УПРАВЛЕНИЕ ОБРАЗОВАНИЯ АДМИНИСТРАЦИИ ГОРОДСКОГО ОКРУГА СОЛНЕЧНОГОРСК МОСКОВСКОЙ ОБЛАСТИ

Муниципальное бюджетное общеобразовательное учреждение лицей № 7 города Солнечногорска

Исследовательский проект по информатике: «Создание нейросети, подбирающей музыку по предпочтениям пользователя»

Работу выполнили:

Дергунов Сергей Владимирович

Цейтлин Евгений Владимирович

10 «А» класс

Руководитель проекта:

Федорова Наталья Евгеньевна

учитель информатики

Цель работы

- Написать нейронную сеть на языке Python при помощи библиотеки Keras, которая будет распознавать образцы музыки в соответствии с жанром.
- Создать веб-интерфейс для взаимодействия пользователя и нейросети

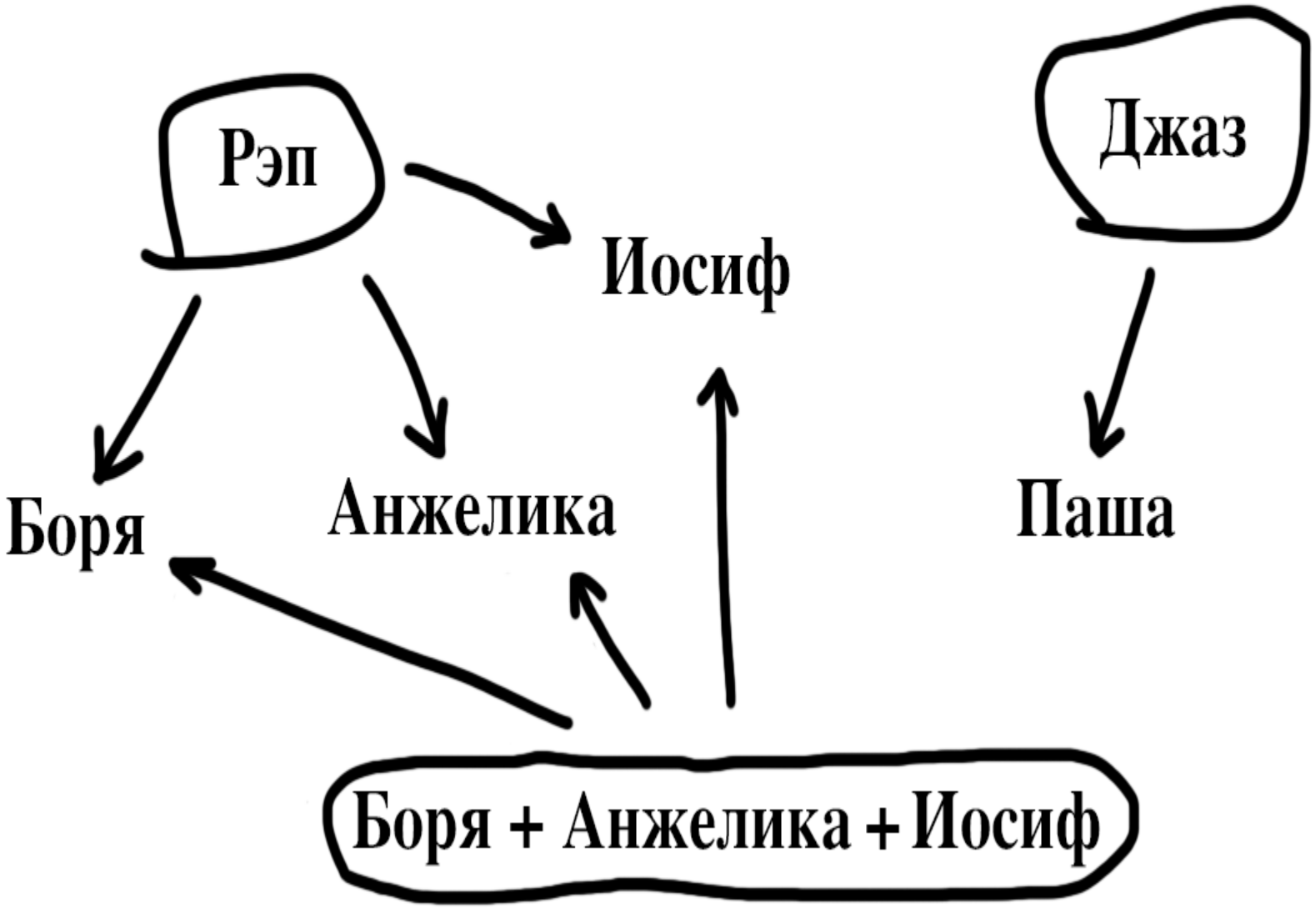
Задачи:

- Изучить архитектуру нейросети.
- Изучить алгоритм ранжирования музыки.
- Создать свой алгоритм ранжирования музыки основанный на нейросети.
- Написать программу, распределяющую Датасет на выбранные классы для обучения нейронной сети.
- Подобрать Датасет для проекта.
- Написать нейронную сеть, которая будет определять объекты, такого же класса, как из Датасета.
- Обучение нейронной сети классификации предложенных объектов.
- Оценка точности обученной нейросети (%).
- Отладка и тестирование полученной нейронной сети.
- Написание интерфейса пользователя (сайта)

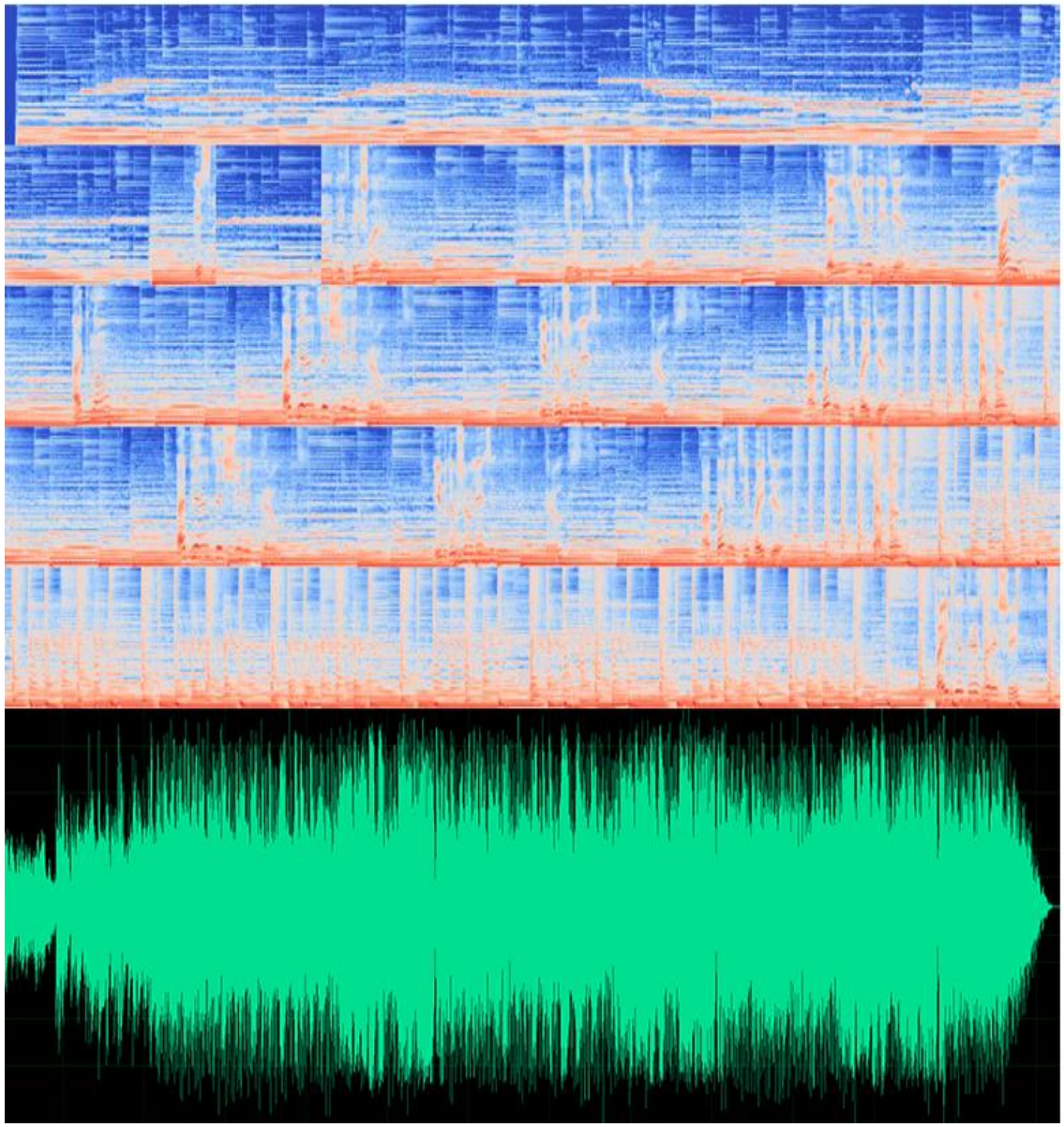
Тезаурус:

- *Что значит обучить нейросеть?* Научить нейросеть распознавать объекты максимально точно. В процессе обучения нейросеть обрабатывает датасет и находит признаки, по которым осуществляет предсказания.
- *Датасет* – набор данных (картинки, текст, музыка), на котором нейросеть обучается и тестируется. Например, чтобы нейросеть смогла распознать на картинке объект, ей нужно показать множество изображений, где он присутствует. Датасет нужно подбирать с умом, чтобы затронуть максимум возможных вариантов.
- *Как работает нейросеть?* Она создает множество весов случайных весов в начале обучения, которые позже корректируются на тренировочных данных. В результате нейросеть улавливает логические связи, которые обобщают объекты.

Схема работы популярных медиасервисов



Графическое представление трека



Название: Осень

Исполнитель: Лицей

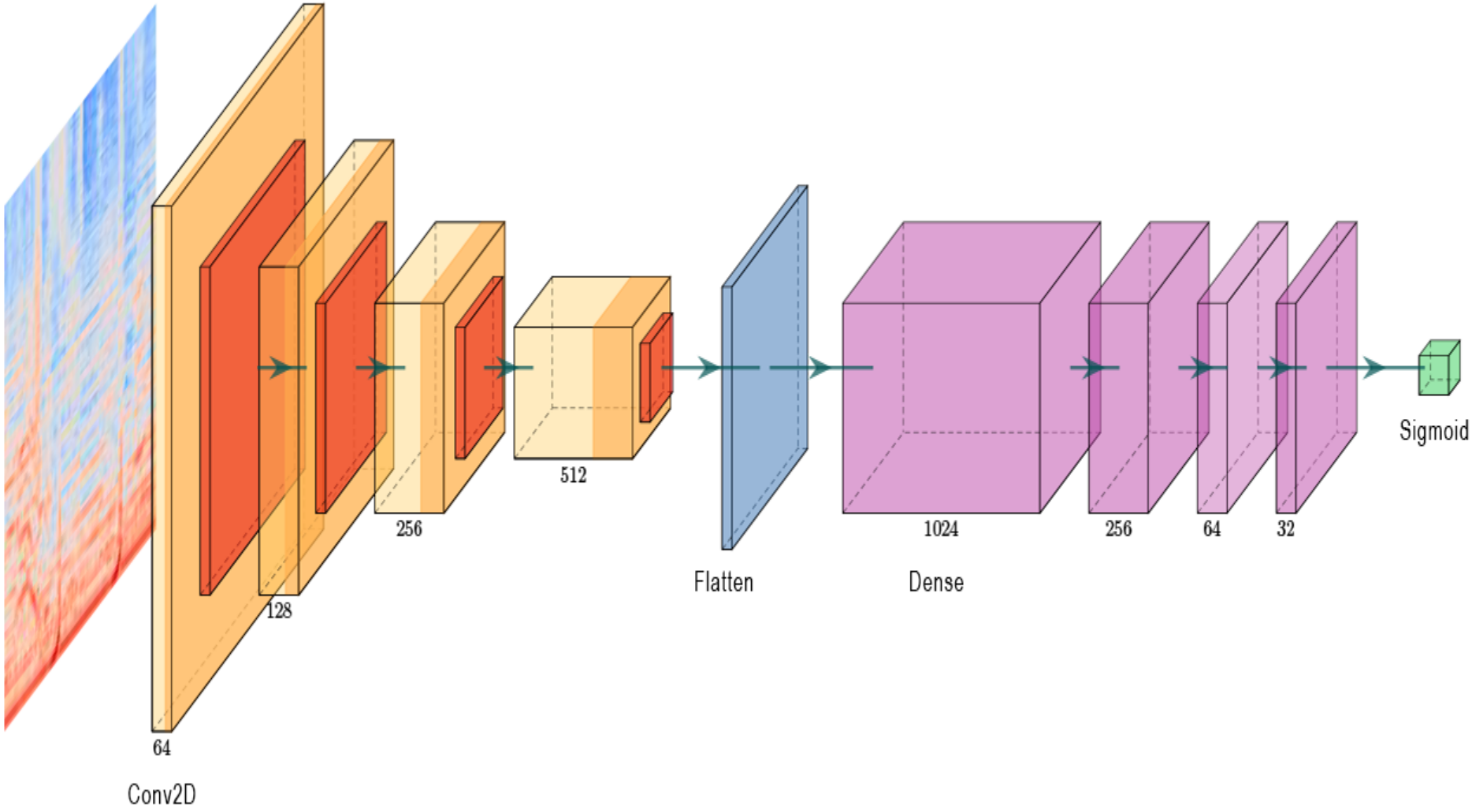
Верхний рисунок:

На рисунке изображена **mel** спектрограмма, идеальный формат чтобы записать аудиофайл в картинку. Обычно используется в сверточных нейронных сетях и компьютерном зрении. Получается путем преобразования амплитуды в децибелы.

Нижний рисунок:

Столбчатый график амплитуды в диапазоне от -32768 до 32768. Это стандартный формат кодирования аудиофайлов без сжатия.

WAV – без сжатия; MP3 – MPEG сжатие.


64
Conv2D
128
256
512
Flatten
1024
Dense
256
64
32
Sigmoid

Так выглядит архитектура модели в «нечитаемом виде»

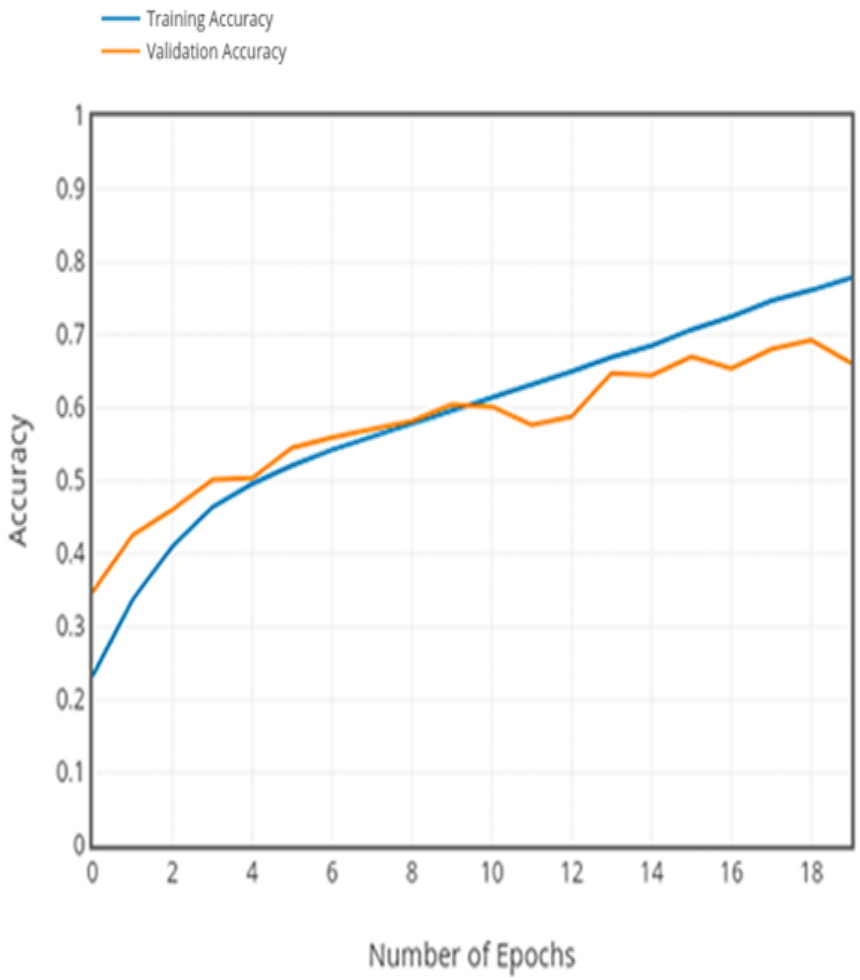
Терминал — -bash — 79×54

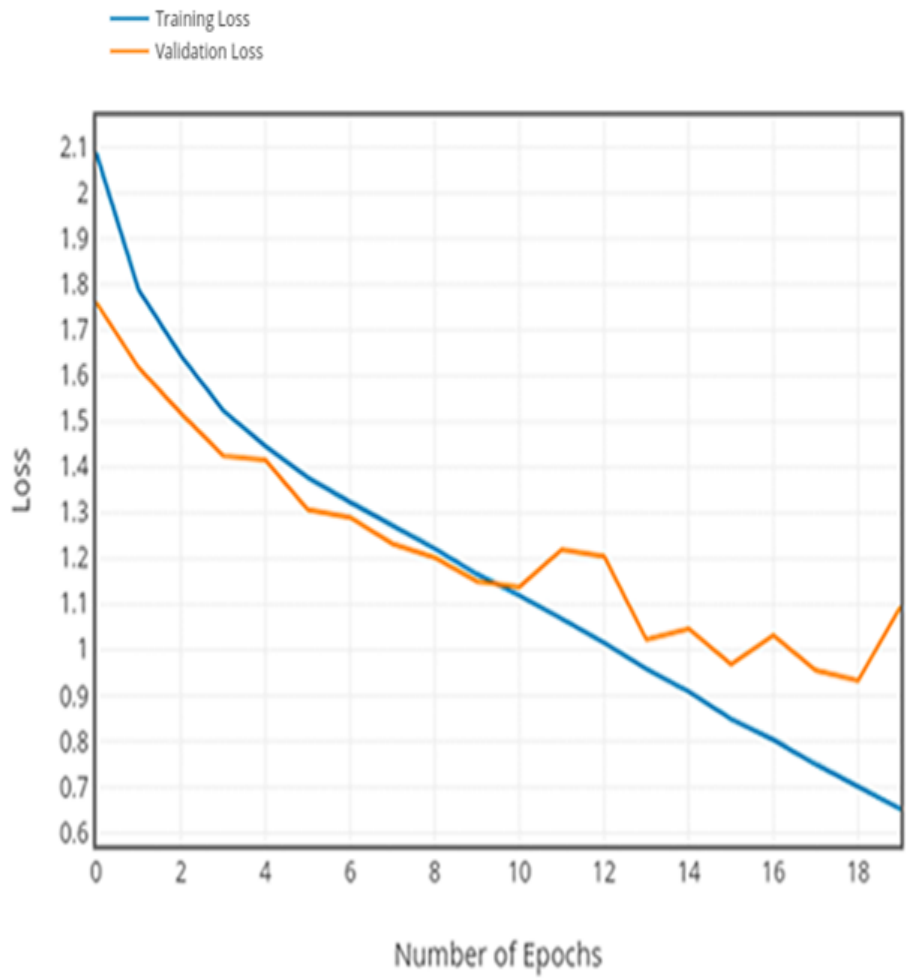
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 122, 122, 64)      3200
_________________________________________________________________
batch_normalization (BatchNo (None, 122, 122, 64)      256
_________________________________________________________________
average_pooling2d (AveragePo (None, 61, 61, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 128)       401536
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
average_pooling2d_1 (Average (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 6, 256)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 512)         1180160
_________________________________________________________________
batch_normalization_3 (Batch (None, 4, 4, 512)         2048
_________________________________________________________________
average_pooling2d_3 (Average (None, 2, 2, 512)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
batch_normalization_5 (Batch (None, 2048)              8192
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,273,281
Trainable params: 4,266,241
Non-trainable params: 7,040
_________________________________________________________________
```

train.py

```
23
24  model = Sequential()
25
26  model.add(Conv2D(filters=64, kernel_size=[7,7], kernel_initializer = initializers.he_normal(seed=1), activation="relu", input_shape=(128,128,1)))
27  model.add(BatchNormalization())
28  model.add(AveragePooling2D(pool_size=[2,2], strides=2))
29
30  model.add(Conv2D(filters=128, kernel_size=[7,7], strides=2, kernel_initializer = initializers.he_normal(seed=1), activation="relu"))
31  model.add(BatchNormalization())
32  model.add(AveragePooling2D(pool_size=[2,2], strides=2))
33
34  model.add(Conv2D(filters=256, kernel_size=[3,3], kernel_initializer = initializers.he_normal(seed=1), activation="relu"))
35  model.add(BatchNormalization())
36  model.add(AveragePooling2D(pool_size=[2,2], strides=2))
37
38  model.add(Conv2D(filters=512, kernel_size=[3,3], kernel_initializer = initializers.he_normal(seed=1), activation="relu"))
39  model.add(BatchNormalization())
40  model.add(AveragePooling2D(pool_size=[2,2], strides=2))
41
42  model.add(BatchNormalization())
43  model.add(Flatten())
44
45  model.add(BatchNormalization())
46  model.add(Dropout(0.6))
47  model.add(Dense(1024, activation="relu", kernel_initializer=initializers.he_normal(seed=1)))
48
49  model.add(Dropout(0.5))
50  model.add(Dense(256, activation="relu", kernel_initializer=initializers.he_normal(seed=1)))
51
52  model.add(Dropout(0.25))
53  model.add(Dense(64, activation="relu", kernel_initializer=initializers.he_normal(seed=1)))
54  model.add(Dense(32, activation="relu", kernel_initializer=initializers.he_normal(seed=1)))
55
56  model.add(Dense(1, activation="sigmoid"))
```

Line 1, Column 1 master


Training Accuracy
Validation Accuracy
Accuracy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16
18
Number of Epochs


Training Loss
Validation Loss
Loss
2.1
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0
2
4
6
8
10
12
14
16
18
Number of Epochs

Выводы:

- В результате работы была написана нейронная сеть, которая по загруженному музыкальному треку выстраивает рекомендательную систему выбора музыки.
- Создан web-интерфейс для упрощения взаимодействия с сетью.