בה"ו

Using RNN To Predict Music Track Genre

Ori Darshan – 212458244

Amir Sabag – 316049311

Daniel Rosenberg – 319645735

Deep Learning and Natural Language Processing course

Git repository

Abstract

In this exercise, we tried to predict a music-track genre based on its characteristics (a classification problem).

In previous exercise we used simple methods like Softmax and Multy-Layer-Perceptron, This time we used more advanced techniques like LSTM to improve on older results.

We also tried to explain the low accuracy all methods suffered from by reducing the number of possible track genres.

Related Work

A great example for the use of LSTM networks can be found here

Many Jupyter Notebooks exploring the dataset we use can be found here

In particular, this notebook and this notebook managed to get very high accuracy predicting genres, in the code appendix of this exercise we show why their accuracy does not represent the quality of their models.

Experiment Description

Data

The Dataset is taken from Kaggle and consists of Spotify tracks over a range of 125 different genres. Each track has some audio features associated with it. The features are for example the duration of the track, its danceability – which describes how suitable a track is for dancing – etc. (See link for more information)

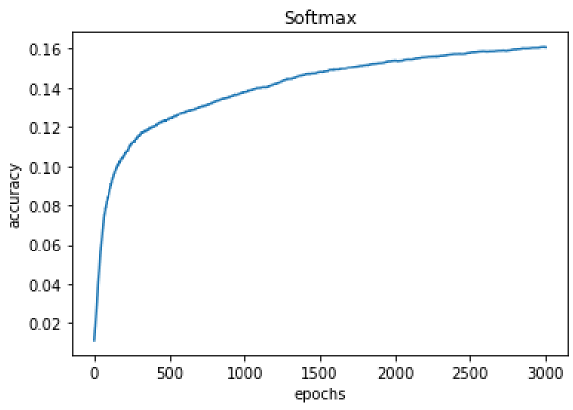
Previous Results

In the previous exercise we used 2 different algorithms to try and predict a track's genre based on its characteristics, Softmax and Multilayer-Perceptron (MLP).

This experiment gave as the following results:

Softmax:

The learning curve (accuracy/epochs) of the algorithm was :



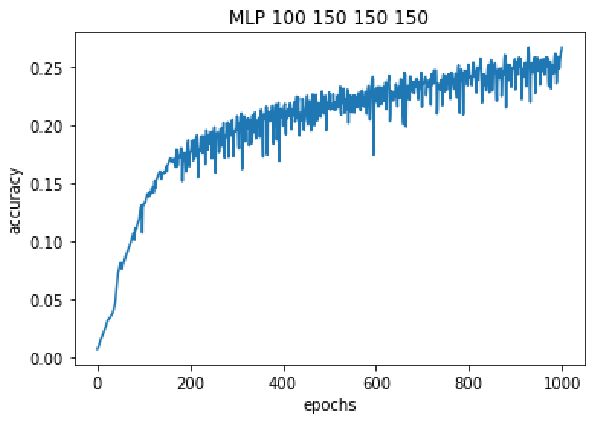
The accuracy of the algorithm on the test set was: 0.15667108

The accuracy is low but expected, as we explained in the exercise.

MLP:

We tried multiple architectures which yielded different results, the best architecture we tried was 4 hidden layers with 100 neurons on the first layer and 150 on the second, third and fourth.

The learning curve (accuracy/epochs) for that architecture was:



The accuracy on the test set was: 0. 25931308

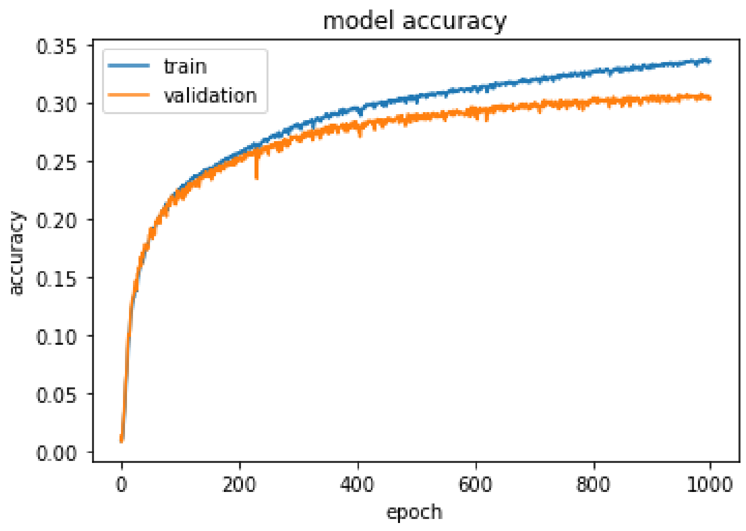
Experiment

In this experiment we used more advanced technics which we learned in the course, namely Recurrent Neural Networks (RNN) using LSTM (Long Short-Term Memory) units and Feed-Forward Neural Networks with Dropout layers.

First Model

The first architecture we tried was a neural network with 2 hidden layers, The first layer is a dense (fully connected) layer with 250 neurons, the second layer is an LSTM layer with 80 units.

The learning curve (accuracy/epochs) for that architecture is:



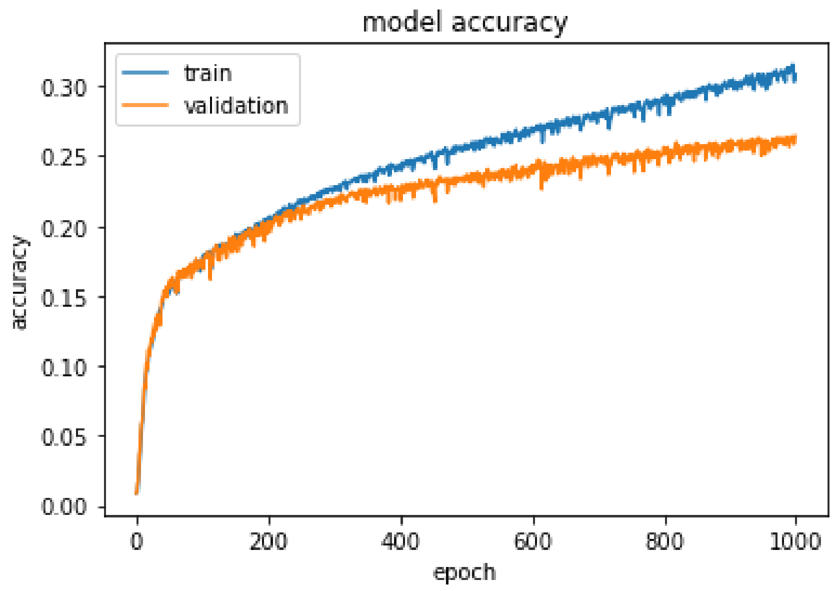
The accuracy on the test set was: 0.30048435

This accuracy is high which demonstrates the strength of the more complex model over the simpler Softmax model.

Second Model

Next, we tried to use an even more complex architecture, this time also with 2 hidden layers but now both layers are LSTM layers, the first with 80 units and the second with 150 units.

The learning curve (accuracy/epochs) for that architecture is:



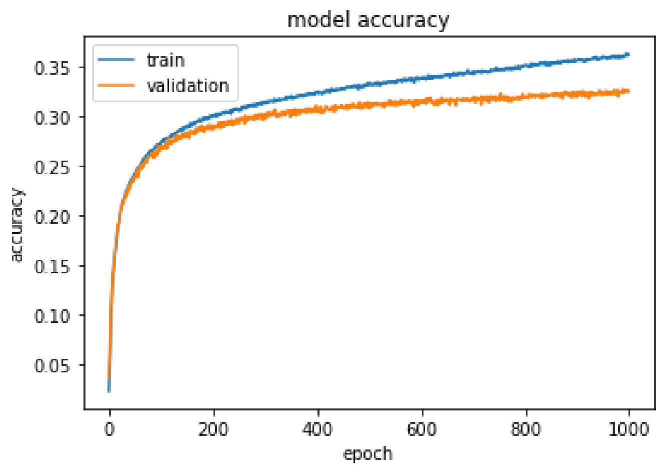
The accuracy on the test set was: 0.25834435

We can see the accuracy is low even though the architecture is more complex, this is because more complex models take more time to converge, and we ran all our models over 1000 epochs (which by itself took a long time)

Third Model

The third architecture we tried was a neural network like the MLP from the previous exercise but with added Dropout layer.

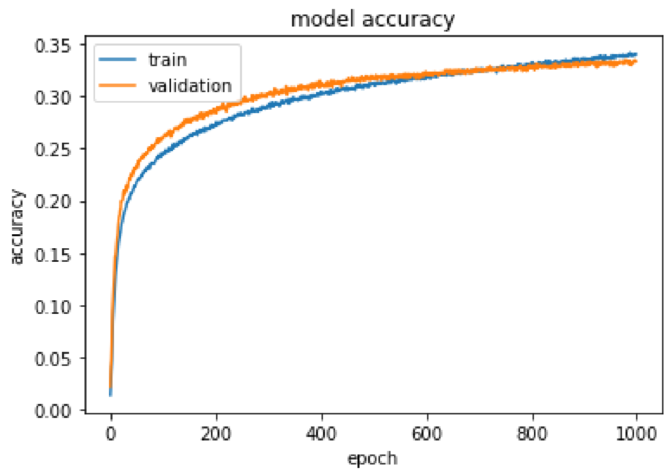
First, we replicated that architecture with TensorFlow 2, this gave as the following learning curve:



The accuracy on the test set was: 0.3172611

Better than our implementation!

Then, we turned the first hidden layer to a dropout layer, this gave as the following learning curve:



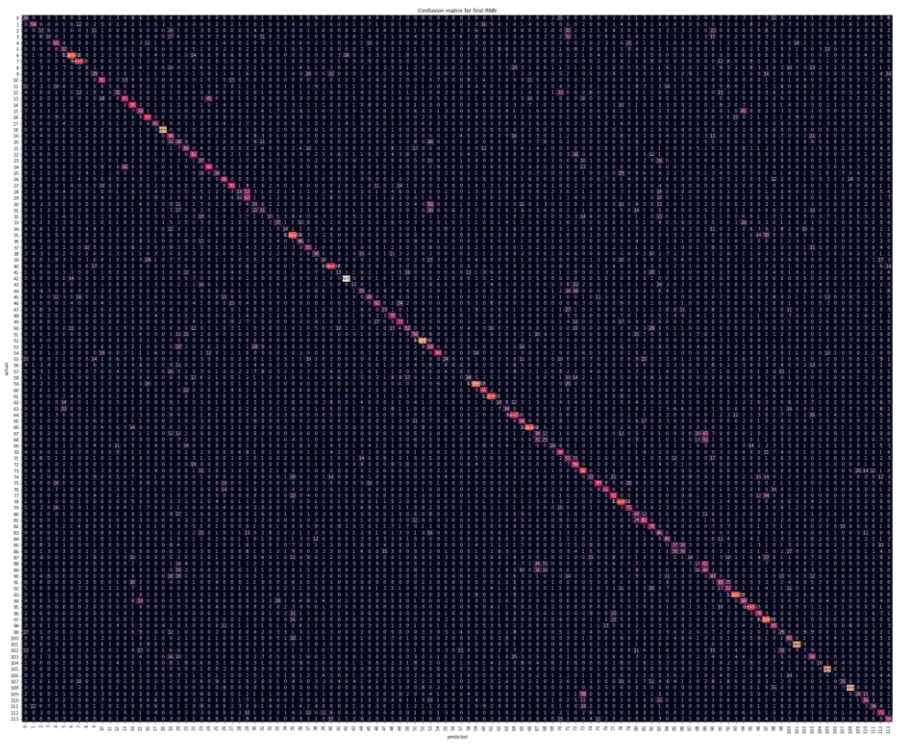
The accuracy on the test set was: 0.32245707

We can see that the dropout technique did not help our model get better results, this is probably due to the objective of the dropout technique- to reduce potential overfitting of the model to the data.

Our model didn't manage to interpolate the train set therefore a dropout layer isn't needed as regulation.

Genre Reduction

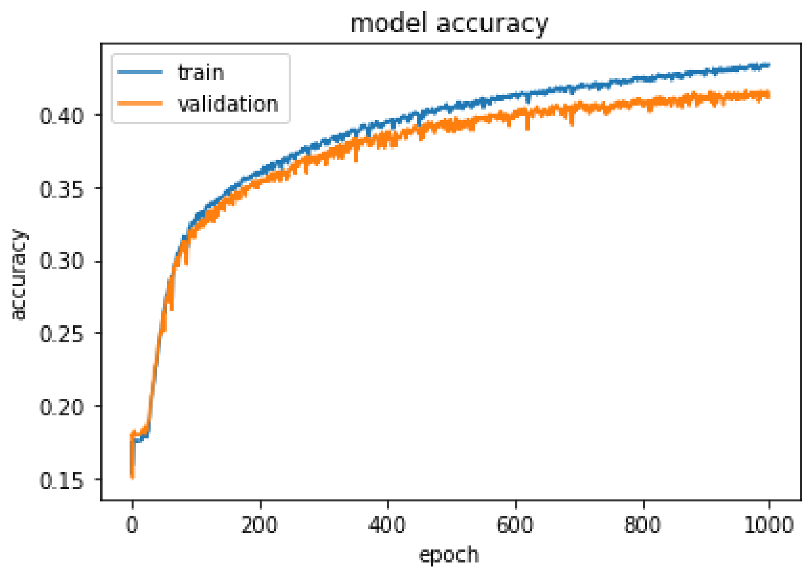
From the first model we trained we received the following confusion matrix:



The errors on the matrix seem to be parallel, this leads us to believe that the main reason for the low accuracy are some interchangeable genres.

Looking at the labels of our dataset we notice many overlapping genres like pop, indie-pop, j-pop, k-pop, mandopop etc.

We tried combining multiple labels to one in order to improve accuracy, training the first architecture (RNN) on the reduced dataset with only 66 labels instead of 114 gave us the following learning curve:



The accuracy on the (label-reduced) test set was: 0.4078820

The accuracy is much higher, as expected, which leads to the conclusion that similar genres are a major factor in the low accuracy of the models.

Conclusions:

Predicting music tracks genre based on their characteristics turned out to be a difficult task.

Complex models like LSTM did bring better results, however too complex models were hard to train. Methods that help against overfitting didn’t help because we didn’t manage to interpolate the data.

We showed that the high number of possible genres is a primary reason for the low accuracy of our models and demonstrated how reduction of their number help achieving better results.