Deep Learning Methods Explored for Music Genres Classification

Haopeng Wu
University College Dublin
haopeng.wu.ucdconnect.ie

Abstract

Music plays an important part in people's daily life. The classification of music genres is important for music recommendation. **Deep learning** methods have shown promising results in the automatic **music genres classification**. In this document, various deep learning techniques for music genre classification are explored and compared, including Multilayer Perceptrons (**MLPs**), Convolutional Neural Networks (**CNNs**), Recurrent Neural Networks (**RNNs**), and **hybrid architectures**. The impact of different input representations are also investigated, such as **Mel Spectrograms** and **MFCCs**, on the performance of these models. **Transfered learning** is employed to use pretrained models.

A lot of interesting findings are found along the way of the exploring journey. The most important finding is how easy it is for one to build a decent deep learning model to address a problem in a particular field with the help of the abundant resources online.

1 Introduction

Music is a universal language that has been enjoyed by people across the globe for centuries. Over time, music has evolved and diversified, resulting in the emergence of various music genres that cater to different tastes and preferences. While humans have been able to classify music into various genres based on their unique characteristics, the advent of deep learning technology has made it possible for machines to do the same.

Deep learning is a branch of machine learning that uses neural networks to learn and make decisions based on input data. With the help of deep learning algorithms, it is possible to classify music genres automatically by different music features.

Music genre classification by deep learning has various applications, including music recommendation systems, audio-based search engines, and music analysis tools. In recent years, researchers have explored various deep learning techniques such as MLPs, CNNs, and RNNs to classify music genres with high accuracy by different music data.

This document explores various deep learning techniques to address music genres classification and the differences are compared and analysed. Specifically, 5 deep learning models are implemented based on 3 different audio features. Each model is comprehensively hypertuned against architecture, activation function, units in layers, dropout rate, learning rates, batch size, and epochs. The 5 models are MLP, VGG16, YAMNet, LSTM, and a hybrid of YAMNet and LSTM. The 3 features are audio waves, MFCCs, and Mel Spectrograms.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

There are some of the interesting findings along the way. 1. Simplicity beats complexity. The simplest model is a 3-layer MLP. It took the least time to train yet it has the best result of an accuracy of 89% whereas LSTM took the longest time but has the worst result of 60%. 2. Hyperparameters are very important. Some of them leads to a terrible result not better than random guessing. 3. Preprocessing is so important. When doing MLP, non-scaled MFCCs led to a bad result below 15%. 4. Transfer learning is so powerful that it saves a lot of time to train complex models. Pretrained models can be used to extract features that can be used for the problem at hand. 5. VGG16 is for image classification yet it can be used for the image representation of audio. In other words, the features extracted by an image model is relevant to audio classification. However, it performs less than those audio dedicated models, e.g. VGGish and YAMNet. After all, VGGis and YAMNet are trained against audios and VGG16 is trained against images. 6. When got a bad result not better than random guessing, it could be a data problem or a hyperparameter problem. 7. A CNN model's weight file is much bigger than other, in this case, VGG16. It's because of the great number of weights, which is due to the fact that each pixel is regarded as a feature in CNN. 8. When the accuracy is 100% and the val-accuracy is 70%, it means overfitting, so more anti-overfitting should be applied like dropout layer. Also, the complexity of the model should be reduced, maybe by reducing the number of layers or units of a layer. 9. If the train-accuracy is lower than val-accuracy, it feels like the model is not complex enough. Maybe add more layers or units. 10. If all the units number goes to the maximum in a hypertuning process, it might indicate an underfitting. 11. For VGG16, I first followed some tutorial online, but got poor results. The reason is I cropped the images to 150 by 150 from the top left corner, during which more than half of the information was lost since the original size is 432, 288. After applying the full image, the result got much better. 12. tanh should be used with LSTM so that the cuDNN kernels can be used to speed up the training process significantly. 13. Memory is so important to deep learning training. Colab crashed a lot due to limit of memory. VGG16 needs the most memory since it has a lot of learnable weights. 14. When I got an accuracy of 99%, it looked so suspicious that I examined the codes several times and I found out that the reason was I had not initialed a new model instance for it. Simply put, I used the fitted model that had seen the training and testing data so even the validation accuracy is as high as a 99%.

The structure of the document is as follows. In section 2, related work around music genres classification is introduced and compared with my approaches. In section 3, a detailed description of the project is given with respect to the dataset used, preprocessing, baseline approach, hyper-parameter tuning, and etc. In section 4, the results of these experiments are reported and analysed. In section 5, an overview of the main findings is shown. Also, proposals regarding how the models can be improved in the future are made. References lied in the section 6.

2 Related work

In 2002, Tzanetakis and Cook presented music genre classification as a pattern recognition task[1]. In that work, the authors assessed the classification using acoustic features extracted from the sound on a dataset with 1000 music pieces labeled according to 10 musical genres. Instead, the MLP in this document uses MFCCs to train a MLP. A baseline is also created in a similar way for this document.

Gwardys and Grzywczak [2] used a CNN trained on the Large Scale Visual Challenge (ILSVRC) as a feature extractor. In their case, they did not have enough data to train a classifier, but enough data was available in another domain of interest where the data could be in a different feature space or follow a different data distribution. Similar to my work, I used VGG16, which is a visual model, to extract features for processing audio. Pan and Yang [3] showed that this knowledge transfer, also known as transfer learning, if done successfully, would greatly improve the performance of the learning algorithm hence avoid expensive data-labeling efforts.

Sigtia and Dixon [4] trained a neural network with rectified linear units (ReLUs) using Stochastic Gradient Descent (SGD). They used the activations of the hidden layers of the neural networks as features and trained a Random Forest classifier on top of these features to predict the classes. Similar work in this document that use batch Stochastic Gradient Descent and applied RelU on hidden layers

and trained MLP, CNN, and RNN on the top of the pretrained VGGish and YAMNet. Thus, transfered learning is also applied.

3 Experimental setup

The GTZAN dataset[5] is used for all models. It has 1000 30-second music pieces classified into 10 classes. Each class has 100 audio pieces. The distributed dataset consists of three parts. The audio file in the wav. format. Mel Spectrograms in png. format. And the extracted MFCCs features in CSV, which have been summarized by taking the mean and std. All three forms of data are used. They are used as follows.

models	MLP	VGG16	LSTM	YAMNet	YAMNet&LSTM
data features	MFCCs	Mel Spectrogram	MFCCs	audio wave	audio wave

Figure 1: The implemented models and the data feature used

Each model is comprehensively hypertuned using KerasTuner[6] in terms of model architecture(whether a particular layer is included, e.g. Dense, LSTM, Dropout), layer units, learning rate, activation function, dropout rate, learning rate, batch size, and epochs. The best epochs is chosen separately using early stop callbacks, i.e. stopping the training before the result is not improving. Different preprocessing methods are compared for MLP and VGG16. Hyperparameters are chosen based on a validation set, which is split from the training set. The weights from the best hyperparameter set is saved.

The preprocessing on the label data is one hot encoding since we are doing multiple classification problem here. The proprocessing on the training data varys in accordance with concrete models.

tanh activation function is used for LSTM and the hybrid of LSTM&YAMNet for optimised training.

3.1 MLPs

The first model MLP, which contains 3 layers and takes summarized MFCCs(means and stds) as input features. The summarized MFCCs is included in the distributed dataset in two versions. One is for 30-second audio. The other is for 3-second audio. 3-second version is chosen since it led to better results. The MFCCs in the csv is not normalized. It was normalized before used.

MLP is chosen to be the baseline model because it is simple and effective.

3.2 VGG16

The second model of a CNN model built and pretrained by Google, VGG16. Transfer learning is applied here because the CNNs are usually hard to train both in terms of computing power and dataset scale.

The data used is Mel Spectrogram, which is included in the distributed dataset in the png format. The images are converted into RGB, then three preprocessing methods are tried on them. 1) no preprocessing; 2) normalized by 255; 3) use a function preprocess_input, which is in the package of tensorflow.keras.applications.vgg16.

On top of the pretrained VGG16, three or four(depend on a hyper parameter) Dense layers are applied. The first two layers have the sizes up to 3072(try to match the size of the VGG16 features), and then a layer with size of a range from 32 to 1024.

3.3 YAMNet

Three Dense layers are seated on top of the pretrained YAMNet model.

The input is the audio wave files. 30-second audio pieces are loaded and then each is split into 10 3-second smaller pieces to produce more data. At this stage, some pieces have different sizes with the majority. Therefore, a trimming process is applied here to include only the first 62 frames. Then, the data is summarized by take its mean and std across the time axis, after which the data is fed into the Dense layers.

3.4 LSTM

The input data is MFCCs. But the MFCCs csv. file in the given dataset can not be used because they are already summarized across time. So, the audio wave files are loaded, and split. After clearing some malformed samples, MFCCs features are generated from the 3-second audio pieces. At last, three LSTM layers are applied to these data.

3.5 Hybrid of LSTM&YAMNet

On top of YAMNet features, instead of putting MLPs, LSTMs are put on it. Specifically, Audio waves are used to generate YAMNet features. Then, three LSTM layers are applied on the extracted YAMNet features. Data preprocessing is the same with YAMNet.

4 Results

The evaluation phase involves spliting the data twice. The first time splits the data into training and testing datasets(The ratio is 4 : 1). The testing dataset will not be used until the very end to test the final generalizing ability of the model. The training data is then further split into two parts. One is for training and the other is for validating(The ratio is again 4 : 1). Validating is for finding the best set of hyper-parameters.

Early stop was used to determine the best epochs. With this technique, the training process stops itself when it sees no further improvement for the validating accuracy in a number of iterations. We need to set a patience value for this. The greater the value, the more epoches will be run.

The measure used here is accuracy, which describes the rate of correctly classified samples. In this case, it means how accurate the model is when classifying the samples. A value of 80% means it got 80% of the sample correctly classified and the rest 20% incorrectly classified. An accuracy of 10% in our case means the model does not perform any better than random guessing.

For the models that are learned on 3-second audios features, the model can classify a genre by listening only to any 3-second piece of the audio. However, if that 3-second piece happen to be blank or silent, the classification would not be accurate. For the models that are learned on 30-second audios features, the model can classify a genre by listening to a 30-second piece of music.

executions_per_trial is set to 2 so that two executions are run for each trial instead of 1, so that the result is more stable and representative.

4.1 MLP

The MLP model reached the best result of 88.9% accuracy. The best set hyperparameters is as follows.

This model has the least amount of trainable weights and took the least amount of time to search the best hyperparameters and to train the model. Yet, it has the best result of a 88.9% accuracy. This model is regarded as the baseline model because of its simplicity.

I only did 5 epochs when doing hypertuning. As we can see above, both are increasing but the validation accuracy is flatter from 1 to 3 epochs. Two lines in each graph are the two executions for each trial. The two executions performed similarly in training but more differently for validating.

```
'activation': 'relu', 'units_1': 448, 'dropout_1': False,
'units_2': 672, 'dropout_2': False, 'units_3': 32,
'dropout_3': False, 'learning_rate': 0.001, 'batch_size': 12,
'epochs': 11
```

Figure 2: The best set hyperparameters for the MLP.

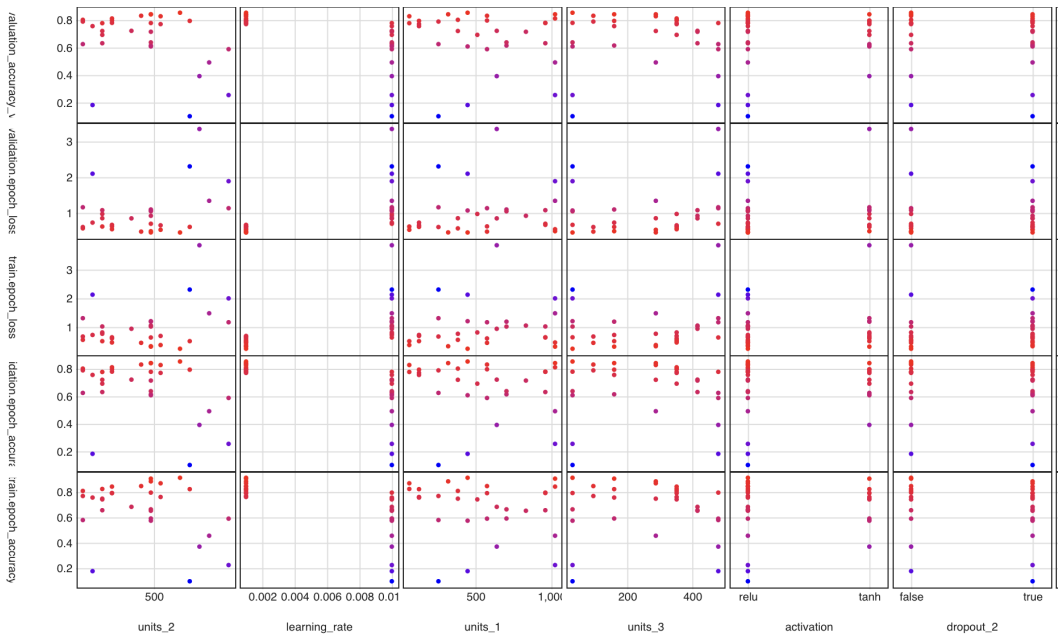

Figure 3: Part of the hyper parameter search space and result for the MLP.

Figure 4: Train and validation epoch accuracy

4.2 VGG16

The pretrained VGG16 is utilised to give a result of 65.5% accuracy on the following set of hyperparameters.

The result is much less than the baseline model. It took a considerable amount of memory to train due to the number of learnable weights. After VGG16 is an image model. The features it extracts is

```
'activation': 'relu', 'units_1': 3072, 'dropout_1': True,
'units_2': 1536, 'dropout_2': False, 'units_3': 288,
'dropout_3': False, 'has_forth_layer': False, 'dropout_4': False,
'learning_rate': 0.001, 'units_4': 736, 'batch_size': 28,
'epochs': 6
```

Figure 5: The best set hyperparameters for the VGG16.

for classify images not sounds. 65% is much better than random guessing, 10%. Therefore, VGG16 features are helpful with audio classification in some way.

4.3 YAMNet

YAMNet is a pretrained audio model, which is good at extracting audio features. This model reached a result of 84.9% of accuracy on the following hyperparameters.

```
'activation': 'relu', 'units_1': 1024, 'units_2': 512,
'units_3': 128, 'dropout': True, 'learning_rate': 0.001,
'batch_size': 52, 'epochs' : 5
```

Figure 6: The best set hyperparameters for the YAMNet.

This model is a bit less than the baseline model but close and much better than VGG16. After all, this extracts audio features instead of image features in the case of VGG16.

4.4 LSTM

LSTM is a RNN, which is good at working with time series. This model reached a result of 77.3% of accuracy on the following hyperparameters.

```
'units_1': 416, 'units_2': 32, 'dropout_2': True,
'units_3': 48, 'dropout_rate2': 0.2, 'learning_rate': 0.001,
'batch_size': 12, 'epochs': 16
```

Figure 7: The best set hyperparameters for the LSTM.

This model is a less than the YAMNet model but still has a good accuracy of 77%, which is much more than VGG16. This model is really slow to train and very computing demanding.

4.5 Hybrid of YAMNet&LSTM

The hybrid model has a result of 84.2%, which is significantly better than LSTM alone. The hybrid did show some advantage over its component. However, it achieved a similar result to the other component, YAMNet.

5 Conclusion & Future work

By exploring MLP, CNN, RNN, and transfer learning technique, music genres classification problem is solved by achieving high accuracies on the five models. MLP with MFCCs is the simplest yet

```
'units_1': 480, 'dropout_rate1': 0.15, 'units_2': 128,
'has_additional_layer': False, 'dropout_2': False,
'units_3': 80, 'dropout_rate2': 0.25, 'learning_rate': 0.001,
'units_additional': 64, 'batch_size': 44
```

Figure 8: The best set hyperparameters for the LSTM.

the best one. VGG16 is not for audio features but did okay. YAMNet is a fine model for audio features and gave 85% of accuracy. LSTM worked well but took too much time to train. A hybrid of LSTM&YAMNet worked better than pure LSTM but not better than pure YAMNet.

5.1 Findings

There are two additional findings that I would like to add here.

The first one is that there is really not that one architecture fit for all. The choice of architecture is very open. It is not that more complex model is better. It is not that fancier models can beat the simple ones. It is not true in the machine learning world.

The last one is how easy and cost-effective it is for an individual to build a decent deep learning model to address a real world problem nowadays. That might be the output of the development of science.

5.2 Future work

The champion among this models is the 3-layer MLP, which is the baseline model, yet, it discarded most of the sequential information. LSTM that respects the sequential information should have better performance yet the YAMNet and LSTM hybrid devised has achieved 85%, which is lower than 89% achieved by MLP. One focus of future work might be experimenting and tuning the performance of RNN models and its hybrid. Hopefully, better results can be achieved.

Try more anti-overfitting techniques, such as regularization, data augmentation, ensemble methods, and cross-validation.

At last, there might be a defect with CNNs processing Mel Spectrograph because unlike photos, vertical translation should not be implied on the Mel Spectrograph. The reason is different vertical lines representing different pitch of sounds when the same pattern found across verticle lines. The same shape in the higher pitch could represent totally unrelated things to that in the lower pitch. Maybe, 1D CNN could help here because there is only one axis of translation.

References

[1][5] G. Tzanetakis, P. Cook Musical genre classification of audio signals IEEE Trans. Speech Audio Process., 10 (5) (2002), pp. 293-302

[2] G. Gwardys, D. Grzywczak Deep image features in music information retrieval Int. J. Electron. Telecommun., 60 (4) (2014), pp. 321-326

[3] S.J. Pan, Q. Yang A survey on transfer learning IEEE Trans. Knowl. Data Eng., 22 (10) (2010), pp. 1345-1359

[4] S. Sigtia, S. Dixon Improved music feature learning with deep neural networks IEEE International Conference on Acoustic, Speech and Signal Processing (2014), pp. 6959-6963

[5] Karen Simonyan and Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition karen,az@robots.ox.ac.uk

[5] Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR) (2015). Retrieved from https://arxiv.org/abs/1409.1556

[6]https://keras.io/keras_tuner/