделения набора данных на обучающие и тестовые наборы:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

После предварительных установок можно приступить к созданию модели (рис. 9). Слои модели будут последовательно сложены. Последний уровень будет иметь уровень активации softmax:

```
model=Sequential()
###первый слой
model.add(Dense(100,input_shape=(40,)))
```

```
model.add(Activation('relu'))
model.add(Dropout(0.5))
###второй слой
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###третий слой
model.add(Dense(100))
model.add(Activation('relu'))
model.add(Dropout(0.5))
###заключительный слой
model.add(Dense(num_labels))
model.add(Activation('softmax'))
```

Softmax функция активации используется последней для нормализации выходных данных. Использование других функций нормализации с Softmax выдаст выходные данные в вероятностной форме.

Функция активации ReLU используется для всех слоев, кроме выходного.

model.add(Dense(100, input_shape=(40,)))теперь модель будет принимать в качестве входных массивов фигуры (*, 40)# и выходных массивов фигуры (*, 100)

Слой Dense - это слой, выходные нейроны которого связаны со всеми входными нейронами, в котором каждый нейрон предыдущего слоя связан с каждым нейроном данного. Параметры dense:

Первый параметр – выходной массив нейронов (100, чем больше нейронов, тем больше чувствительность сети к изменению), второй параметр (input_shape) – входной массив нейронов (40, так как у нас такое количество параметров, полученных с помощью librosa).

Параметр Activation отвечает за активацию функций нормализации.

Параметр Dropout (метод регуляризации переобучения сети) выключает нейроны с вероятностью p и оставляет их включенными с вероятностью q=1-p.

Теперь можно приступить к обучению модели. Чем больше число эпох, тем больше увеличивается точность. Рассмотрим модель с 200 эпохами (рис. 10).

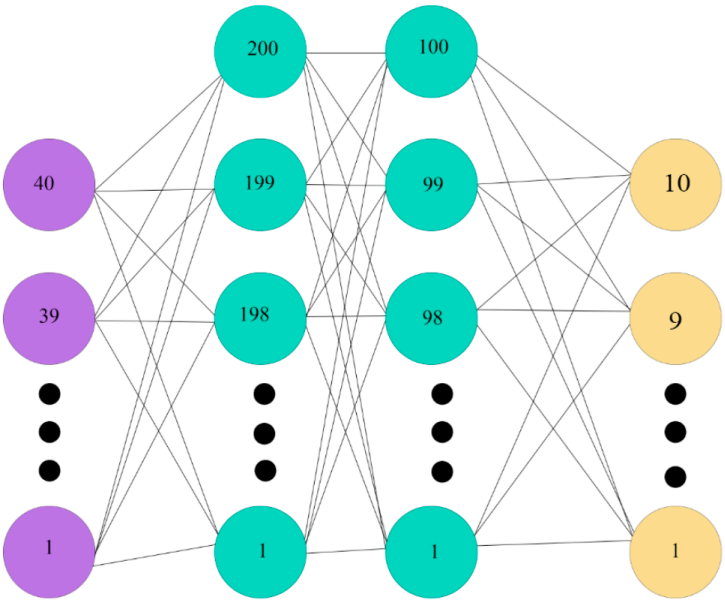


Рисунок 9. Модель нейросети

```
from tensorflow.keras.callbacks import ModelCheckpoint
from datetime import datetime

num_epochs = 200
num_batch_size = 32

checkpointer = ModelCheckpoint(filepath='saved_models/audio_classification.hdf5',
                               verbose=1, save_best_only=True)
start = datetime.now()

model.fit(X_train, y_train, batch_size=num_batch_size, epochs=num_epochs, validation_data=(X_test, y_test), callbacks=[checkpointer]


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

Рисунок 10. Обучение модели

Можно посмотреть точность модели. Делается это с помощью следующей команды:

```
test_accuracy=model.evaluate(X_test,y_test,verbose=0)
print(test_accuracy[1])
```

Результат проверки точности имеет значение 0.7870635390281677. если переводить в проценты, то модель имеет точность в 78,71%. Данное значение удовлетворительно. Тестирование модели будет осуществляться с помощью аудиофайла с собачим лаем из набора данных. С помощью библиотеки Librosa извлекаем необходимые данные о файле. Данные нужны, для работы модели, так как на входе она принимает не аудиофайл, а данные о дискретизации звука. Использую inverse_transform метод из scikitlearn, для получения предсказания - к какому классу относится аудифайл (рис. 10). В завершении работы модели, получится результат, что файл принадлежит к классу «dog_bark», что в свою очередь является верным.

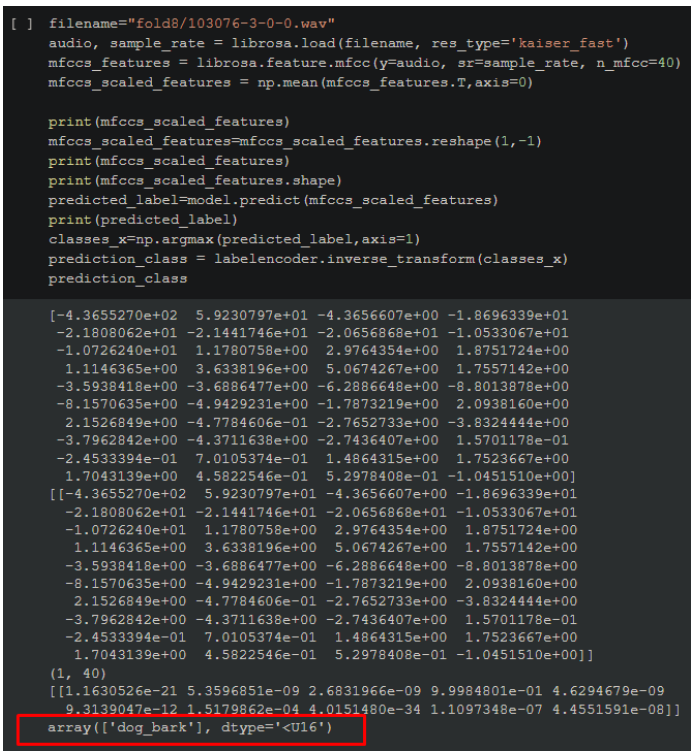


Рисунок 11. Результатат работы модели.

Для оценки точности для текущих параметров сети, используются дополнительные два звуковых файла не из загруженного dataset, то есть взятые извне. Взятые звуки: звук музыки на улицы, два аудиофайла лая собак, два аудиофайла гудка автомобиля. В результате, нейронная сеть точно определила аудиофайлы с лаем собак и звуком музыки на улицы. При этом один из аудиофайлов гудка автомобиля был отнесен к неверному классу – к лаю собак. Результаты (рис. 12,13 и 14).

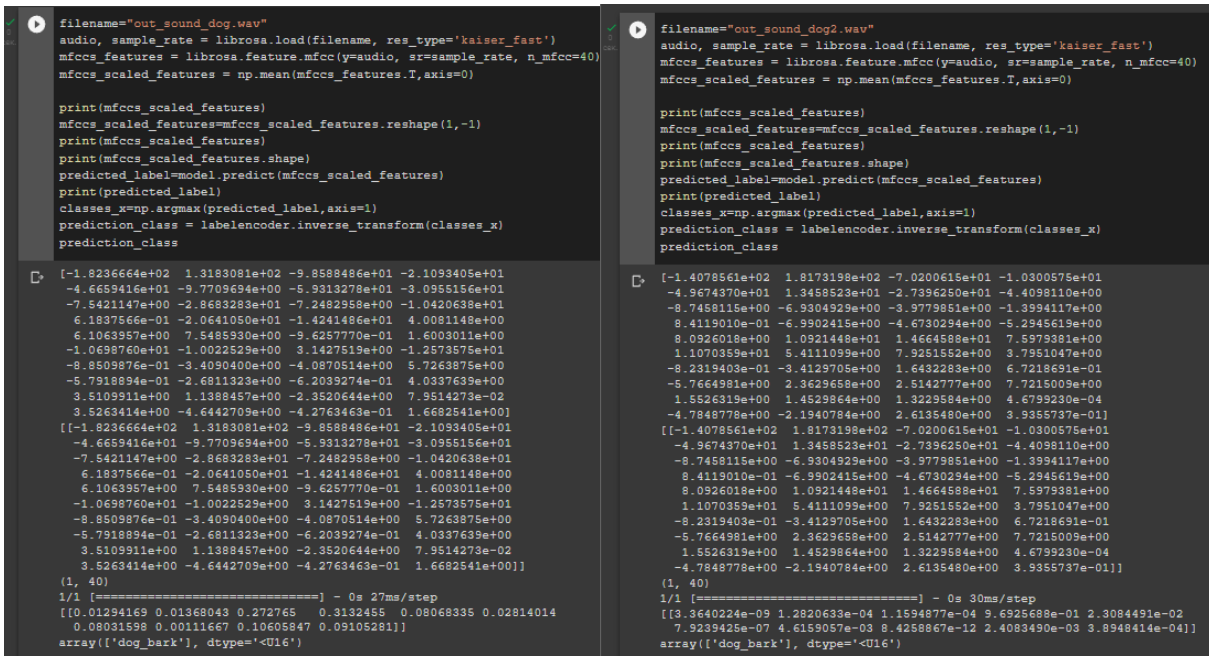


Рисунок 12. Тестирование нейронной сети на внешних аудиофайлах со лаем собак

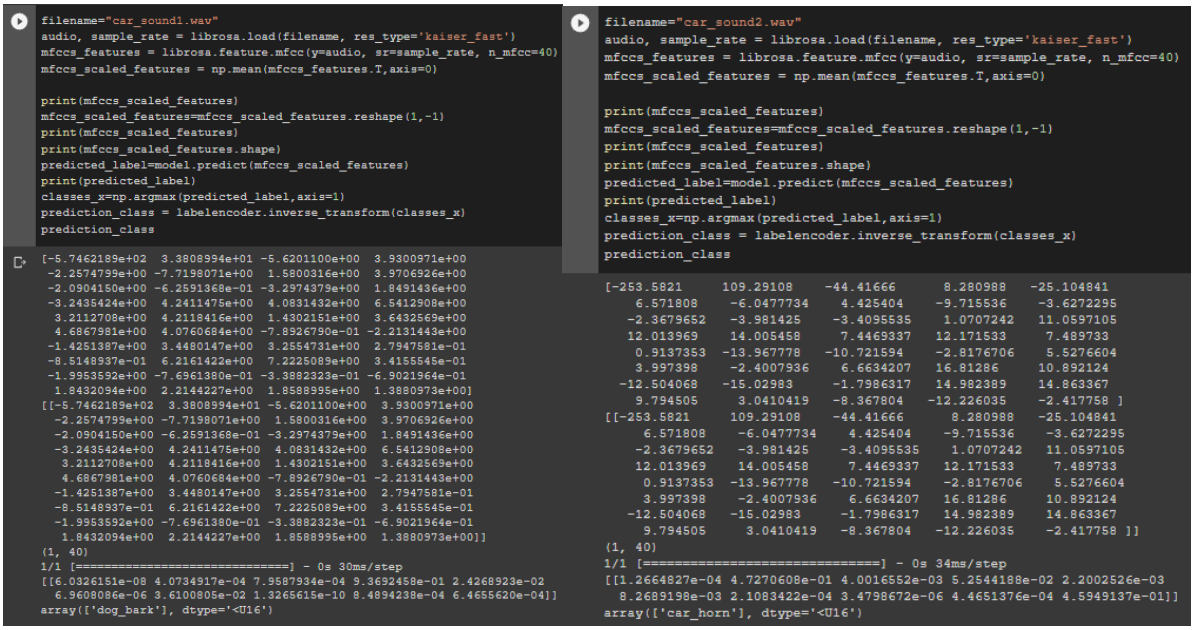


Рисунок 13. Тестирование нейронной сети на внешних аудиофайлах со гудком автомобиля

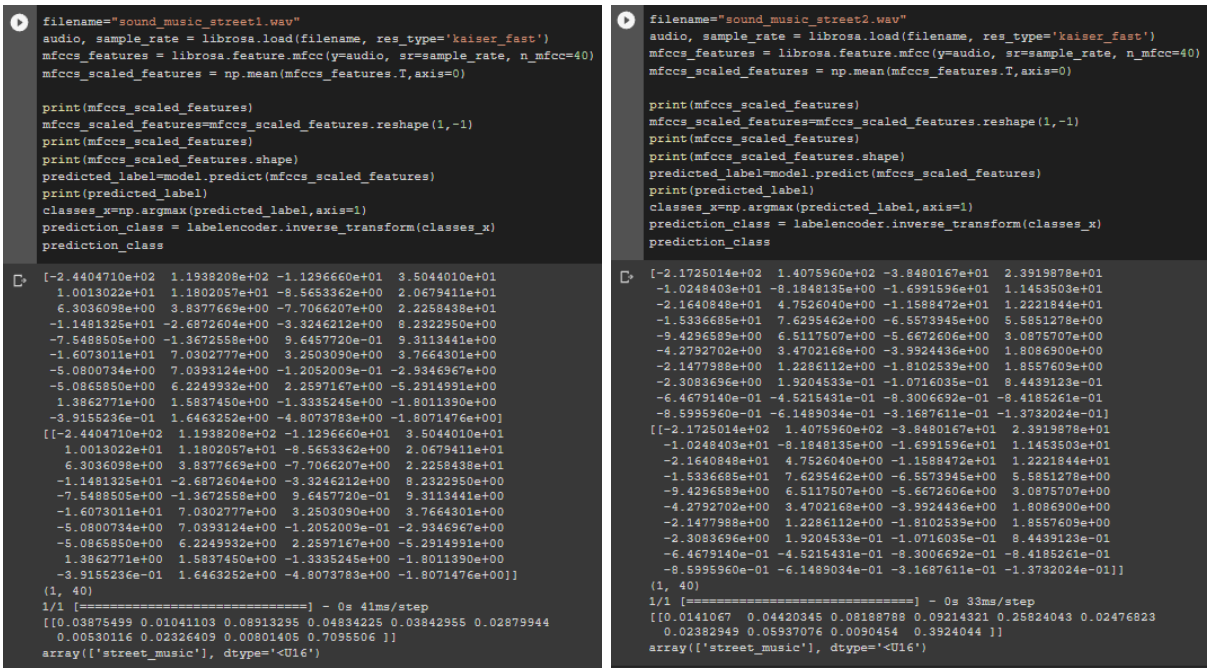


Рисунок 14. Тестирование нейронной сети на внешних аудиофайлах со звуком музыки на улице

Изменим параметры модели. Поменяем выходные нейроны у параметра Dense со 100 до 50. Количество эпох (сколько раз алгоритм обучения будет работать со всем набором обучающих данных) останется неизменным - 200. Тест нейронной сети пройдёт с теми же данными. В результате подстановки точность модели упала с 78,71% до 67,88%. Можно предположить, что нейронная сеть допустит больше ошибок в этот раз. Проверка работы. Лай собак распознался верно в обоих случаях, в то время как один из файлов с гудком автомобиля был определён верно, а другой не

верно, а оставшиеся аудиофайлы со звуком музыки на улице были определены полностью неправильно (рис. 15,16).

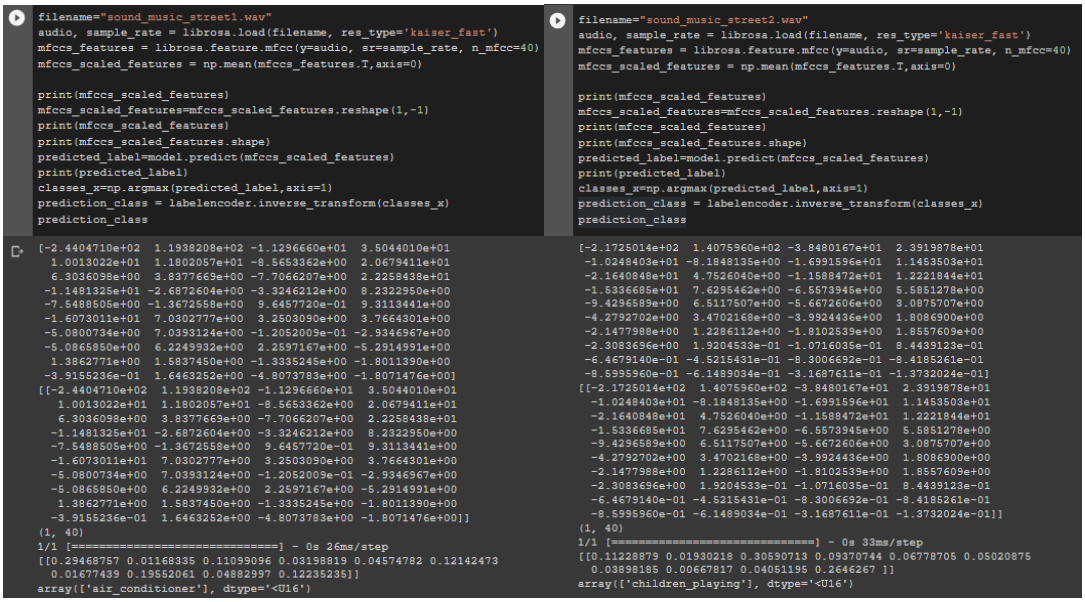


Рисунок 15. Тестирование нейронной сети с альтернативными параметрами на внешних аудиофайлах со звуком музыки на улице

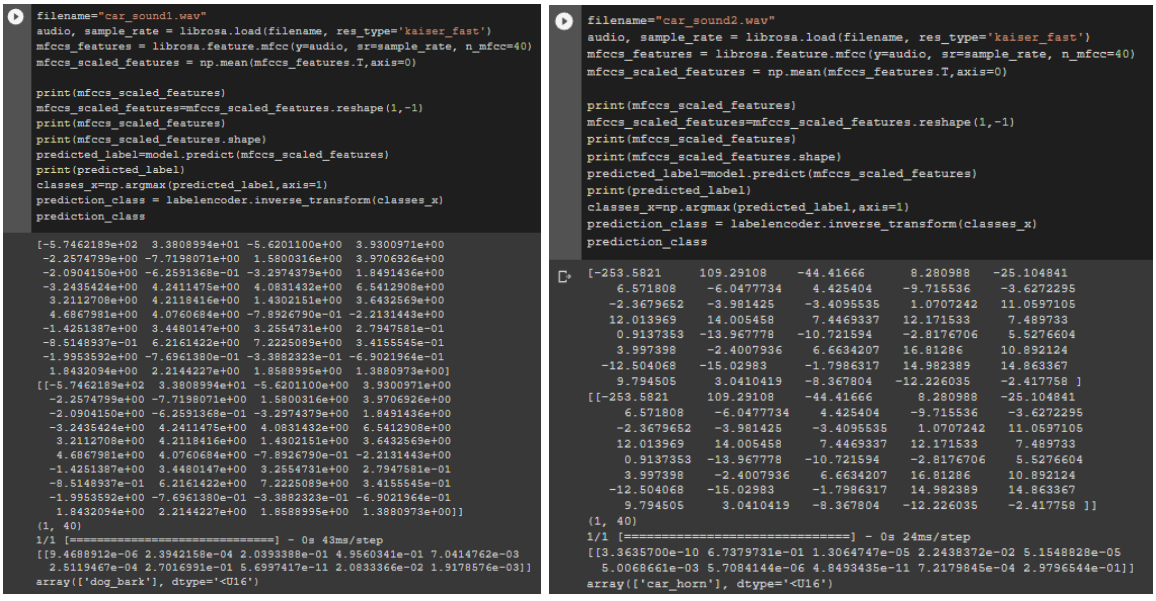


Рисунок 16. Тестирование нейронной сети с альтернативными параметрами на внешних аудиофайлах со звуком гудка автомобиля

При изменении параметров Dense в большую (со 100 нейронов до 500), увеличилась точность нейронной сети в процентах (88,49%), но также увеличилось время обучение нейронной сети. Тесты нейронных сетей одинаковы с первым тестом – наблюдается тот же самый результат, где 5 из 6 файлов распознаны правильно, а один распознан неверно, причём это тот же самый файл с гудком автомобиля. Исходя из данных тестов, можно сказать, что для решение поставленной задачи достаточно 78,71% точности нейронной сети. Увеличение нейронов в параметре Dense ведёт к

увеличению времени обучения нейронной сети, при неизменном результате относительно первого эксперементального тестирования.

Выводы

Таким образом была разработана программа с алгоритмом машинного обучения, при использовании библиотек TensorFlow и Librosa. Несмотря на точность в 78,71%, модель успешно определила к какому классу относится входящий аудиофайл из тестирующего набора, а в эксперименте с другими внешними, то есть не входящими в dataset файлами, нейронная сеть повела себя также удовлетворительно – 5 из 6 аудиофайлов были отнесены к верным классам. Уменьшение параметра Dense привело к существенным ошибкам классификации файлов, в то время как существенное увеличение параметров Dense привело к увеличению времени обучения нейронной сети, но не к улучшению нейронной сети в тестах.

Библиографический список

1. Астапов Р.Л., Мухамадеева Р.М. Автоматизация подбора параметров машинного обучения и обучение модели машинного обучения // Актуальные научные исследования в современном мире. 2021. № 5-2 (73). С. 34-37.
2. Горбунов П.М., Мацкевич Ю.А., Чубарь А.В. Машинное обучение. Автоматизация подбора модели машинного обучения // В сборнике: Робототехника и искусственный интеллект. материалы XIII Всероссийской научно-технической конференции с международным участием. Министерство науки и высшего образования Российской Федерации, Сибирский федеральный университет, Межинститутская базовая кафедра «Прикладная физика и космические технологии». 2021. С. 155-160.
3. Даулет С.Ж. Извлечение данных в twitter, предварительная обработка и анализ настроений с использованием python 3.0 // Молодой ученый. 2020. № 13 (303). С. 260-264.
4. Мельникова В.А., Медведев Д.А. Анализ больших данных с использованием python // Труды Братского государственного университета. Серия: Естественные и инженерные науки. 2019. Т. 1. С. 46-49.
5. Григорьев Е.А., Климов Н.С. Разведочный анализ данных с помощью python // E-Scio. 2020. № 2 (41). С. 165-176.
6. Ермаков О.А., Брозгунова Н.П. Python - как инструмент для анализа данных // Наука и Образование. 2020. Т. 3. № 4. С. 26.