Project Report

Austin Kreulach
CAP6610 - Machine Learning
UFID: 40970653
akreulach@ufl.edu
University of Florida

Ethan Yanez
CAP6610 - Machine Learning
UFID: 21611875
ethanyanez@ufl.edu
University of Florida

May 3, 2024

1 Introduction

This paper covers a software development project focused on creating a classifier to discriminate between progressive rock, hereafter 'prog,' and all other forms of music. Progressive rock is a genre defined by experimentation and fusion of styles which makes differentiating between true prog and prog-adjacent pop or rock a difficult task that is ultimately subjective. Nonetheless the genre has certain hallmarks such as longer songs, longer solos, and more technical instrumentation. The stylistic influence of genres such as jazz, classical and folk sum to a unique sound while creating many commonalities with the origin genres. As a result, this is a difficult problem to create true boundaries operating rather on a you-know-it-when-you-see-it standard.

Having fully-stated the difficulty of the challenge at hand, this paper now covers the approach, methods, and results. The problem is approached in two layers by first segmenting the songs into snippets which bear the brunt of processing before the snippet-level labels are aggregated into the song labels. This is achieved via several different model architectures explored for optimal hyperparameters. The primary structure examined is the convolutional neural network, hereafter CNN. Section 2 will cover the feature extraction process and justify the selected features. Section 3 concerns the loss, aggregation of labels, and other elements of the model evaluation. Section 4 covers the various model architectures and results before section 5 describes the specific performance and architecture of the best performing model. Finally section 6 is a summary of the work performed and possible future directions.

2 Feature Extraction

This section concerns the processing of the data prior to training and classification. In raw form, the songs are very large collections of frequency signals and much work needs to be done in order to efficiently learn from them. Most of this work is attempting to transform an audio task closer to a visual task due to the prolific literature on image classification as compared to the relative scarcity of audio processing research. Much of the information in this section is standing of the shoulders of the past semesters' giants with a few important

exceptions.

The first major task is to make the songs as uniform as possible through controlling the sampling rate and song length. Common audio formats sample at 44,100Hz or 22,050Hz which is the number of data points per second of recording. All the data is downsampled to 11,025Hz which is a reduction by a factor of two to four. This allows for significantly faster training at the cost of some resolution. This is inspired by the common practice in computer vision of using low fidelity images as an optimization which has empirically been found to cause minimal loss in discriminatory power. The next preprocessing problem comes from the decision to cut the songs into ten second snippets. Most songs are not perfectly divisible into ten second snippets so some amount must either be treated uniquely or else discarded. In this pipeline, the decision was made to trim the songs at random points along their length until they were divisible into ten second segments. I.e. if a song is 127 seconds, then seven random one second sections of the song will be removed. This avoids the problem of trimming the start or end of a given song since that might contain important distinguishing factors unique to the genre.

2.1 Feature Selection

Once the input data is sanitized and preprocessed, the next step is to extract meaningful features for the classifier to operate on. Based on prior literature, the features selected were spectrograms, Mel-frequency cepstral coefficients, spectral bandwidths, chromagrams, beat onset envelopes, and spectral centroids. These features represent qualities of the sound such as volume/intensity (spectrogram, beat onset envelope, Mel spectra), pitch class (chromagram), and measures of center or size in the spectrum (spectral bandwidth, spectral centroid). These features are extracted using the library librosa which contains native support for each of these operations. Each feature was normalized prior to use using L2 normalization which provides some benefit over L1 normalization in that is more resilient to outliers.

Once the features were selected and calculated, feature importance was investigated using an initial random forest. The features had some overlap in concept and this represents a fairly significant number of representations. It would be reasonable to suspect that there would be significant redundancy in this feature set. However, the results of the evaluation are in figure 1, which indicate that each of the features provides significant information on their own merit. As a result, all of the features were included in the training and evaluation of the models.

3 Evaluation

Once the data is loaded, processed into feature sets, and encoded into segmented snippet objects, the next step is defining performance of the models. This requires providing an architecture (more on this in section 4), a loss function, the processed training data, and values for hyperparameters. For every model the loss function used was simple cross entropy

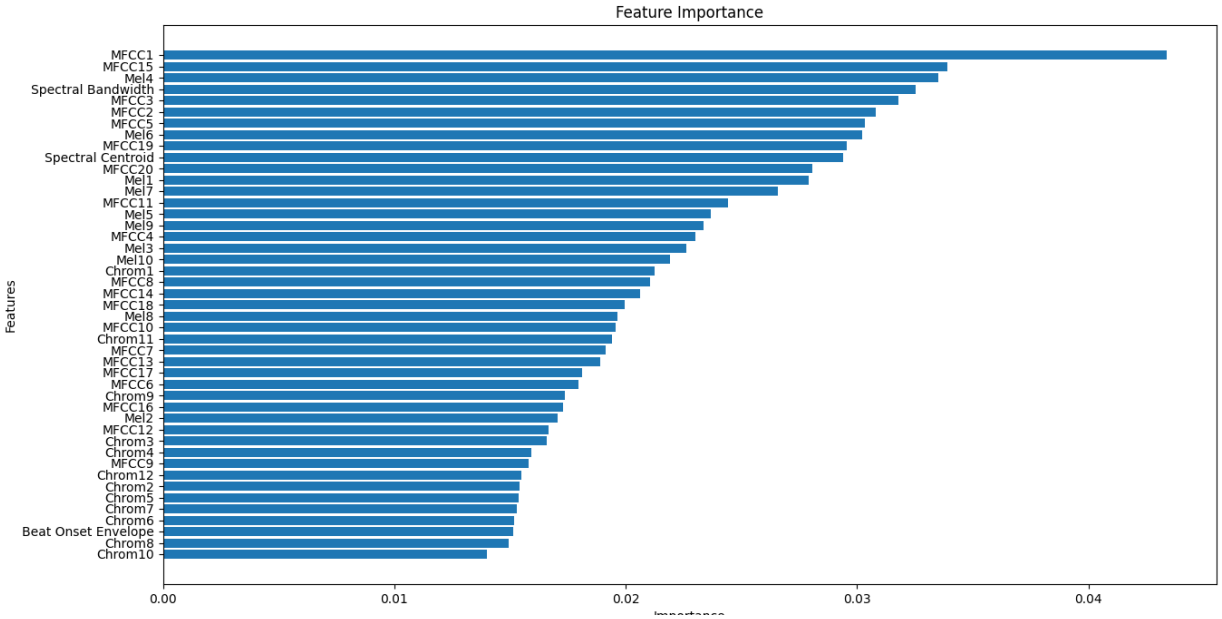

Figure 1: Results of a random forest classifier on initial feature set.

which turns the output into a probability via the softmax function; this was used due to prevalence in the literature. The train-test split used was 80:20 due to the relative richness and variety of the training data. To determine optimal values for the hyperparameters the initial model was run repeatedly to perform a grid search centered on reasonable first estimates. The hyperparameters were the epoch count and learning rate. The train-test split could be considered a hyperparameter but this was not one that was optimized for in this project. The determination was made that training past one hundred epochs provided diminishing returns and a learning rate of 0.001 led smoothly to convergence in reasonable time. During more extended testing, there was a persistent but not consistent error where the loss would freeze in the first epoch that was solved by starting the learning rate at 0.0001 before raising it to the standard 0.001 for all future epochs.

Evaluating each model was done by following the above procedure ten times per model and then averaging the results to account for the random initial seeding of the weights. This provided fairly tight bounds on performance and is likely a good representation of the expected performance. The statistics recorded for each model were the snippet-level and song-level accuracy on the training data as well as the song-level accuracy on the test data. The initial hypothesis was that song accuracy should consistently outperform snippet-level accuracy due to the effect of aggregating labels. Assuming the snippet accuracy was reasonably high, the occasional errant snippet would be eliminated while consistent mislabeling would be less likely. At risk of spoilers, this held true.

Two more points of interest before moving to the main topic. The first is that the snippets were randomized across songs, meaning the segments belonging to a single song were not kept in order nor were the snippets fed to the optimizer in song-order. This is a choice

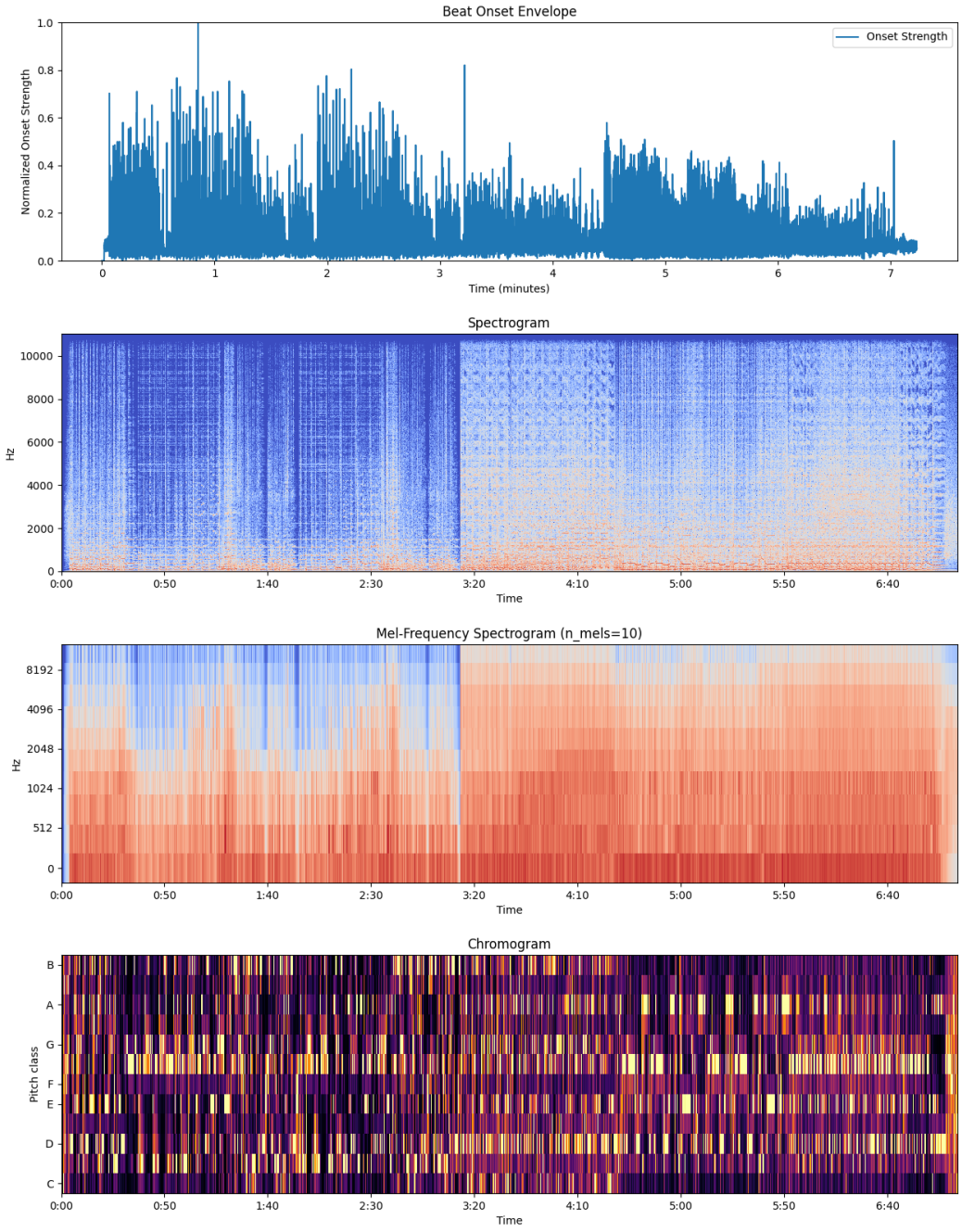

Figure 2: Graph representations of features including, in order, beat onset, spectrogram, MFCC, and chromagram.

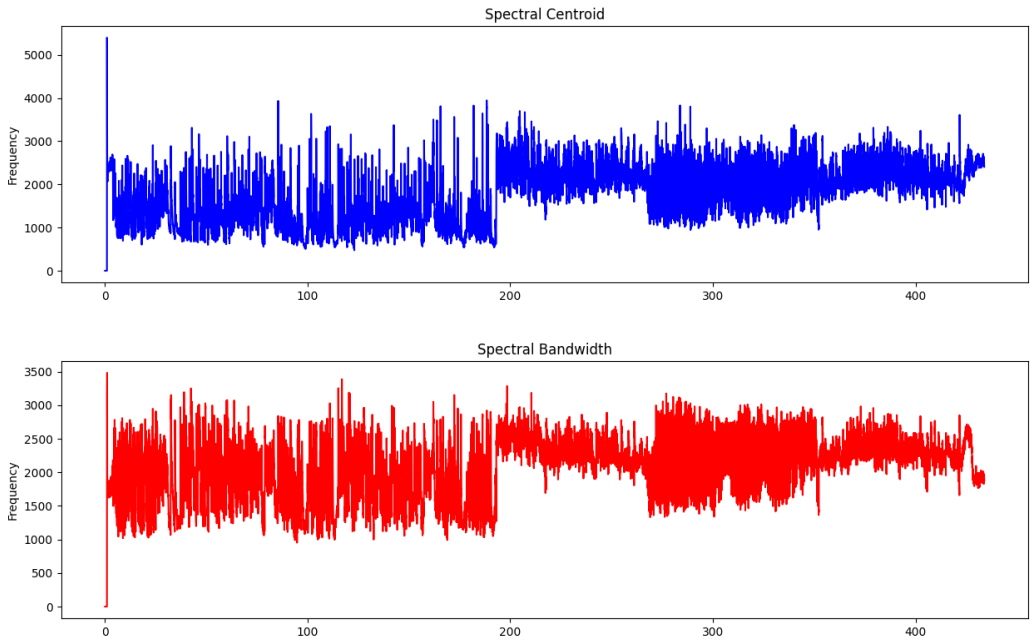

Figure 3: The additional feature graphics for the spectral centroid (top) and spectral bandwidth (bottom).

discussed in greater detail in the conclusion. Secondly, the classifier models all operate on the snippet-level. They assign a label of prog or not prog to each snippet and the task of classifying the songs as a whole (which is the true purpose of this project) is handled by aggregating these snippet labels by a majority vote. This system does fail if there are an equal number of snippets labeled prog and non prog, so there is a failsafe to assign any tied songs as non prog. This was chosen due to the relative prevalence of prog in the dataset but in a more forgiving environment it might be better to default to prog. Further, several other methods were experimented with during the training process. This first method was termed the coin-flip since it is ultimately arbitrary. The second method was more sophisticated. Instead of assigning a label to each snippet and treating them as equal votes, the output certainty was summed and then assigned a label at the song level. This gave greater weight to more confident snippets. Ultimately this did not produce better results so a compromise method was devised to use the sum method when a tie-breaker is needed but otherwise the coin flip method. However even this could not beat the coin flip method, so ultimately that is the policy of the final model. This is likely a product of the significant numerical advantage of non prog songs over prog and it is possible the other methods might perform better with a more balanced data set.

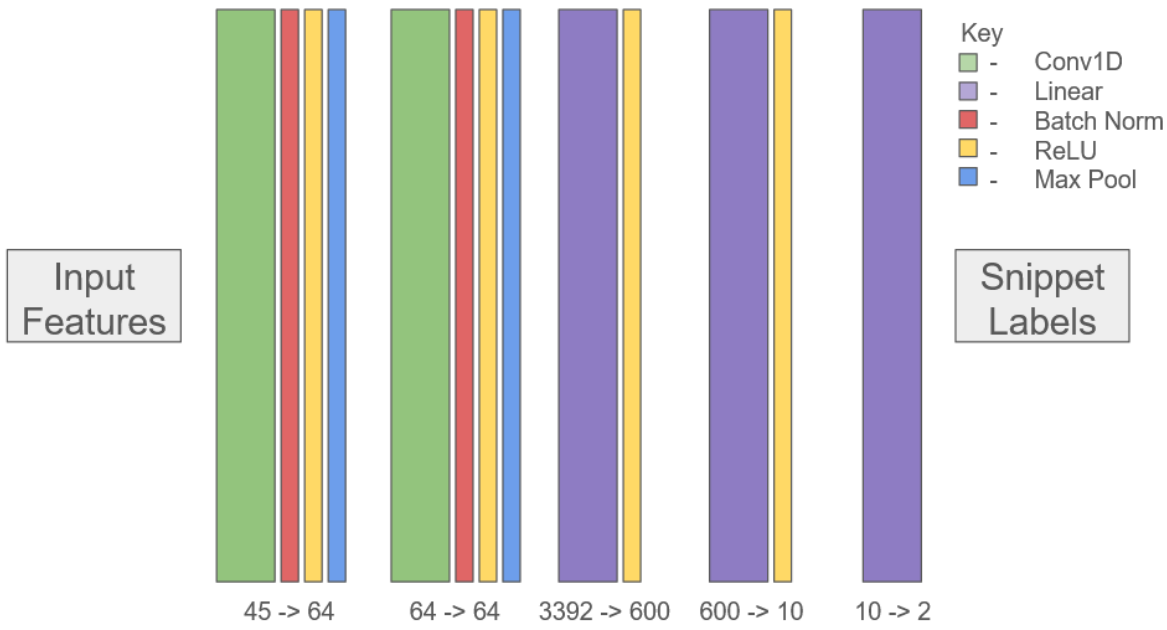

Figure 4: Structure of the base 1D model.

4 Models

4.1 Base 1D CNN

To begin the architecture design stage, the first step was to define a base model from which to develop variations and make comparisons. The benchmark chosen was a near-copy of previous work with the addition of the new features developed in section 2.1. This meant that the input to the models was a set of ten spectrogram bands, twelve pitch classes, twenty MFCCs, the onset envelope, the spectral centroid, and the spectral bandwidth giving forty-five features across 216 data points per ten second snippet. These dimensions are plausible for a well downsampled image analysis problem. From the determination of input dimensions, the next question is the number of layers and the size of the layers. The structure can be seen graphically in 4 but will be reproduced here in written form. The first layer takes the 45x216 input and provides sufficient padding that, after the batch norm, ReLU, and max pooling, returns a 64x107 output where this process is repeated to generate a 64x53 output. At this stage the convolutional layers are done and this structure is flattened before being fed through a triplet of linear layers each followed by a ReLU activation to progress from 3392 to 600 to ten to finally two outputs. These output are then interpreted by the loss function as logits, softmaxed into class probabilities for prog and non prog.

With the model now understood and stated, the initial performance was commendable. The baseline provided 76.9% ±0.7% snippet accuracy and 86.7% ±1.8% song accuracy on the training set and 69.1% ±2.0% snippet accuracy and 81.5% ±2.8% song accuracy on the test set. The additional training time and lower learning rate seem to have provided slight but significant performance improvements over the prior implementation. Although the change

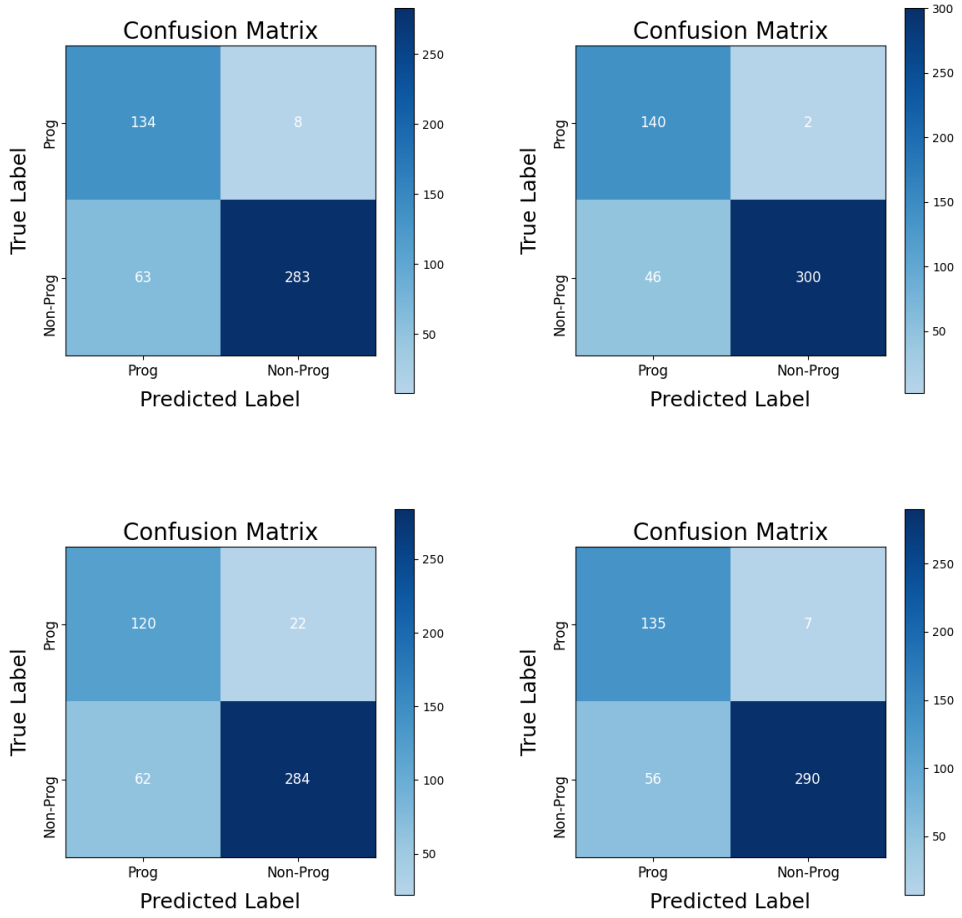

Figure 5: Confusion matrix showing performance of the four candidate models on the train dataset. In order clockwise starting at the top left we have 1D shallow, 1D deep, 2D deep, 2D shallow.

in data set might also explain this change.

4.2 Deep 1D

Once a baseline that performs adequately existed, focus shifted to creating variations that would hopefully perform better. At this stage the changes were largely made in ignorance, simply arbitrary changes. The first of these was deepening the convolutional part of the network from two layers to four. This would provide the model greater granularity in analysis of the snippets, ideally finding some very low-level details that the more shallow network was unable to discern. In addition, this network now increased the number of features at each convolutional layer by a factor of two while still reducing the channels. This took the form of four convolutional layers in the same style as before with each followed by a batch norm, ReLU activation, and max pooling, then three linear layers with ReLU activation. Thus the progression was from 45x216 to 64x107, then 128x53, 256x25, and at the end of

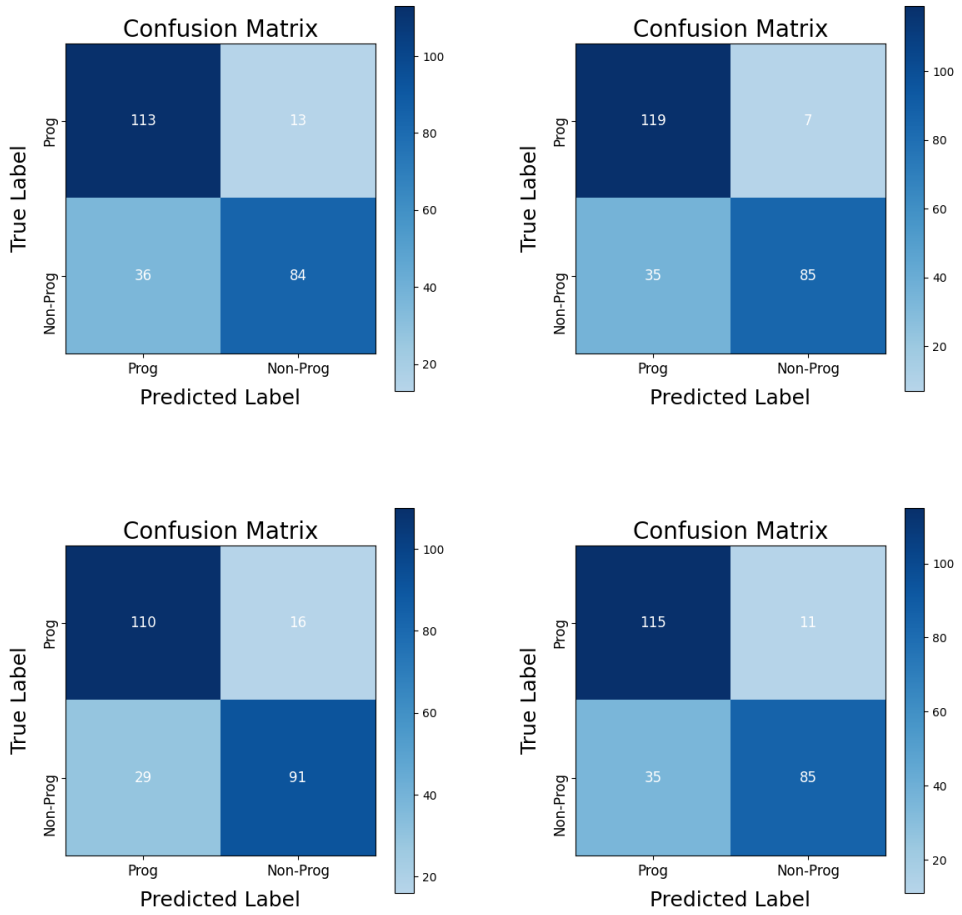

Figure 6: Confusion matrix showing performance of the four candidate models on the test dataset. In order clockwise starting at the top left we have 1D shallow, 1D deep, 2D deep, 2D shallow.

the convolutions 512x11. The linear layers proceeded as before compressing from 512x11 to 600 to 100 to 10 to the final 2 features.

The deep 1D CNN returned 80.7% ±0.6% snippet accuracy and 90.2% ±2.4% song accuracy on the training set and 70.4% ±1.3% snippet accuracy and 82.5% ±2.3% song accuracy on the test set. This is a small but measurable improvement. This seems to indicate there is more complexity to the problem than the shallow model could learn, but perhaps the form is imperfect. This will be discussed at greater length in the conclusion. The main increase in performance is on the train set which indicates the increased depth may be leading to some overfitting but ultimately the test set performance is also slightly improved.

4.3 2D

After deepening the 1D network, the decision was made to expand to a 2D version of both 1D networks. Initial impressions would hold that 1D networks would train faster, and since the features are not ordered along the features but only in time that a 2D network would only introduce hallucinated connections that do not exist. However, there's no good reason to not confirm this impression. In order to not needlessly beleaguer the point the exact details of the linear and convolutional layers will not be repeated here as they follow the earlier models so closely. The primary differences are that the convolutional layers now treat the initial 45x216 input as a single feature, outputting first 16 and then 32 such features before again flattening to 32x210 or 6720 features at the first linear layer.

The shallow 2D CNN returned 71.1% ±1.0% snippet accuracy and 82.1% ±1.7% song accuracy on the training set and 65.8% ±2.1% snippet accuracy and 81.1% ±1.9% song accuracy on the test set. This is, as predicted, a small decrease in performance compared to the 1D network. The interesting thing is that the main drop is on the training set while the test set performance is very comparable. Nonetheless the added weight is a distinct mark against the 2D approach.

4.4 Deep 2D

Finally, the same process applied to the shallow 1D network was applied to the deep 1D network to create the deep 2D CNN. This is by far the most neurons of any network used here and took by far the most training time. Subjected to the same scrutiny as the shallow 2D network, it is doubtful there is meaningful information between the extracted feature maps. As with the shallow 2D network, this architectural cover will discuss primarily the differences in the 2D layers and largely ignore the performance of the linear layers as they are still largely identical. Within the 2D layers, the first two layers are in fact identical to the start of the shallow 2D network, upscaling from 1 to 16 to 32 features while compressing the channels to 210, but now there are two additional 2D convolutional layers that further double the features to 64 and then 128 with 12 channels at the final step. This means that the first linear layer compresses only 1536 features to 600 before following the usual pattern.

The deep 2D CNN returned 76.8% ±1.2% snippet accuracy and 86.8% ±2.2% song accuracy on the training set and 67.3% ±3.0% snippet accuracy and 79.8% ±4.2% song accuracy on the test set. This matches expectations in a very peculiar way. The 2D networks did indeed perform worse than the 1D networks but while deepening the 1D network created the overall top performer deepening the 2D network created the worst. This relates to the idea that there is no actual spatial data along the second dimension of this data. Giving the 2D network greater depth gave it greater ability to hallucinate connections where none truly existed.

5 Final Model Performance

The 1D deep convolutional network is the best performer out of the tested models. This makes intuitive sense, given that the shallow 1D netowrk already performed very well and the data is most suited to a 1D convolution. Having confirmed that the 1D deep convolutional network has the best performance on the training and test sets, the final analysis is to test the model on prog adjacent music and provide a more detailed breakdown of performance on individual songs.

First, to reiterate the previous explanation, this architecture is a simple extension of the base model with two additional convolutional layers. The further caveat is that instead of maintaining the number of features between convolutional layers it instead doubles them. This provides a pipeline from 45x216 to 64x107, then 128x53, 256x25, finishing the convolution at 512x11. From there the standard assortment of four linear layers flattens this and reduces the feature count steadily until only two remain. At each convolutional layer a batch norm, ReLU activation, and max pooling occurs, and at each linear layer a ReLU is applied.

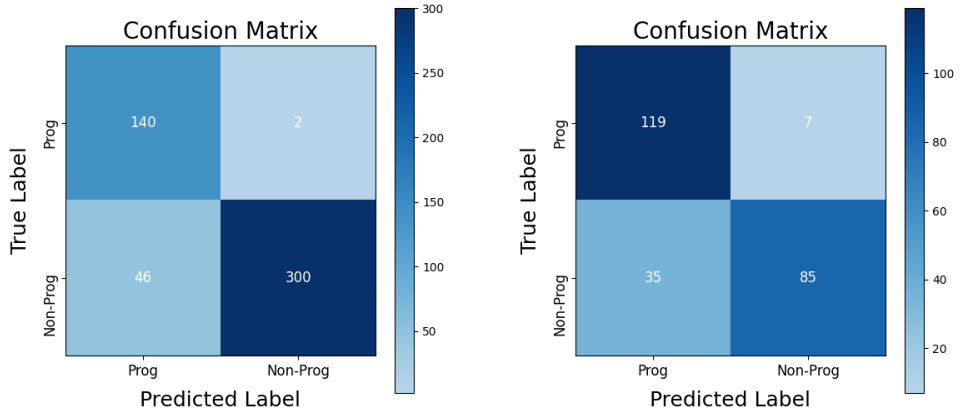

Figure 7: Repeat of the earlier confusion matrices. The 1D Deep performance on the training data (left) and the test data (right).

The detailed breakdown of performance on each song in the training and test set is attached

in a separate file to prevent a very long appendix. This section will still cover a selection of songs that proved difficult in both the training and test data sets. In the training set, the song "Bye Bye Love" by The Cars was deemed prog with 99.95% probability. This is false. That gap between certainty and accuracy is interesting. The song has many clear signs of being progressive rock which somewhat explains the difficulty. Bye Bye Love is rock, was recorded in the peak decades for prog rock, and has significant electronic instrumentation on the keyboard. Nonetheless, these signs are not sufficient and are merely very misleading. Another good example from the training set is "The Garden" by Rush. This song is labeled prog and the algorithm was very conflicted, reporting a 62% probability of not prog and 38% prog. This is usual given how many songs have near certain answers. The long periods of relative quiet and low activity might go some distance to explaining the distinct split in the snippet labels.

For the test data, the very first song "All Mine" is similar in character to "Bye Bye Love" in that the algorithm is very certain that it is prog at 99.9998% probability while the song is in fact not progressive rock. The song is 90s jazz fusion that I would describe as a lone singer being chased by an orchestra. It bears a strangeness and variety that are consistent with prog rock but it significantly lacks the 'rock.' Finally the song "Birthright" is definitively progressive rock and is mislabeled by the model with near certainty that it is not, reporting a full 100%. This result is perplexing. The opening is quiet and meandering, which could plausibly be read as not rock, but the steel guitar arrives eventually and it would be difficult to see those segments being mislabeled. The likely cause is the long stretches of relative quiet and simple vocalization.

5.1 Additional Test Data

As a bonus layer of difficulty, there is a second test data set that is intentionally *not* progressive rock but is very closely related to progressive rock. That is, songs from genres such as math rock and grindcore. These genres should be especially difficult to accurately discern and so they were excluded from the initial training and test sets. However, now at the final step, they can act as a more difficult challenge.

Name	Label
A Wolf Amongst Ravens	Not Prog
Soul Burn	Not Prog
DMA 4 Cascade	Not Prog
Let Yourself Be Huge	Prog
Zyglrox	Prog
Hollow	Prog
Arithmophobia	Not Prog
The Race Is About To Begin	Prog
The Haarp Machine Over	Prog
I, The Creater	Not Prog
Voyeur Will Shine, Fight for Disctinction, Evolution Is Mine	Not Prog
Hold My Finger	Not Prog
Language II Consider	Not Prog
DECEIVE AND DEFY	Not Prog
Divergency	Not Prog
Punisher	Prog
Bad Code	Prog
Laments of An Icarus	Prog
Splinter Cell	Prog
Isometry	Not Prog
Physical Education	Prog
A Light Will Shine	Not Prog

In brief, the performance on the prog adjacent set was significantly inferior to the normal test set, coming in with an accuracy of 54.5%. This is to be expected given the difficulty of the problem but shows that the model is barely better than random guessing at the borders of the genre. Given that this algorithm clearly has some knowledge of what prog is, this indicates that the prog adjacent problem is likely more difficult than the size of this dataset can support.

6 Conclusion

This project required the development of many skills, largely relating to data processing and audio in particular. Managing the data in the many forms required to properly create the various graphics was a challenge independent of creating an ideal architecture. This further highlights that in the end the highest performer architecture was a very small step away from the baseline. At some point a problem is solved and significant improvement becomes more and more difficult to find. There remain many small points of interest found during this project that are worth some additional thought. They are covered below.

The decision to randomize across songs may well have lost some valuable information given the nature of the data. That is, during training the segments are not processed in song order. Changing this to instead treat the snippets as subdivisions within a song and proceeding one to the next would allow for treating the data as a time series with dependence

on previous snippets. This opens the possibility for several techniques from the field of time series analysis.

The model that ultimately performed best is a deepening of the 1D network. This indicates that there is at least some complexity left in the problem that the shallow network wasn't able to fully exploit. The marginal nature of the improvement may indicate that the exact nature of the extension isn't a perfect match. Perhaps instead of adding more layers it would be useful to use a larger kernel size or a different approach to the pooling.

During the final stages of this project this team attempted to implement a CNN with more inventive additions such as LSTM layers or dropout. However time constraints proved too great a challenge to getting yet more models operational and properly evaluated. These ideas are left to future implementations.

Lastly, thank you for introducing us to all of these songs. Well, not all. We knew My Sharona.