Progressive Rock and Non-progressive Rock Classification

Dazhi Wang
dazhi.wang@ufl.edu
UFID: 1558-1148

Yujie Tang
yunjie.tang@ufl.edu
UFID: 1585-8535

Huaiyue Peng
huaiyue.peng@ufl.edu
UFID: 8521-6080

April 2021

1 Introduction

Progressive rock is a broad genre of rock music that first emerged from the mid to late 1960s and saw a high level of popularity in the 1970s [1]. In this project, we implemented some classifiers to distinguish progressive rock from non-progressive rock. We used 156 non-progressive rock songs and 96 progressive rock songs as our training set, and 111 non-progressive songs plus 101 progressive songs for testing. The music itself will be sole input for our classifier and no other meta-information (title, artist etc.) will be used. By extracting features from the data set, we trained our models and used these models to predict which category a song belongs to. In addition, the classification results of 14 "Other" songs will be indicated in each part.

2 Data processing

2.1 Dataset

Our training set contains 157 non-progressive (nonprog) rock songs and 98 progressive (prog) rock songs. The songs are provided as audio files of different formats and sample rates.

2.2 Preprocessing

We use librosa to process the audio files. By default, librosa will convert input audio files to a sample rate of 22050Hz when the input audio's sample rate is not, which results in long data processing time. To facilitate the data processing, all input audio files are converted to the same sample rate 22050Hz using ffmepg before any further processing. The command below is used.

```
ffmepg -i input.mp3 -ar 22050 output.mp3
```

Given the small size of our training set, data augmentation is needed. For each song, the first and last 10 seconds are discarded and divide the remaining parts into 4 second segments. Such procedure generates about 22692 song segments. Of all the segments, about 64% are prog samples, so we downsample the prog samples to make the number of prog songs and nonprog songs equal in the training set.

2.3 Feature generation

Features are calculated for each of the 4-second segments. The features are mostly readily calculated using the librosa library. We list all the features in our baseline classfier below.

- 40 MFCC coefficients are calculated using the librosa default settings. Each of the coefficient is averaged over the 4 seconds. Figure 1 is an example of MFCC features extracted from an example song.
- Root mean square (RMS).
- Spectral centroid.
- Spectral bandwidth.
- Spectral flatness.
- Spectral roll-off.
- Zero crossing rate.

There are certainly other features that can be of use, but these features should be good enough for a baseline classifier. All the features are normalized. Figure 2 shows the distribution for all 46 features for prog songs and nonprog songs. The distributions are all quite similar which suggests the it is a hard problem to classify prog songs and nonprog songs. Also it suggests the difference between prog songs and nonprog songs might be coded in the correlation between these features, or a sequence model which does not involve averaging over time might do a better job.

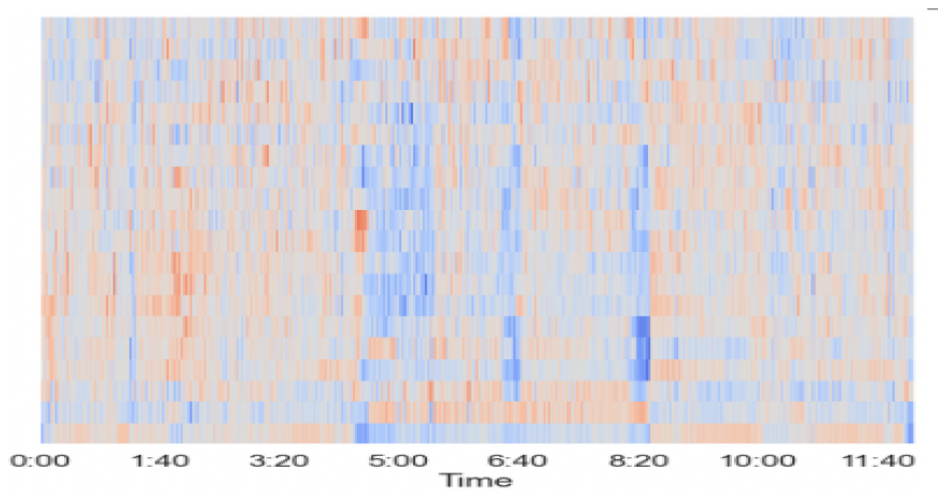

Figure 1: MFCC features of progressive rock song named 01 - Luminol.mp3. In mel-frequency cepstral coefficients (MFCCs), the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response [2].

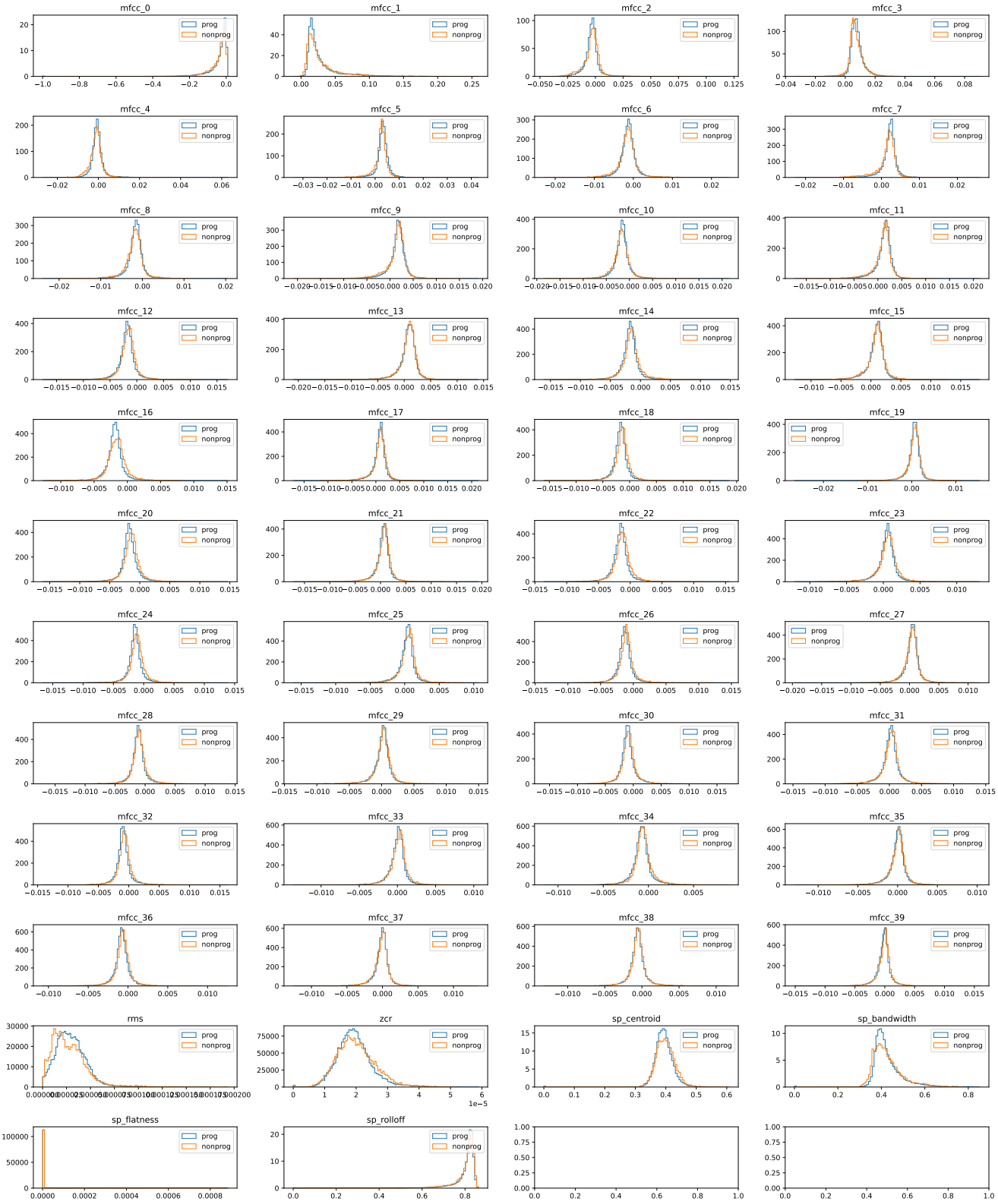

Figure 2: Distribution of the 46 features for prog and nonprog songs

3 Baseline DNN classifier

3.1 DNN model

We built 3 neural network models consisting of dense layers only. The activation function is ReLu. For all 3 models, we used the Adam optimizer with learning rate 1e-3 and binary cross entropy loss function. The batch size is 32 and 1000 epochs are trained. To monitor overfitting, 20% of the training set is used as the validation set.

Model name	Model configuration
dnn.h128	1 hidden layer with 128 units
dnn.h128h32	2 hidden layer with 128 and 32 units respectively
dnn.h128h64	2 hidden layer with 128 and 64 units respectively

Table 1: Model configuration for baseline dnn models

3.2 Features

For the baseline DNN classifier, a total of 46 features are calculated.

3.3 Performance on training set and validation set

3.3.1 The 1 hidden layer dnn.h128 model

This is the simplest dnn baseline model with only one hidden layer. As shown in Figure 3, there is definitely some degree of overfitting. At about 800 epochs, the validation loss stops decreasing. The difference between training accuracy and validation accuracy is about 5%. Fig 4 shows the confusion matrix on the training set and the validation set. The accuracy for nonprog songs is 82% while the accuracy for prog songs is 92% for training set. For the validation set, the accuracy for nonprog songs is 78% and that for prog songs is 88%.

3.3.2 The 2 hidden layer models

Figure 5 and 6 shows the loss and accuracy over the training process for the two models. There is more overfitting compared with the one hidden layer model. For both models, the validation accuracy levels off at around 85%. The confusion matrices shown in Figure 7 and 8 are calculated at the point where the validation loss starts to rise.

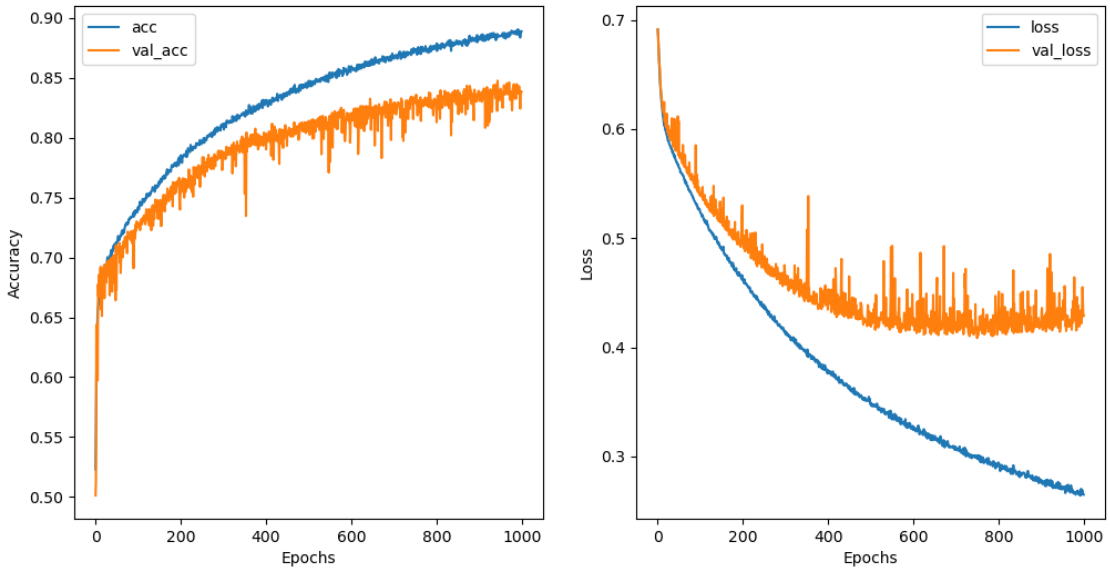

Figure 3: Accuracy and loss over the training for the dnn.h128 model

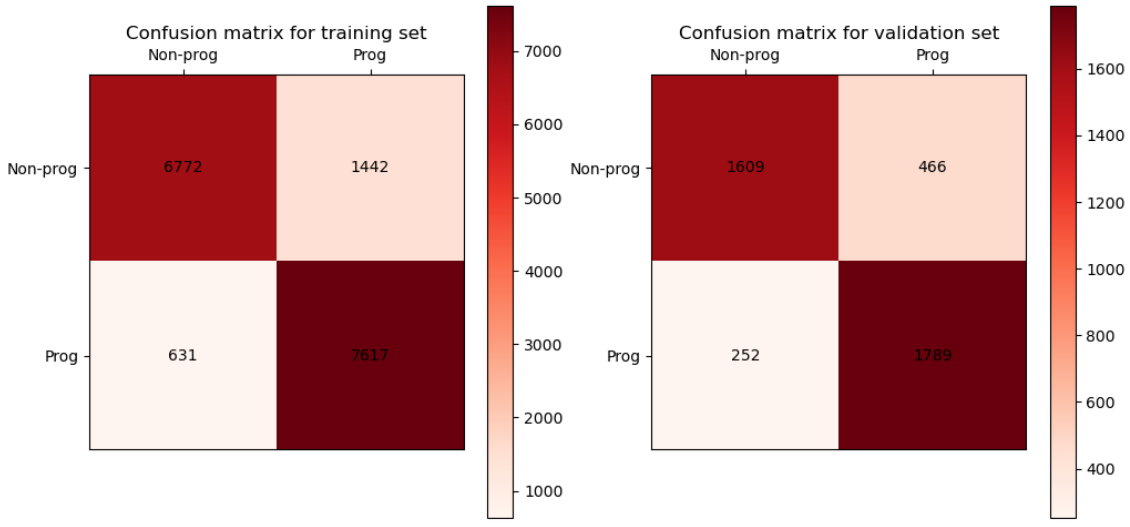

Figure 4: Confusion matrix for the dnn.h128 model

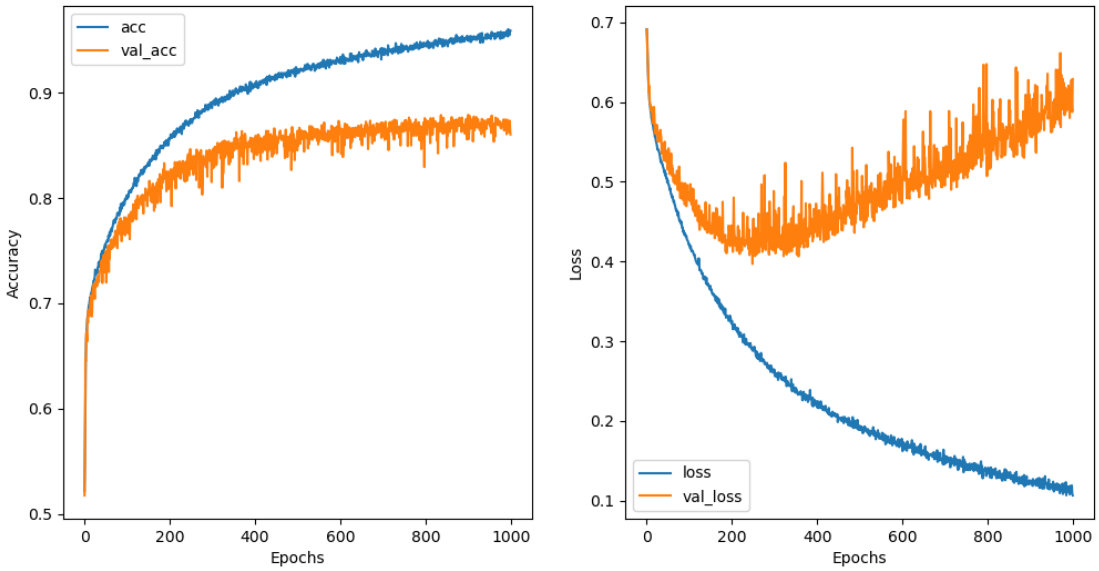

Figure 5: Accuracy and loss over the training for the dnn.h128h32 model

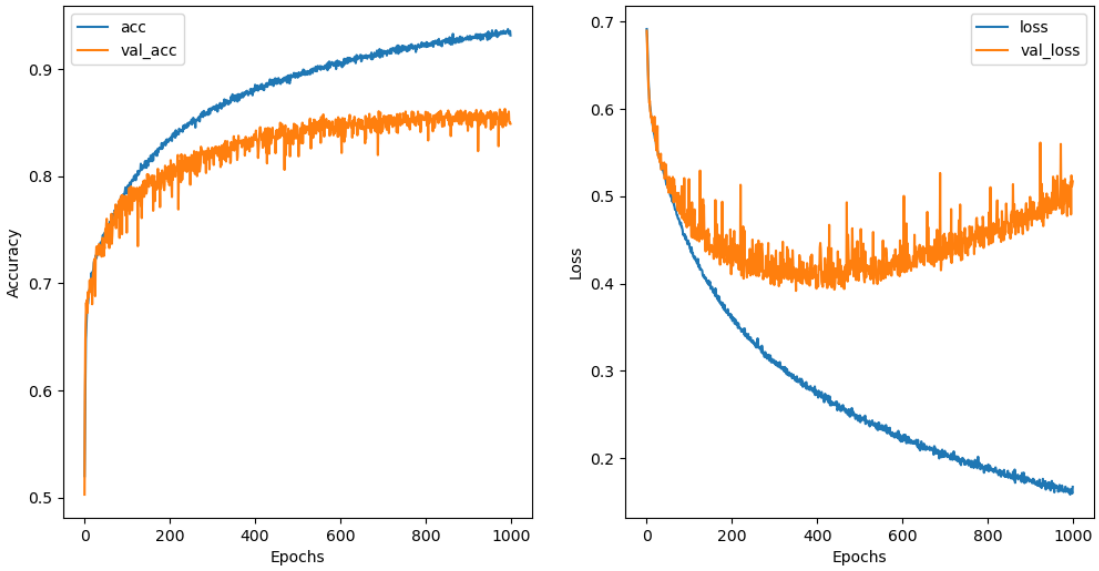

Figure 6: Accuracy and loss over the training for the dnn.h64h32 model

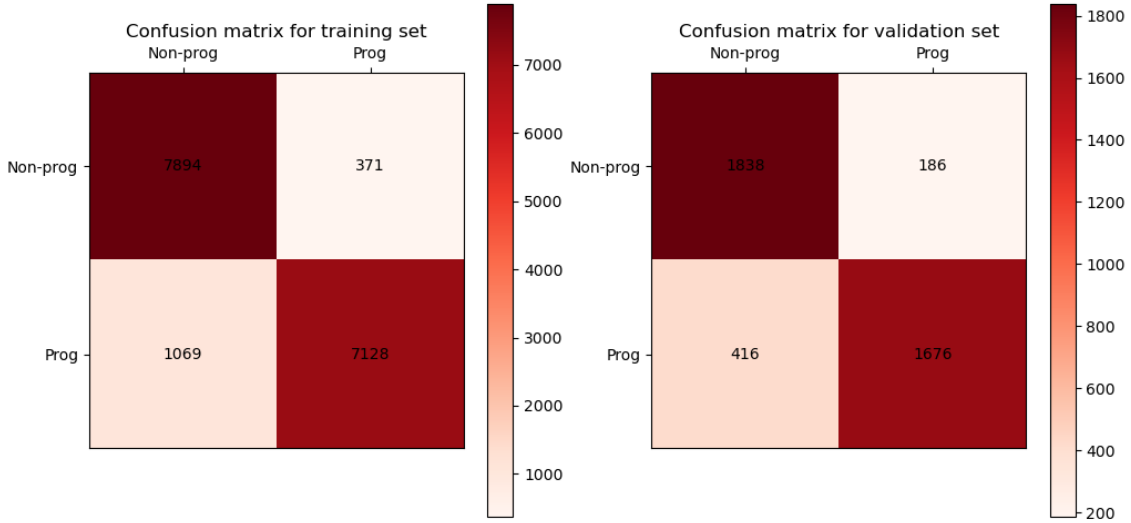

Figure 7: Confusion matrix for the dnn.h128h32 model

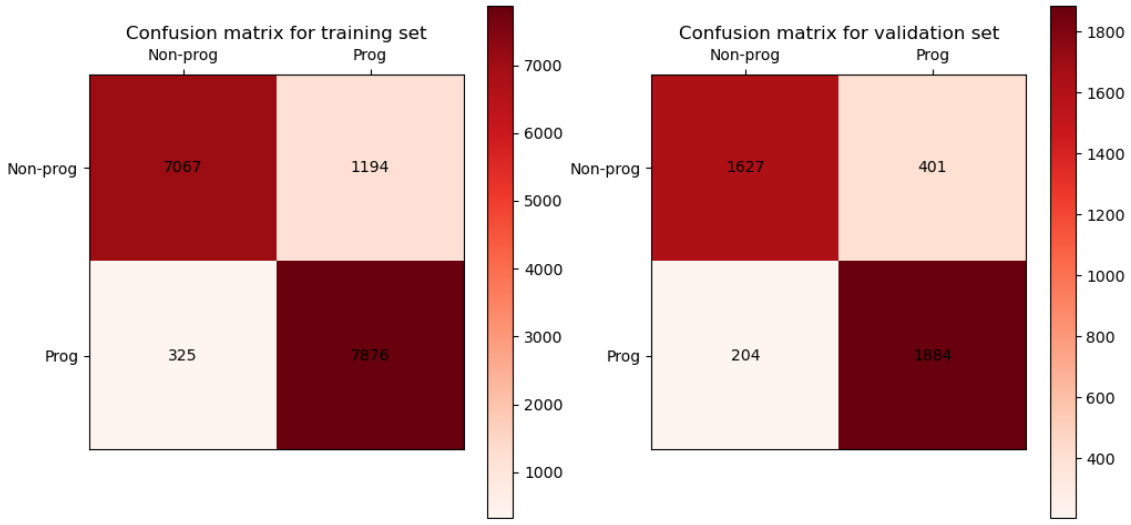

Figure 8: Confusion matrix for the dnn.h64h32 model

3.4 Performance on test set

Table 2 shows the performance of all 3 baseline dnn models on the test set. Our baseline classifiers tends to classify songs as prog songs more often. The accuracy for prog songs and nonprog songs fluctuate quite much individually, but the overval accuracy is somewhat more stable. This also suggests there is some degree of overfitting for our classifiers.
Plus, there are 13 songs from the file 'Other' are classified as progressive songs.

Model name	Overall accuracy	Accuracy for prog	Accuracy for nonprog
dnn.h128	0.64	0.90	0.43
dnn.h128h32	0.68	0.73	0.63
dnn.h64h32	0.62	0.96	0.34

Table 2: Performance of baseline models on test set

Figure 9,11 and 10 shows the confusion matrices.

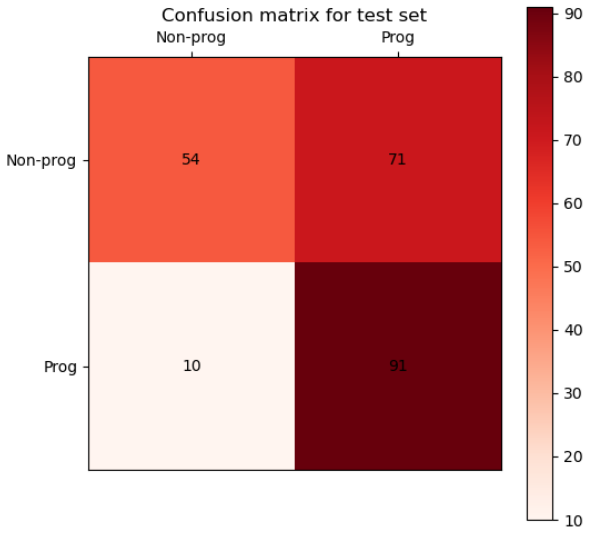

Figure 9: Confusion matrix for the dnn.h128 model on test set

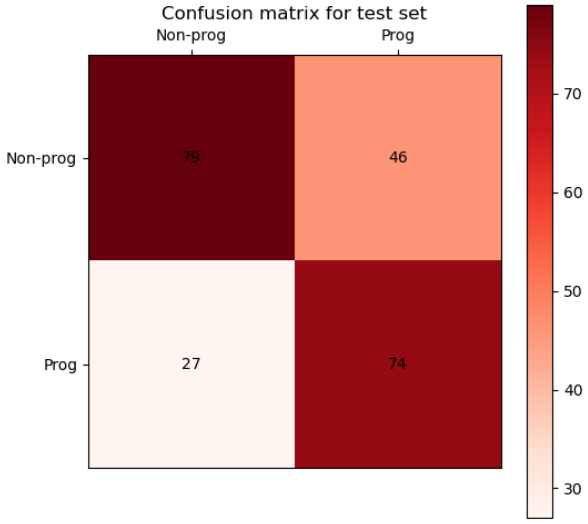

Figure 10: Confusion matrix for the dnn.h128h32 model on test set

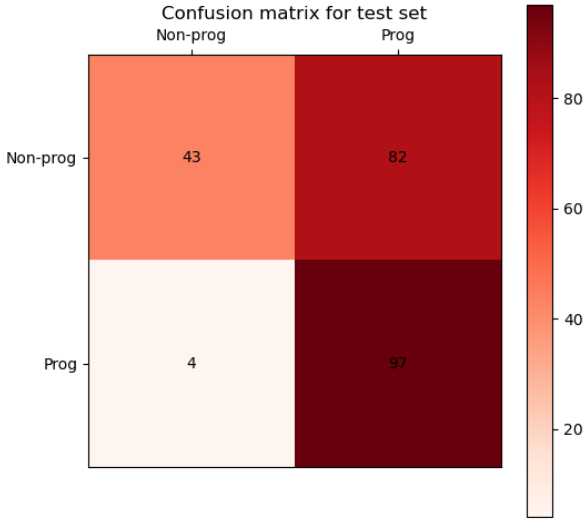

Figure 11: Confusion matrix for the dnn.h64h32 model on test set

4 Boosted decision tree classifier

4.1 Motivation

Neural networks are boosted decision trees are two go-to method for classification on tabular data. In the last section, the tried out the DNN baseline classifier and the overall accuracy is about 0.65. In this section, we switch to boosted decision trees and add more features to help with the overfitting issue. Adding dropout layers would reduce the overfitting in principle. However, from our experiment, adding a dropout layer after the hidden layer does not improve the overall accuracy even though it reduces the overfitting. Since the boosted decision trees consists of an ensemble of trees, there is hope that the overfitting can be kept low and the boosting will help learn useful features from the data.

4.2 Features

The additional features compared with the DNN baseline features are listed below. The total number of features is 59 (46 + 13).

- 12 choma stft coefficients.
- Tempo.

4.3 Boosted decision tree model

The configuration of our boosted decision tree is listed in Table 3.

Parameter	Value
`max_depth`	4
`eta`	0.5
`subsample`	0.5
`gamma`	0.01
`objective`	`binary:logistic`
`early_stopping_rounds`	10

Table 3: Boosted decision tree training parameters

4.4 Performance

The performance of the xgboost classifier is listed in Table 4. The classifier does well on the prog songs but does a much worse job on identifying nonprog songs. Figures 12 to 14 show the confusion matrix for each of the data samples.
In addition, there are 11 songs from the file 'Other' are classified as progressive songs.

Dataset	Overall accuracy	Accuracy for prog	Accuracy for nonprog
Training set	0.996	1.0	0.994
Validation set	0.929	0.959	0.911
Test set	0.659	0.851	0.504

Table 4: Performance of the boosted decision trees

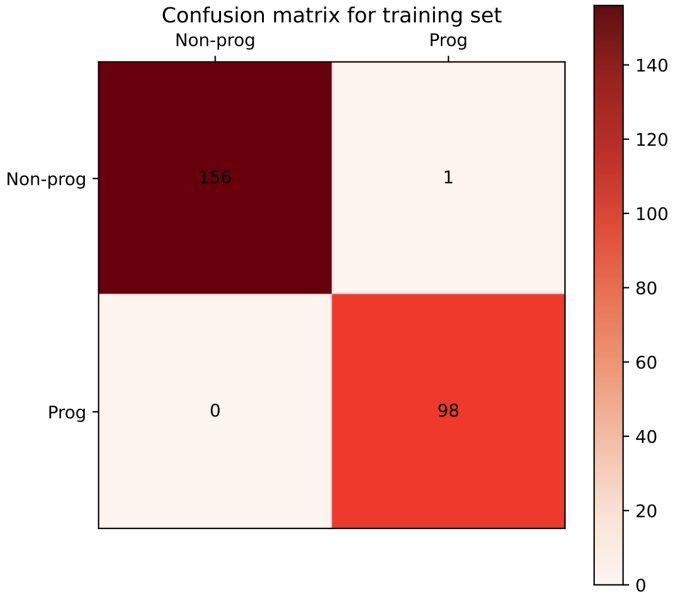

Figure 12: Confusion matrix for training set

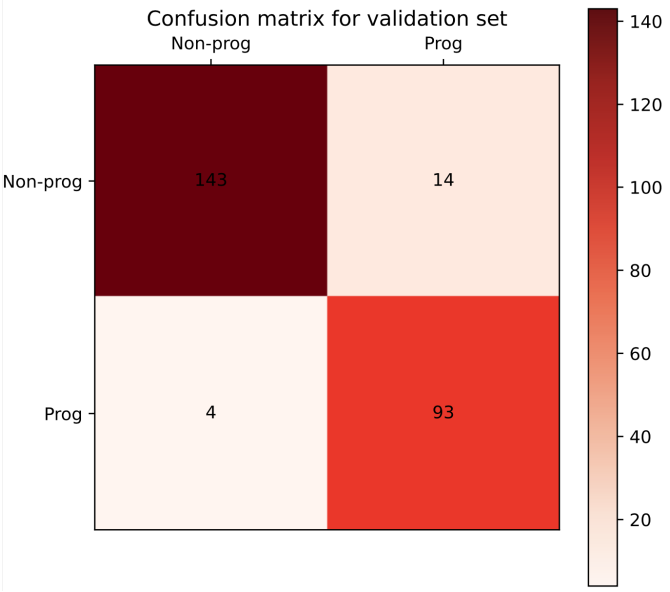

Figure 13: Confusion matrix for validation set

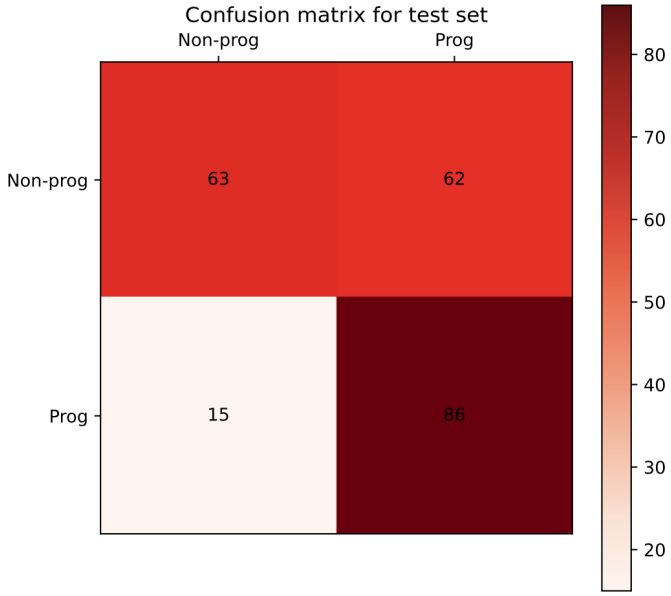

Figure 14: Confusion matrix for test set

4.5 Light Gradient Boosting Machine

4.5.1 Introduction of lightgbm

Apart from the above implemented model—xgboost, we also implemented lightgbm (Light Gradient Boosting Machine). Lightgbm is a gradient boosting model based on decision tree algorithm and takes less time on large training data set. Different from other boosting framework, this algorithm splits the tree leaf-wise, while xgboost and others are based on level-wise or depth-wise. Figures 15 and 16 show the structure of level-wise and leaf-wise respectively.

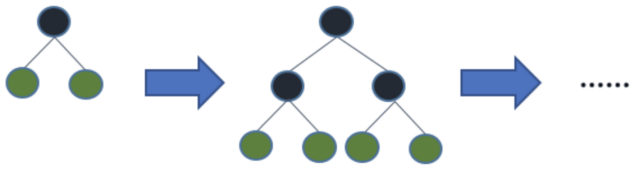

Figure 15: level-wise tree growth [3]

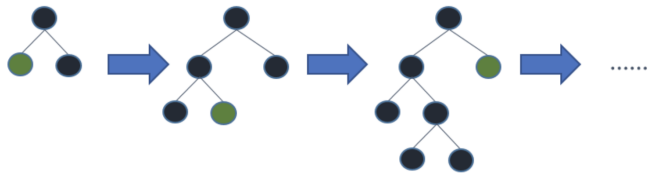

Figure 16: leaf-wise tree growth [3]

4.5.2 Advantages of lightgbm

Compared with xgboost, we found that lightgbm has faster training speed and higher efficiency. Besides that, it is friendlier with computers which has limited memory resource. Furthermore, since it produces complex trees using leaf-wise split approach, it achieves satisfying accuracy.

4.5.3 Implementation and performance

After installing the lightgbm package, we set and tuned parameters as shown in Table 5. The accuracy is slightly lower than xgboost and they both have compromised results on progressive songs. Generally, the overall accuracy score on validation set is around 0.80 and 0.62 on test set.

Parameter	Value
`learning_rate`	0.1
`boosting_type`	gbdt
`objective`	binary
`metric`	binary_logloss
`objective`	binary:logistic
`sub_feature`	0.5
`num_leaves`	20
`min_data`	50
`max_depth`	5

Table 5: Boosted decision tree training parameters

5 RNN model based on LSTM

5.1 Introduction of LSTM

Traditionally, a basic neural network is simply a collection of connected units or nodes. The network simulates the structure of biological brain and transmit signals, however, it lacks the ability of memorizing information. Here comes the Recurrent neural networks (RNN), it solves this issue and allows information to persist. This feature seems extremely useful to make predictions, but the simple RNN only has short-term memory. To solve this, scientists later developed Long short-term memory (LSTM), which avoided the vanishing of gradient so that the RNN is allowed to predict based on long-term memory.

5.2 Parameters

The configuration of our RNN model based on LSTM with two hidden layers is listed in Table 6.

Parameter	Value
`Activation`	sigmoid
`loss`	binary_crossentropy
`optimizer`	adam
`metrics`	acc
`Dropout`	0.2
`return_sequences`	False

Table 6: lstm parameters

5.3 Performance and some comments

The average performance of lstm in 5 trials is listed in Table 7. As we can tell from the table, the performance of classifier is compromised on both validation and test sets. Though we tuned the parameters or added more layers into the model the results changed very slightly.Figurefig. 17 shows the confusion matrix on test data set. Besides, there are 10 songs from the file 'Other' are classified as progressive songs.

Based on some online resources, we speculate that the unsatisfying result might be caused by the property of lstm. Lstm has excellent performance on time series prediction, whereas we tired it on randomly distributed binary classification problem.

Dataset	Overall accuracy	Accuracy for prog	Accuracy for nonprog
Training set	0.610	0.597	0.613
Validation set	0.608	0.601	0.613
Test set	0.601	0.593	0.604

Table 7: Performance of lstm

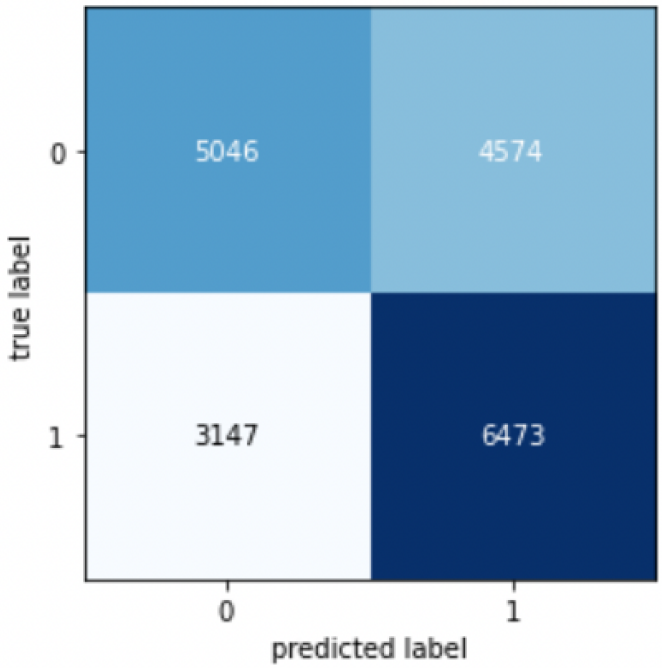

Figure 17: confusion matrix of test data set('1' stands for progressive rock song '0' stands for non-progressive rock song)

6 CNN model based on Resnet50V2

6.1 Motivation

One of the best models from the previous years used a CNN model to recognize the melspectrogram. It quite surprising they came up with a multi-layer CNN model instead of using a pretrained models, which is faily common in image recognition tasks. Since there is some temporal information preserved in the melspectrogram without time averaging, the CNN model is expected to do better.

6.2 Model and training procedure

We used the Resnet50V2 model came with the keras library without the fully connected layers. A hidden layer of 64 relu units is added, followed by a single sigmoid unit for the output. Due to limited computation resource, during the first 10 epochs of the training, the parameters of the Resnet50V2 layers are frozen. For the next 20 epochs, the parameters are allowed change.

6.3 Performance

The performance of our CNN model is tabulated in Table 8 and the confusion matrices are shown in Figures 18 and 19. As well, there are 12 songs from the file 'Other' are classified as progressive songs.

Dataset	Overall accuracy	Accuracy for prog	Accuracy for nonprog
Training set	0.980	0.967	0.991
Validation set	0.910	0.881	0.934
Test set	0.701	0.950	0.512

Table 8: Performance of the cnn model

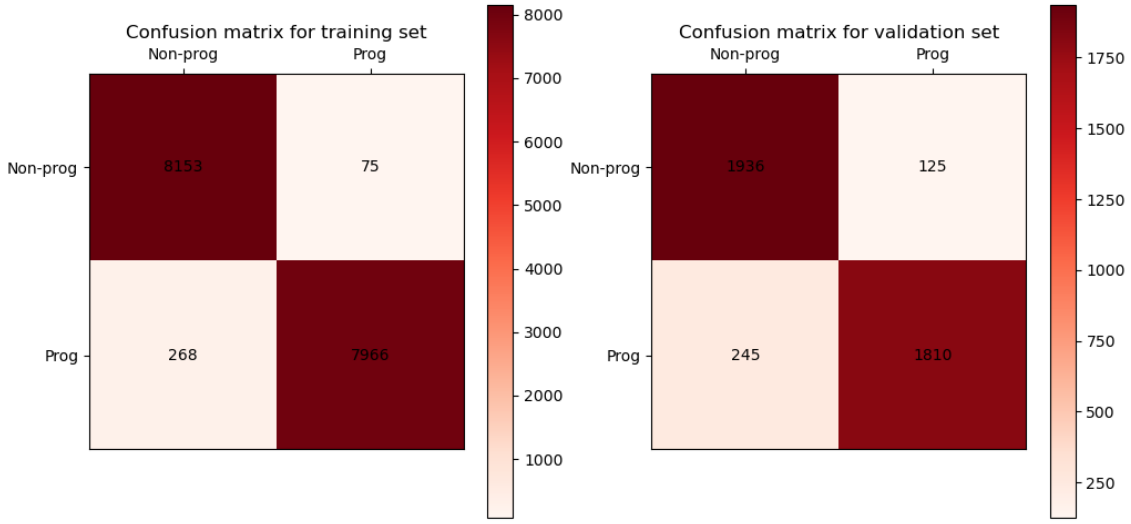

Figure 18: Confusion matrix for the CNN model

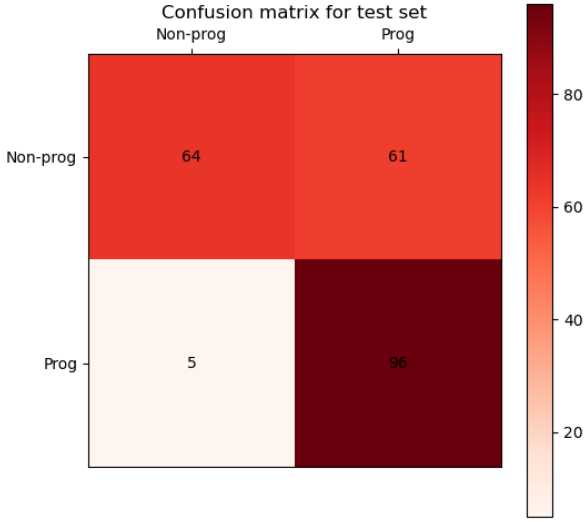

Figure 19: Confusion matrix for the CNN model

7 Summary

From the above mentioned models, some non-progressive songs are misclassified as progressive songs which indicate that these songs might have progressive features. Here is the list:

- 07 - Signals (Orchestrion Sketch).mp3
- 02 QUEENS OF THE STONE AGE-NO ONE KNOWS.mp3
- 01_Mere Sapnon Ki Rani.mp3
- Reptile.mp3
- Kiss Me, Kiss Me, Kiss Me-The Cure-07-The Snakepit.mp3,
- A Dream In Red Mansions.mp3, 07 By Myself.mp3.
- Radiohead - Spectre.mp3

In this project, we tried on a number of machine learning models to classify prog and nonprog rocks. Our best method utilizes Resnet50V2 to classify the melspectrogram of the song segments. The best overall accuracy on test set is 0.70.

References

[1] Wikipedia. Progressive rock. `https://en.wikipedia.org/wiki/Progressive_rock`, April 2021.

[2] Wikipedia. Mel-frequency cepstrum. `https://en.wikipedia.org/wiki/Mel-frequency_cepstrum`, April 2021.

[3] Pranjal Khandelwal. lightgbm vs xgboost. `https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/`, June 2017.