CAP 6610

Machine Learning Project

Project Report

By

Group # Moving Pictures

Hao-yu Liao

Ryon Kennedy,

Anirban Bhattacharya

Bhaskar Mishra

Spring 2021

University of Florida

Gainesville, Florida

Abstract

In our final project for our CAP6610 Machine Learning class, we tackle the problem of music genre classification. Specifically, given a dataset provided to us by Dr. Rangarajan containing both songs from and not from the genre of Progressive Rock, and were tasked with training a classifier that could determine whether or not an arbitrary song was from the Progressive Rock genre or not. We experimented with a variety of methods and will use this report to discuss and analyze the results.

Keywords: Machine learning, Deep learning, Transferring learning, Classification

1. Introduction

The application of Machine Learning to audio-based data encompasses a large sector of modern-day research in the field. In our final project for our CAP6610 Machine Learning class, we tackle the problem of music genre classification. Specifically, given a dataset provided to us by Dr. Rangarajan containing both songs from and not from the genre of Progressive Rock, we are tasked with training a classifier that can determine whether an arbitrary song is from the Progressive Rock genre or not. We experiment with a variety of methods including a variety of basic machine learning algorithms, feature extraction approaches, and deep neural network architectures. Through this report, we will discuss each of these approaches and analyze their performances, specifically focusing on our best performing classifier.

2. (20 points) Performance of the classifiers on the Non-Prog and Prog training sets

2.1 (5 points) Discussion of the techniques underpinning your classifiers

We apply the librosa library to extract the feature from each song and the machine learning and deep learning models are implemented in the study. Also, transferring learning is applied in this study. The pre-trained networks such as the GoogLeNet and ResNet50 will show how powerful on identifying the Prog and Nonprog songs in this study.

2.1.1 Feature selection

The primary feature we focus on in this project is the dB spectogram generated from the application of short-time fourier transform (STFT) to the individual songs. These spectograms portray the audio-based information from the songs in an image-based format. We use the librosa library to transfer each song into the dB spectrogram with logarithmic scale as shown in Figure 2.1.1-1, and we retrieve 1 minute of each song from the middle points. For example, if a song has 3 minutes, the spectrogram will plot from 1:30 to 2:30. After retrieving the songs into dB spectrogram images, we resize the images into 200x200 sizes as shown in Figure 2.1.1-2. The resized images will be inputted into the model, and the output will be category labels such as Nonprog or Prog. The Prog images are labeled as 1, and the Nonprog images are labeled as 0. We have 479 songs made of 280 Nonprog and 199 Prog. 80% of the dataset is for training, and 20%

of the dataset is for testing. The input is resized image, and the output is the labeled values (0 or 1). The process of implementation is shown in Figure 2.1.1-3.

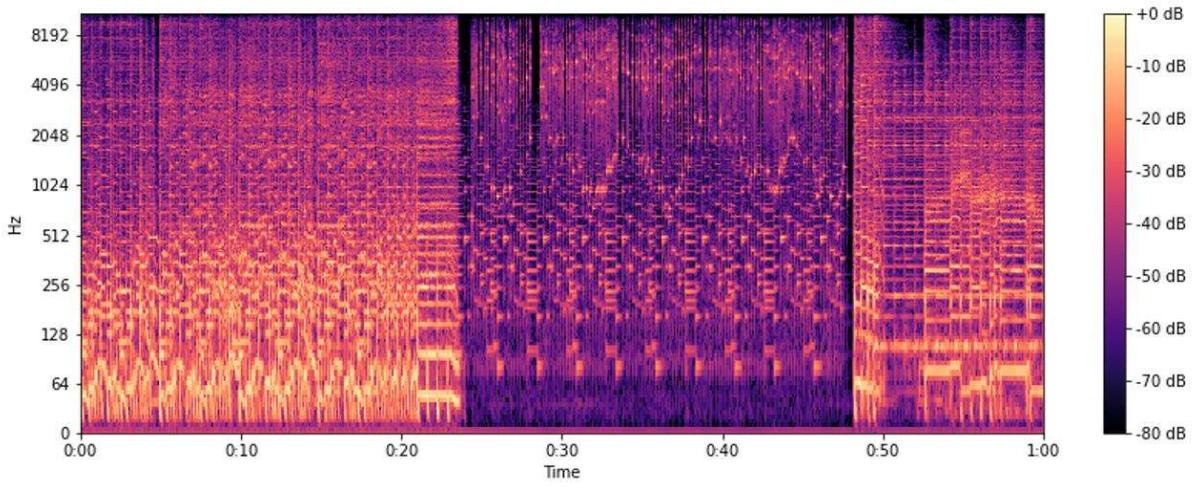

Figure 2.1.1-1 Transfer the song into dB spectrogram with logarithmic scale (Example of -04- Knots.mp3).

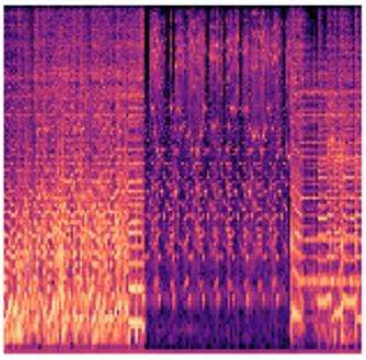

Figure 2.1.1-2 Resize dB spectrogram into 200 by 200 (Example of -04- Knots.mp3).

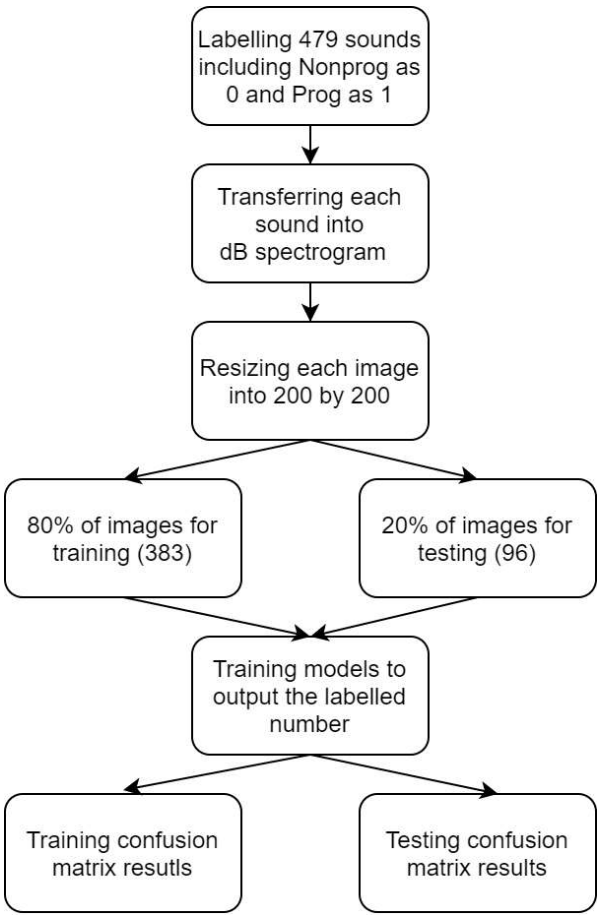

Figure 2.1.1-3 Flowchart of the implementation

For the deep learning models, we input each image as RGB channels to models. The shape of input will be (479, 3, 200, 200). However, for the basic machine learning models we took all the RGB values for each pixel and fed them into a single feature vector for each song instance.

2.1.2 Fisher Discriminant

For one of the baseline classifiers, we choose to use the Fisher Linear Discriminant because it is one of the most common methods used in statistics and other fields. It is used to find a linear combination that separates the data and was one of the earliest machine learning algorithms. It is closely related to other unsupervised machine learning methods such as PCA, Principal Component Analysis.

2.1.3 Linear SVM

We chose Linear SVM for one of the baseline classifiers because it is one of the most common algorithms used in machine learning. Called Support Vector Machines, Many SVMs utilize kernels, but this implementation just uses a simple linear model. SVMs are one of the most thoroughly tested and robust generic machine learning algorithms.

2.1.4 Regression Tree

We chose Regression Tree for one of the baseline classifiers because it is commonly used in machine learning. It constructs/uses a decision tree with the leaves representing the class labels and are valued for their simplicity and are commonly used in data mining. The none leaf nodes represent decisions points in the structure/algorithm.

2.1.5 Random Forests

We chose Random Forests for one of the baseline classifiers because it is one of the most successful non deep learning classifiers used for machine learning. In many analyses, Random Forests performs better than any other generic machine learning algorithms for multiple types of data such as malware for example. Random Forests is an ensemble learning method that works by constructing a multitude of decision trees.

2.1.6 CNN

A convolution neural network (CNN) as the classifier for our project by using Pytorch. The architecture of CNN is shown in Figure 2.1.6-1. We use the ReLU activation for each layer, and the last layer is Sigmoid activation. Also, we apply the batch normalization and max-pooling in the CNN. The proposed CNN has 5 convolution layers and one fully connected layer. The optimizer is stochastic gradient descent (SGD) with a 0.001 learning rate and 0.9 momentum. The loss function is binary cross entropy. The batch size for each iteration is 5. We take 10,000 epochs to train the CNN model.

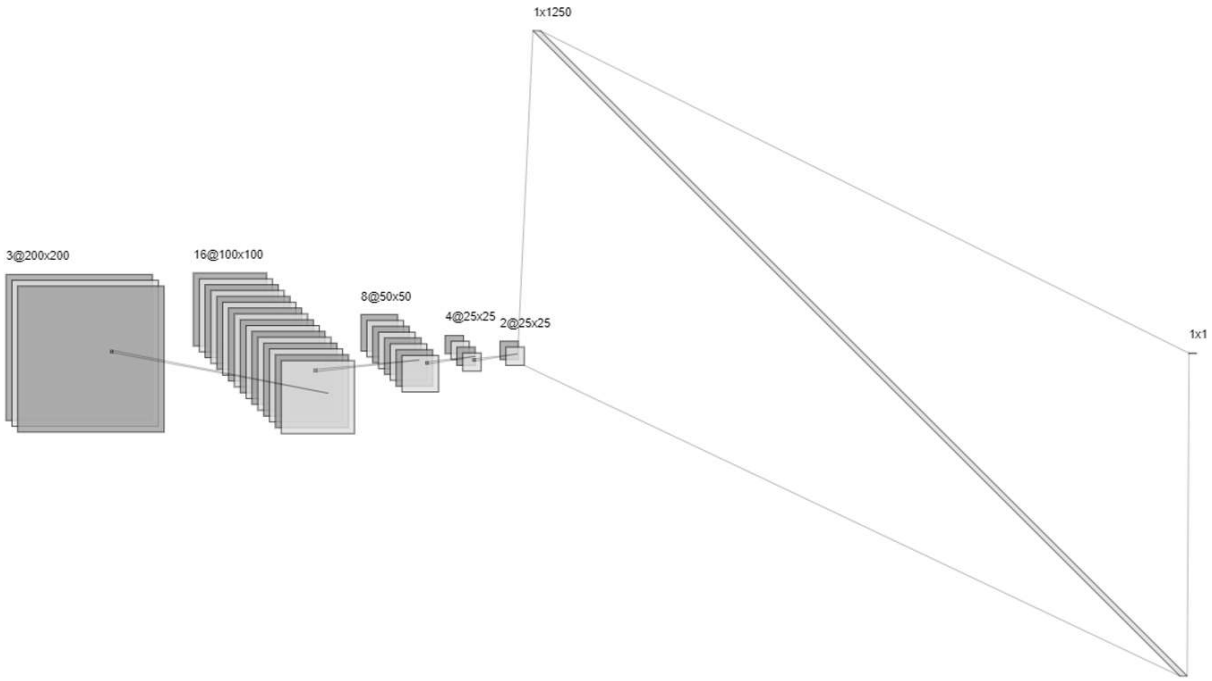

Figure 2.1.6-1 The architecture of CNN in the project.

The project is a binary classification problem because of there are only two categories: non progressive and progressive. Based on the problem, the techniques of our proposed model are following. First, we take the ReLU function for each layer except for the last layer. Because the

project is a binary problem, after taking the ReLU function, each pixel will be transferred from 0 to a positive number. If some information in the feature maps of the convolution layer is not important, the information will be computed as 0 based on the ReLU function. The important information will be counted as a positive number. For the last layer with Sigmoid function because the labeled values are 0 or 1. The sigmoid function will output from 0 to 1. It is suitable for binary problems. Second, we take the binary cross entropy as the loss function because the classification problem is binary. Third, we also take the batch normalization in our architecture. The batch normalization can speed up the training process and enhance the training performance. Finally, the number of channels in the current layer is 2 times of the next layers before connecting a fully connected layer. Because the output value is 0 or 1 and the last layer is fully connected, the 2 times will be appropriate for the classification problem. The detail of the proposed model is shown:

```
Sequential(
  (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (4): Conv2d(16, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): ReLU(inplace=True)
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (8): Conv2d(8, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (9): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (10): ReLU(inplace=True)
  (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (12): Conv2d(4, 2, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): BatchNorm2d(2, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (14): ReLU(inplace=True)
  (15): Flatten(start_dim=1, end_dim=-1)
  (16): Linear(in_features=1250, out_features=1, bias=True)
  (17): Sigmoid()
)
```

2.1.7 GoogLeNet

GoogLeNet is another architecture used in this paper, as shown in Figure 2.1.7-1. It was first proposed by Szegedy et al. in 2014 [1]. It is the first version of Inception, namely Inception-v1. Many researchers used GoogLeNet to classify images in different applications. Singla et al. (2016) applied GoogLeNet to identify food images using transfer learning [2]. Lee et al. (2018) used it to improve the performance of recognition on Korean characters [3]. Jahandad et al. (2019) created an offline signature verification system by using GoogLeNet [4]. The GoogLeNet was trained by over a million images with 1000 different objects and has 22 layers (27 layers considering pooling layers. GoogleNet uses the global average pooling at the last inception module.

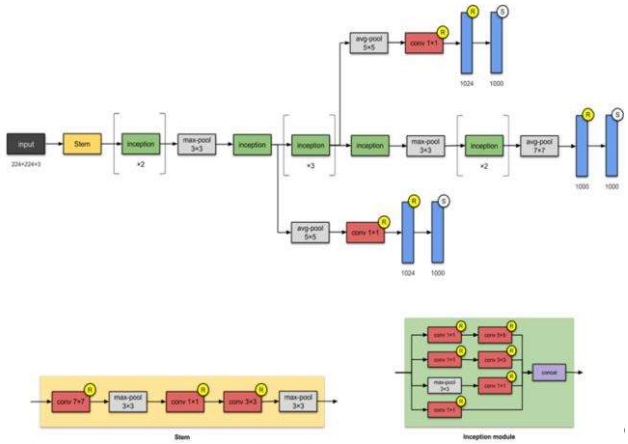

Figure 2.1.7-1: Architecture of GoogLeNet (Inception-VI) obtained from ref [5] based on [1].

2.1.8 ResNet50

Stacking layers and making deeper networks will cause the accuracy to get saturated and degrade rapidly [6]. To address this issue, Microsoft Research launched Residual Neural Network (ResNet) in 2015, as a novel architecture with "shortcut connections" (or skip connections, residuals) [6] that skips one or more layers and has the heavy batch normalization function [7] in building models deeper. ResNet is a deeper trained Neural Network while maintaining lower complexity compared to VGGnet [8]. The original models (ResNet-50, ResNet-101, and ResNet-152) were used in ILSVRC and COCO 2015 competitions, which won the 1st places in: ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation [6].

2.2 (15 points) **Discussion and explanation of the performance of the classifiers on the training set**

2.2.1 Training results

Figures 2.2.1-1 to 2.2.1-7 show the training results of each model. The Fisher Discriminant and Linear SVM have 72% and 99% training accuracy respectively, and other models have 100% training accuracy.

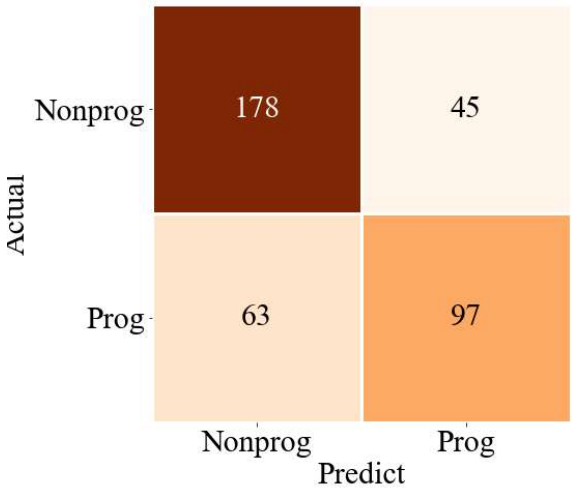

Figure 2.2.1-1: Fisher Discriminant Confusion Matrix for Training Data with 72% accuracy.

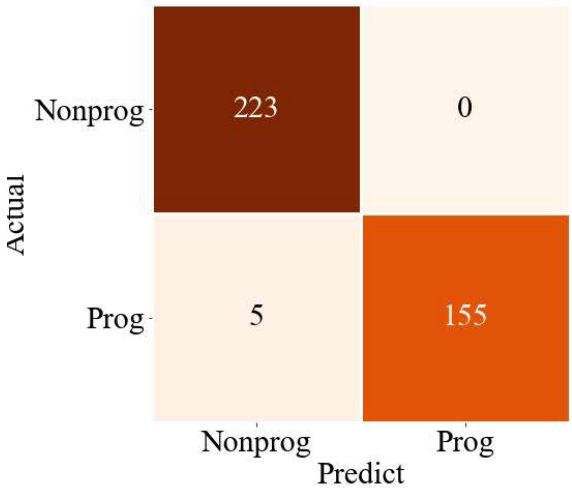

Figure 2.2.1-2: Linear SVM

Confusion Matrix for Training Data with 99% accuracy.

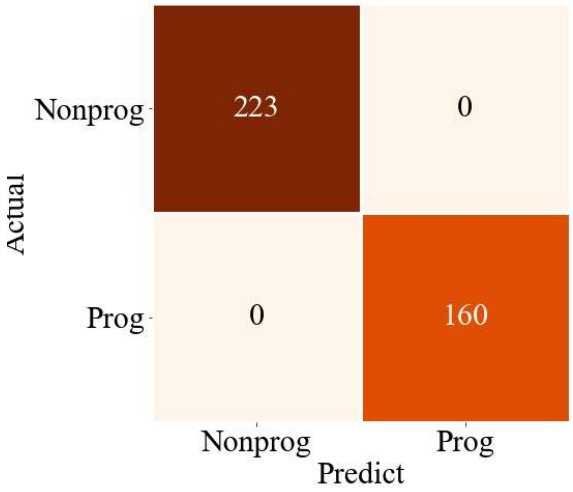

Figure 2.2.1-3: Regression Tree Confusion Matrix for Training Data with 100% accuracy.

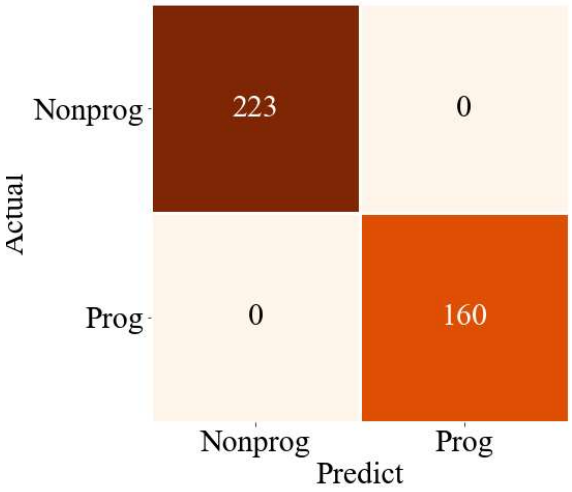

Figure 2.2.1-4: Random Forests Confusion Matrix for Training Data with 100% accuracy.

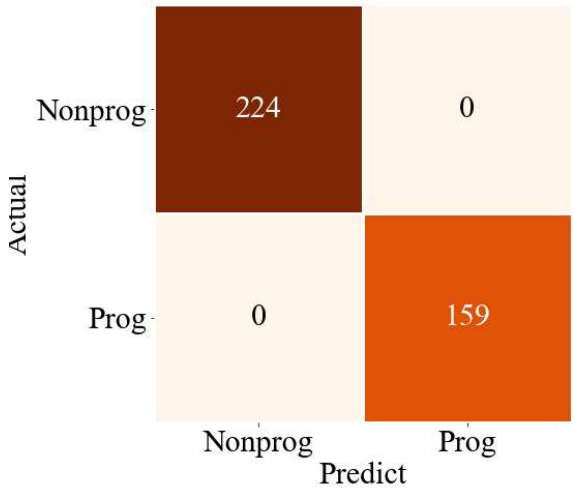

Figure 2.2.1-5: CNN Confusion Matrix for Training Data with 100% accuracy.

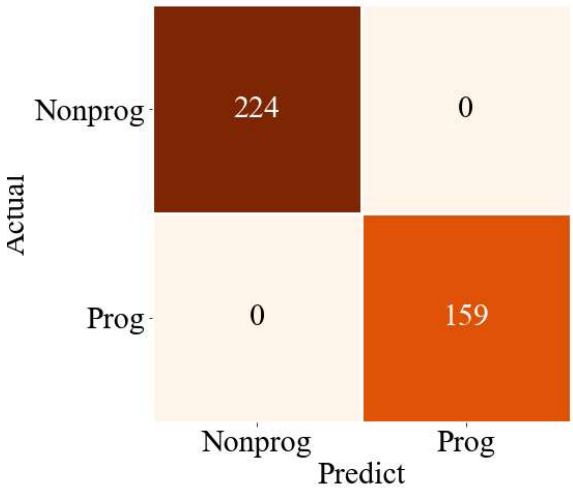

Figure 2.2.1-6: GoogLeNet Confusion Matrix for Training Data with 100% accuracy.

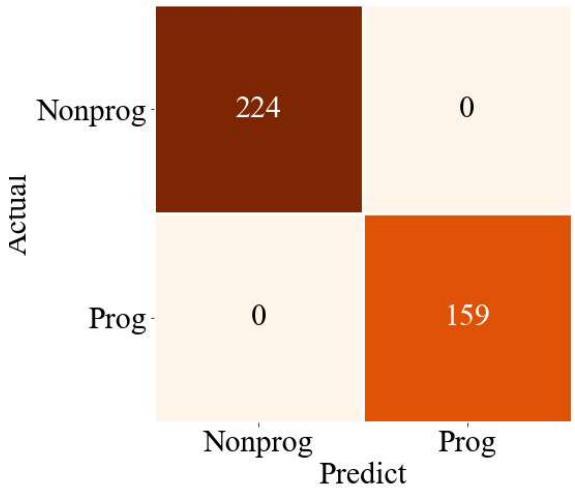

Figure 2.2.1-7: ResNet50 Confusion Matrix for Training Data with 100% accuracy.

2.2.2 Discussion and explanation of the performance of training set

Table 2.2.2-1 presents the training accuracy results of each model. All models have 100% training accuracy, but the Fisher Discriminant and Linear SVM have 72% and 99 % training

accuracy, respectively. It is expected that all our more advanced models are able to achieve 100% training accuracy as our training set is relatively small, and it is very easy for the models to simply memorize identifiable parts of each of the progressive songs. Since Fisher Discriminant is a purely linear algorithm, it makes sense that it wasn't able to achieve a perfect training accuracy.

Table 2.2.2-1 The training accuracy results of each classifier.

Classifier	**Training Accuracy**
Fisher Discriminant	72%
Linear SVM	99%
Regression Tree	100%
Random Forests	100%
CNN	100%
GoogLeNET	100%
ResNet50	100%

3. (40 points) **Performance of the classifiers on the test set**

3.1 (15 points) **Demo of performance of classifier on test set.**

Figures 3.1-1 to 3.1-7 show the test results of each model. The machine learning models such as Fisher Discriminant, Linear SVM, Regression Tree, and Random Forests have below 70% test accuracy. However, the deep learning models have above 70% test accuracy. Among in deep learning models, the GoogLeNet is the best model with 80% test accuracy.

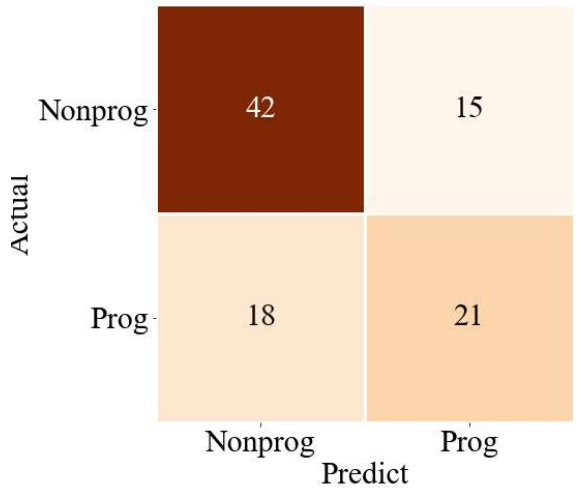

Figure 3.1-1: Fisher Discriminant Confusion Matrix for Test Data with 66% accuracy.

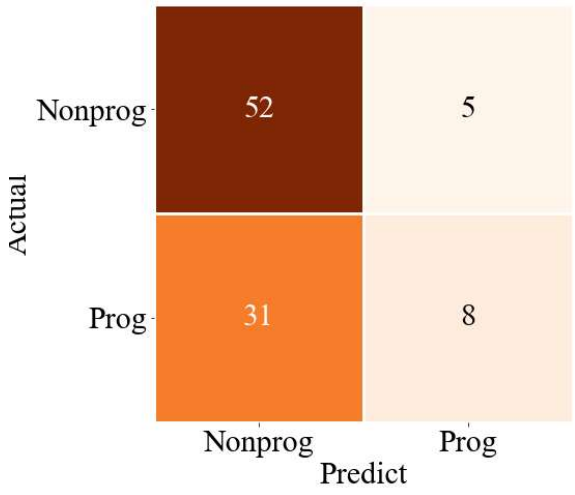

Figure 3.1-2: Linear SVM Confusion Matrix for Test Data with 63% accuracy.

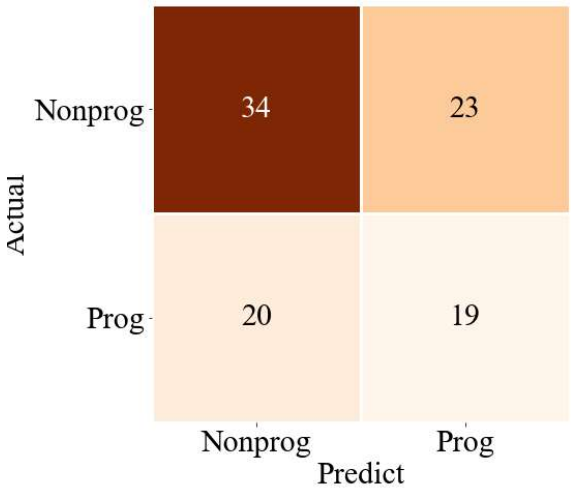

Figure 3.1-3: Regression Tree Confusion Matrix for Test Data with 55% accuracy.

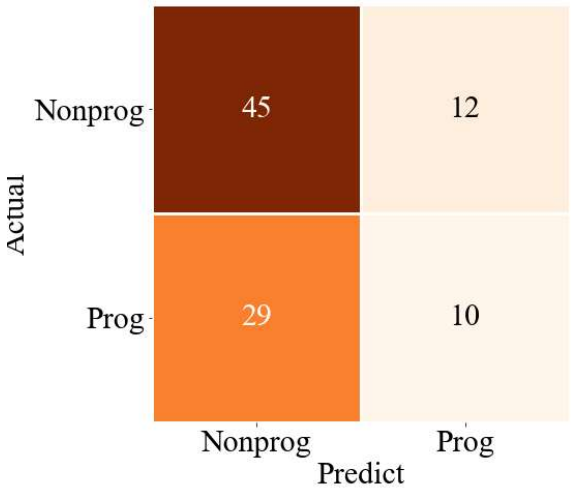

Figure 3.1-4: Random Forests Confusion Matrix for Test Data with 57% accuracy.

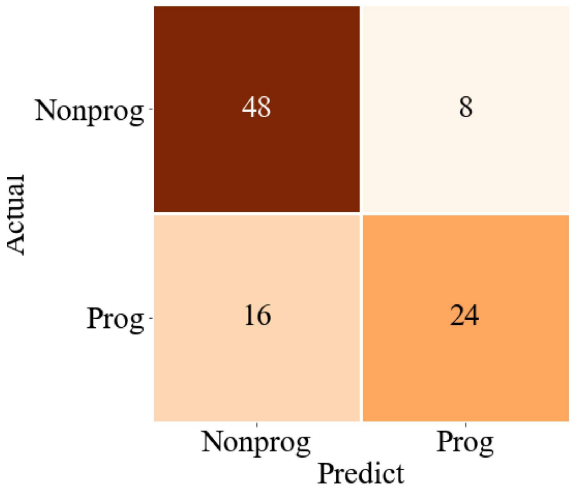

Figure 3.1-5: CNN Confusion Matrix for Test Data with 75% accuracy.

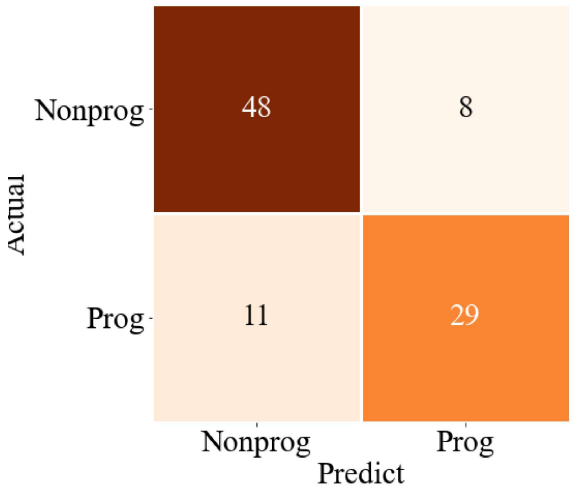

Figure 3.1-6: GoogLeNet (best model) Confusion Matrix for Test Data with 80% accuracy.

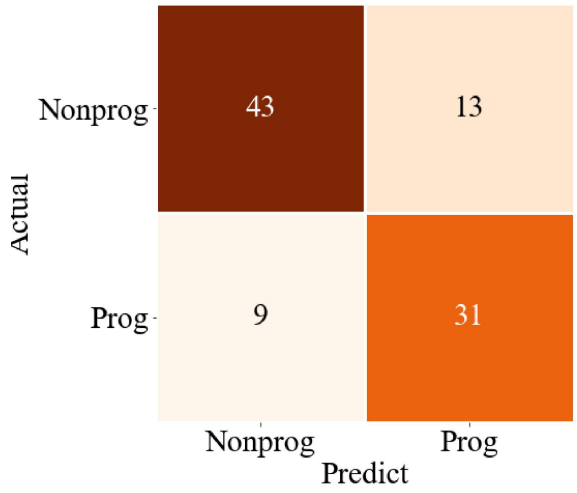

Figure 3.1-7: ResNet50 Confusion Matrix for Test Data with 77% accuracy.

3.2 (25 points) **Discussion and explanation of the performance of test set.**

Table 3.2-1 presents the test accuracy results of each model. For the same reason that the training set accuracy was so high for the more advanced models, we see that the test accuracy is comparatively lower. Each of the models easily overfit on the training set, and thus achieved near perfect training accuracy despite having lower test set accuracy. Of the models, the three models that performed the best were GoogLeNet, ResNet50, and CNN. This is also expected as the CNN-based models most efficiently extract important information from the image-based spectograms. It also makes sense that GoogLeNet and ResNet50 performed better than the CNN, as they used a much larger source of data to train the classifier to have an embedding that is more likely to generalize well. As a result, when applying these techniques to our smaller dataset, by keeping the embedding weights fixed, we made the resulting patterns generalize better to the test set.

Table 3.2-1 The test accuracy results of each classifier.

Classifier	**Test Accuracy**
Fisher Discriminant	66%
Linear SVM	63%
Regression Tree	55%
Random Forests	57%
CNN	75%
GoogLeNet	80%
ResNet50	77%

4 Conclusion

In this project, we have explored a variety of different Machine Learning approaches to training a Progressive Rock classifier. Our best model used a pretrained network known as GoogLeNet and, after being trained on our data, was able to reach a test accuracy of 80%.

5. References

[1] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[2] A. Singla, L. Yuan, and T. Ebrahimi, "Food/non-food image classification and food categorization using pre-trained googlenet model," in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, 2016, pp. 3–11.

[3] S.-G. Lee, Y. Sung, Y.-G. Kim, and E.-Y. Cha, "Variations of AlexNet and GoogLeNet to improve Korean character recognition performance," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 205–217, 2018.

[4] S. M. Sam, K. Kamardin, N. N. A. Sjarif, and N. Mohamed, "Offline signature verification using deep learning convolutional neural network (CNN) architectures GoogLeNet inception-v1 and inception-v3," *Procedia Comput. Sci.*, vol. 161, pp. 475–483, 2019.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[8] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Prog. Artif. Intell.*, vol. 9, no. 2, pp. 85–112, 2020, DOI: 10.1007/s13748-019-00203-0.

[9] H. Lu, H. Zhang, and A. Nayak, "A deep neural network for audio classification with a classifier attention mechanism," arXiv Prepr. arXiv2006.09815, 2020.

6. Appendix

All programs are in the Github as the hyperink https://github.com/haoyuliao/CAP6610-Project. In the link, we don't put any music because of copyright.

6.1 Feature selection program

ExtractFeatures.ipynb will extract features form the songs. The output features are discussed in Section 2.1.1.

6.2 Deep learning programs

All deep learning models are saved in the fold called, Deep learning models. In the fold, the GoogLeNet.ipynb, Resnet50.ipynb, CNN2_BCELoss.ipynb are the main training programs. These main training programs can be run directly.

In addition, the trained parameters are saved in the files such as GoogLeNetEP1000.pth, Resnet50EP1000.pth, and CNN2_SGDlr0.001_BCEL_EP10000_75%.pth.

6.3 Machine learning programs

SciKitLearnClassifiers.py has four machine learning classifiers including Fisher Discriminant, Linear SVM, Regression Tree, Random Forests. The input variables are discussed in Section 2.1.1.

6.4 Other Unmentioned Programs

We experimented with some architectures that didn't make the report due to time constraints and performance. One such architecture was the CAB-CNN architecture introduced in "A Deep Neural Network for Audio Classification with a Classifier Attention Mechanism" by Haoye Lu, et. al. [9]. Though this architecture performed very well in the aforementioned paper, it easily overfitted our training data because of its small size. Such complex deep neural networks require sufficiently large datasets so that memorizing the dataset isn't an easy means to minimizing the error function. The CAB-CNN classifier achieved a final test accuracy of approximately 65%, much lower than the other classifiers trained in this report.