Music Genre Classification using Deep Learning Techniques

Abdelrahman Mohamed Ibrahim: 19105771

Seif Eldin Khaled Abdelwahab 19105632

Omar Mohamed Al-Amir 19100463

Abdelrahman Ragheb Elsergany 202002421

Adham Mohamed Mostafa 19105744

Problem Statement

Music plays a very important role in people’s lives. Music bring like-minded people together and is the

glue that holds communities together. Communities can be recognized by the type of songs that they

compose, or even listen to. Different communities and groups listen to different kinds of music. One

main feature that separates one kind of music from another is the genre of the music. This project is: 1.

To build a machine learning model that classifies music into its respective genre. 2. To compare the

accuracies of this machine learning model and the pre-existing models and draw the necessary

conclusions.

Preparing the dataset

Firstly we used a music genres dataset that contains 10 different genres (blues- classical – country –

disco-hip-hop- jazz-metal-pop-reggae-rock ) each genre has 100 different songs with a duration of 30

secs and each song is divided into different segments and each segment has it’s number of samples so

each sample has an array containing mfcc features extracted for each sample, and then each segment is

labeled by a number to train our model, in this project we used 5 segments and 13 mfcc features for

each sample, and then at the end we saved it in the JSON file to facilitate it for others to work on their

models, we used the librosa library to extract audio features to train

First Model: multilayer perceptron

**Multilayer Perceptron**

**Intro:**

Before we dive deep into the Multiplayer Perceptron(MLP) lets first have a look at the artificial neuron and why we used the MLP . the artificial neuron is considered to be a computational unit that does certain calculation on some input . the artificial single neuron is mainly used for computation that deal with linear problems but it can’t solve nonlinear complex problems that’s why we will be using neural network containing multilayers of nodes. Here are two pictures which compare the **single artificial neuron vs artificial neural network** that contains multiple neurons and then we will talk about the structure of the **artificial neural network**

**The structure of the** **single artificial neuron**

Diagram

Description automatically generated

**The Structure of an artificial neural network**

**Bubble chart

Description automatically generated**

**1-Neurons:**

A neuron in a neural network work works similarly to neuron in human brain . it’s a mathematical function that collects and classifies information according to a specific architecture. They are also organized together to form layers which will be discussed later in the next topic

**2-Layers:**

There are mainly three types of layers in a neural network:

**The first is the input layers:**

The input layer which provides information signal into the network

**The second is the hidden layer :**

Are layers of neurons that are hidden because they are at the center of the topology of the neural network there also could be multiple hidden layers

**The third layer is output layer:**

Its used to get information out of the network after the info had been processed from the hidden layers

**3-Weighted connections:**

They are the links or edges which connect the neurons together and form the neural network

**4-activation function:**

They are fundamental function which are used to process incoming information into the network.

**How Computations In MLP are performed:**

Information is processed from the left to the right in the neural network in it involves multiple steps that first one being the **weights**

how are weights calculated weights and what are the types of weights.

Weights have two types W1 and W2. W1 involves the weighted connections between the input layer and the hidden layer. Each neuron in the input layer is connected with all the neurons in the hidden layer and they are computed as matrix of each input neurons with all of its weighted connection. Each row in the matrix represents the single input neuron and the columns are labeled accordingly with each neuron its connected to in the hidden layer. here is picture to illustrate the idea even further

Diagram, schematic

Description automatically generated

The Second one is the **net input**

Diagram

Description automatically generatedWhich is acquired from the multiplication of two matrices tighter the input layer matrix and the weighted connection matrix W1. N for net input = x for input layers \* W1 and the result is a row matrix [ele1… ele2.. elen…..]

The third and final thing in MLP computation is the **activation function:**

Chart

Description automatically generatedwhich takes net input value and using the sigmoid function calculates the value of the activation function

**Implementation models and architecture in code:**

* We used keras model for implementing the neural network
* We converted the data into 1D array using the Flatten function and we connected each neuron with its previous one from the previous layer using the Dense function which also acts as the activation function using Relu method
* In the final layer we used the SoftMax method in the Dense functions which outputs the result as a probability distribution
* We then used optimizer adam which is a Stochastic gradient descent algorithm

Second Model : CNN

Our second model is CNN Model

CNN is an advanced type of neural network that are mainly used for processing images. they perform better than Multilayer perceptron and have less parameters than dense layers. There are two component of CNN which are convolution and pooling. Firstly, we need to know what a kernel is, it’s a grid of weights that is applied to the image. Also, a matrix, which is slid across the image and multiplied with the input such that the output is enhanced in a certain desirable manner.

CNN architecture:

Diagram, engineering drawing

Description automatically generated

How does convolution/pooling apply to audio?

We can visualize the features as images and it can be interpreted as images.

Chart, histogram

Description automatically generated

With the time as X axis and frequency as the Y axis.

Implementing the CNN into code:

We will start by splitting our data into three parts train, validation and test. Why do we need validation and test? so we can validate with the validation part then see how the model will act on a never seen before data which is in the test.

After splitting the data we must not forget the CNN takes 3D array including the rgb so we are just going to use np.newaxis to add a new dimension into our data.

Building the model:

We will be using keras.sequential() to add 3 conv layer then flatten the output then put it to the dense layer then finally the output layer.

We will pass in the arguments that we want 32 filter and a 3x3 kernel, then we will pass the shape of the X\_train we will skip the index 0 as this is the number of samples but will take the other 3 dimensions which are needed for the CNN.

then we will use max\_pooling to down sample our input, we determined out grid size which is 3x3 and strides 2x2 then we will add the zero padding around all the edges.

Then we will use Batch normalization, which is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.

Will do the same for the 2nd layer but the 3rd layer we will just make the kernel a 2x2 in the conv and pooling.

Then we will flatten the output of the convolution to make it a 1D array, then feed it into a dense layer, then to avoid overfitting we should dropout.

Then in the output layer we make a dense layer and make the number of neural the same as the number of genres we want to predict.

Compiling the network:

We will be using optimizer.adam function with a learning rate of 0.0001

Training the CNN:

Evaluation of the CNN model:

To get the accuracy of the model

Make a prediction on a random sample:

As the .predict function expecting a 4D array and we are passing a 3D array from the model we will just add a dimension with the np.newaxis.

we want to extract the index with the max value.

Third Model is using RNN and LSTM

**What is RNN?**

An RNN model (Recurrent Neural Network Model) is a type of neural networks that is mostly used for Natural Language Processing, Speech Recognition and Synthesis.

In a general neural network, an input is processed through a number of layers and an output is produced, with an assumption that two successive inputs are independent of each other, but RNN adds another parameter: the memory.

**Why is RNN ideal for music labelling?**

We can think of audio as time series data. RNNs perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far. In theory, RNNs can make use of information in arbitrarily long sequences, but in practice, they are limited to looking back only a few steps.

Diagram

Description automatically generated

**What is LSTM?**

Long short term memory (LSTM) is an Artificial Recurrent neural architecture which is used in speech recognition, classifying and making predictions, and specially time series predictions.

**How does LSTM compare to RNN?**

* RNN

RNNs have feedback loops in the recurrent layer. This lets them maintain information in ‘memory’ over time. However, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies.

This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem).

* LSTM

LSTM networks are a type of RNN that uses special units in addition to standard units. LSTM units include a ‘memory cell’ that can maintain information in memory for long periods of time. This memory cell lets them learn longer-term dependencies, so it’s much more effective in case of music.

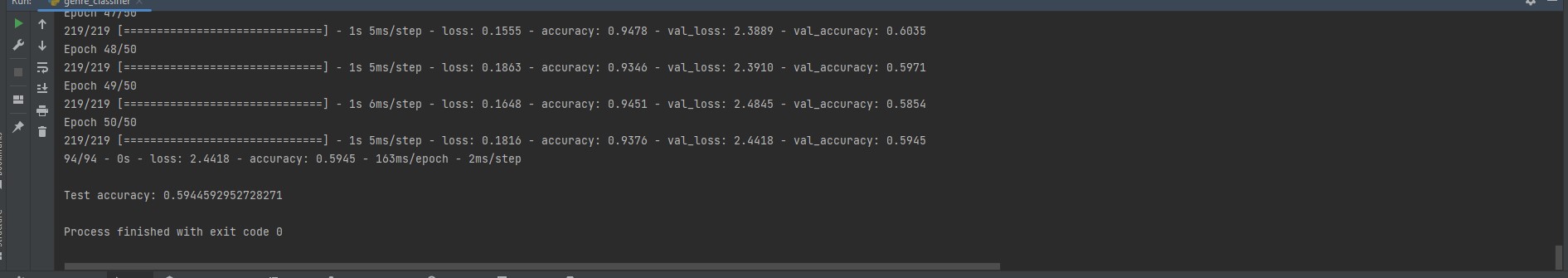
LSTMs deal with vanishing and exploding gradient problem by introducing new gates, such as input and forget gates, which allow for a better control over the gradient flow and enable better preservation of “long-range dependencies”.

**Model Details**

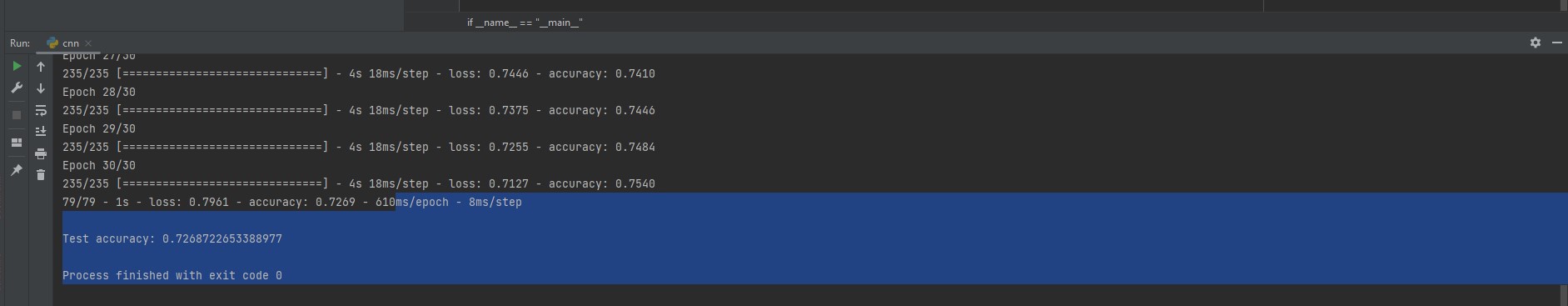
* We used the sequential keras model.
* 2 LSTM layers, 64 neurons each. The hidden state output for each input time step is passed to the second layer from the first.
* A dense 64 neuron layer. Relu (rectified linear unit) is used as the activation for the layer as to replace negative values with zeros.
* Following the dense layer is a dropout layer with a dropout probability of 30%. Dropout is a technique for reducing the odds of overfitting by simply skipping random neurons of the neural network in every epoch so that the network becomes less sensitive to specific weights.
* Output layer is a dense 10 neuron layer. softmax is used as the activation for the output layer to convert the vector of values to a probability distribution.

Results and comments

While using the three different Models we got results

On the mult-layer-perecepton we got

Accuracy of nearly 60%

While using the CNN model we got

Accuracy of nearly 72%

While using the LSTM MODEL we got 61.5% accuarcy

Text

Description automatically generated

So we can see that CNN model has the best accuracy for this problem statement