Kyle Stoudt

A Comparative Analysis of Machine Learning Approaches to Music Genre Recognition

Problem Statement:

During a streamed Spotify event in February 2021, the company's Co-Head of Music, Jeremy Erlich, confirmed that over 60,000 tracks are being uploaded to Spotify every day. This marks a 50% increase in less than two years, when compared to an April 2019 quote of 40,000 daily uploads.[1] At this volume, it would take an incredible amount of human effort to document and categorize these songs. Machine learning provides the opportunity for us to automate this process by analyzing audio signal data to categorize songs into their respective parent genres on upload. Among many other benefits, this would give music streaming platforms the ability to see, in real-time, what genres are being uploaded.

The 'Genre' is an indispensable feature of a song, providing a tag to group it appropriately with similar songs which aids both listeners and other musicians. Genre represents a unique and colorful flag that suggests what an artist, album, or song may sound like in advance. Without this information as a signpost for those trying to find new music, it could become a tireless effort for listeners to sift through potentially millions of songs before finding what they want. The aim of this project is to provide a comparative analysis of different machine learning approaches which results in actionable recommendations for model usage and future refinement.

The Dataset:

The well-known GTZAN dataset is the most-used public dataset for machine learning applications in music genre recognition. It was collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.[2] The GTZAN dataset, downloaded from Kaggle, contains 1000 audio files, all having a length of 30 seconds. These files belong to a collection of 10 genres, with 100 samples per genre. Additionally, the dataset contains Mel-scaled Spectrograms (Mel Spectrograms) in PNG format which act as a visual representation of each audio file. These Mel Spectrograms allow us to evaluate the performance of Convolutional Neural Networks (CNNs), as they typically take data in some sort of image format.

The process of generating Mel Spectrograms from audio data with Librosa can be seen in the "Data Wrangling & EDA" notebook, along with embeddings to listen to a sample from each genre. The GTZAN dataset also contains CSV files containing tabularized data extracted via the Librosa library for each song sample. These were not used in this project, as the tabularized data was manually extracted with Librosa. See Figure 1.1 for an example of both the audio signal data and Mel Spectrogram visualization for a song classified as 'Rock'. Additionally, see Figure 1.2 for a view of GTZAN Mel Spectrograms from different genres. This is an example of what a CNN 'sees' when classifying songs by genre.

Fig. 1.1 – ‘rock.00005.WAV’ Audio Signal and Mel Spectrogram

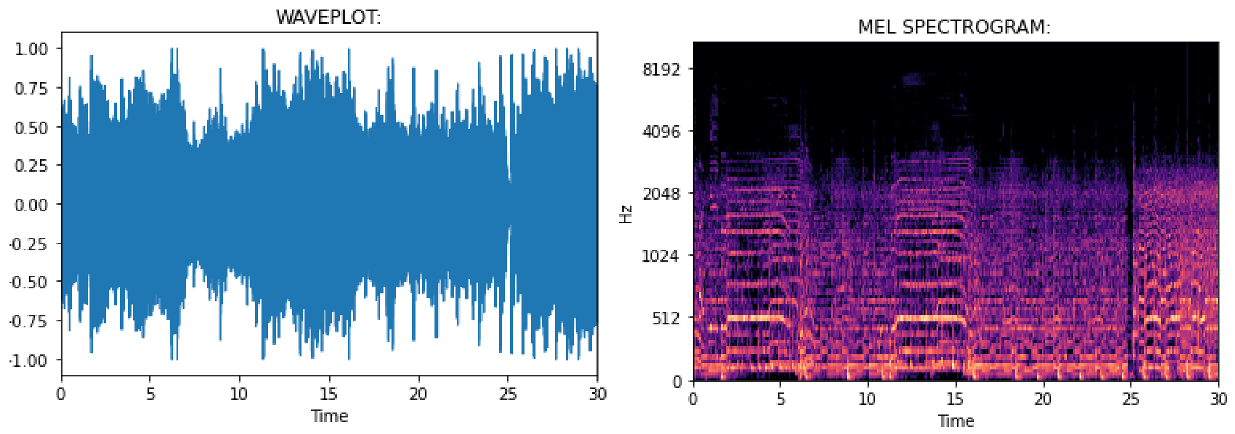
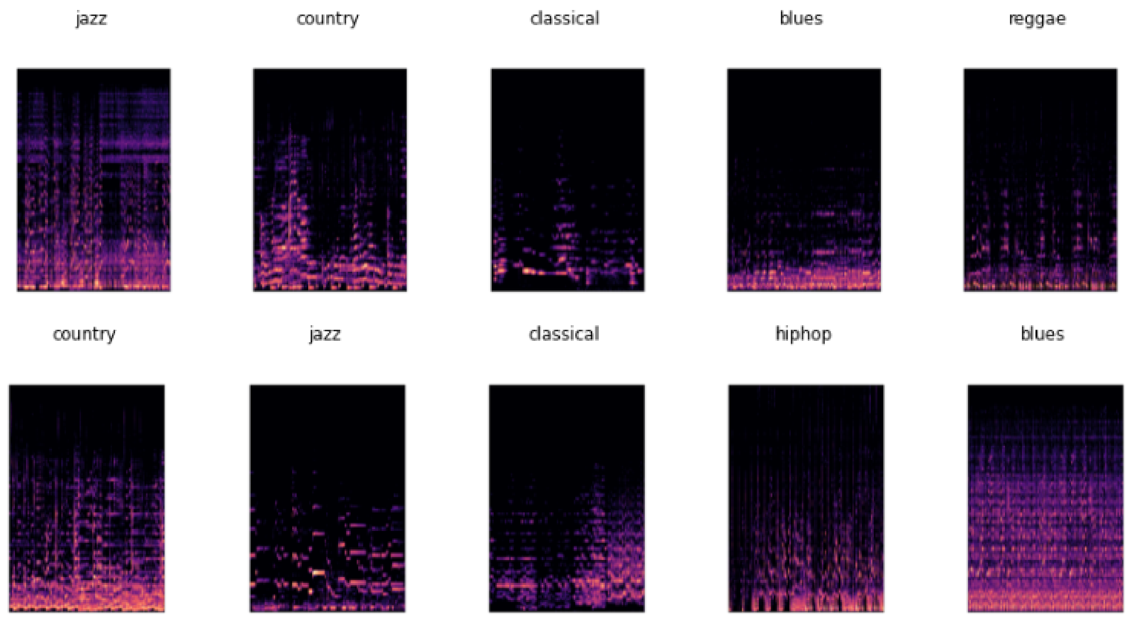


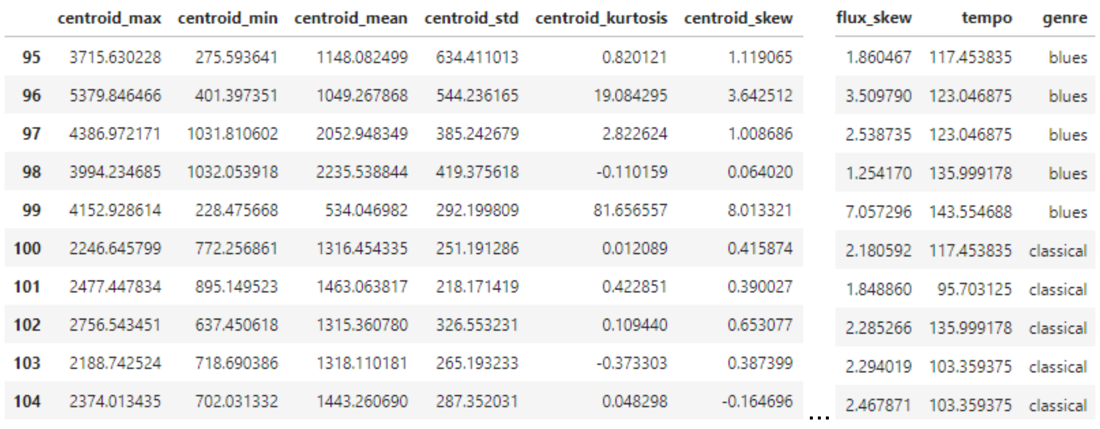
Fig. 1.2 – Sample of GTZAN Mel Spectrograms from Various Genres



Exploratory Data Analysis:

A wide selection of numerical features were generated for each song and tabularized into a DataFrame using Librosa and Pandas (Fig. 2.1) with the target variable of ‘genre’ appended.

Fig. 2.1 – Sample of Tabularized Data Generated with Librosa



	centroid_max	centroid_min	centroid_mean	centroid_std	centroid_kurtosis	centroid_skew	...	flux_skew	tempo	genre
95	3715.630228	275.593641	1148.082499	634.411013	0.820121	1.119065		1.860467	117.453835	blues
96	5379.846466	401.397351	1049.267868	544.236165	19.084295	3.642512		3.509790	123.046875	blues
97	4386.972171	1031.810602	2052.948349	385.242679	2.822624	1.008686		2.538735	123.046875	blues
98	3994.234685	1032.053918	2235.538844	419.375618	-0.110159	0.064020		1.254170	135.999178	blues
99	4152.928614	228.475668	534.046982	292.199809	81.656557	8.013321		7.057296	143.554688	blues
100	2246.645799	772.256861	1316.454335	251.191286	0.012089	0.415874		2.180592	117.453835	classical
101	2477.447834	895.149523	1463.063817	218.171419	0.422851	0.390027		1.848860	95.703125	classical
102	2756.543451	637.450618	1315.360780	326.553231	0.109440	0.653077		2.285266	135.999178	classical
103	2188.742524	718.690386	1318.110181	265.193233	-0.373303	0.387399		2.294019	103.359375	classical
104	2374.013435	702.031332	1443.260690	287.352031	0.048298	-0.164696	...	2.467871	103.359375	classical

Numerous comparisons and visualizations were made with the tabularized features, but none of them provided sufficient insight into the very high-dimensional dataset (48 dimensions, to be exact). One of the more useful visualizations was a boxplot of tempos grouped by genre (Fig 2.2), though it only informs on the distribution of tempos across genres. To provide a more comprehensive visualization, principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) were applied to sufficiently visualize the dataset in two dimensions (Fig. 2.3).

Fig. 2.2 – Boxplot of Tempos by Genre

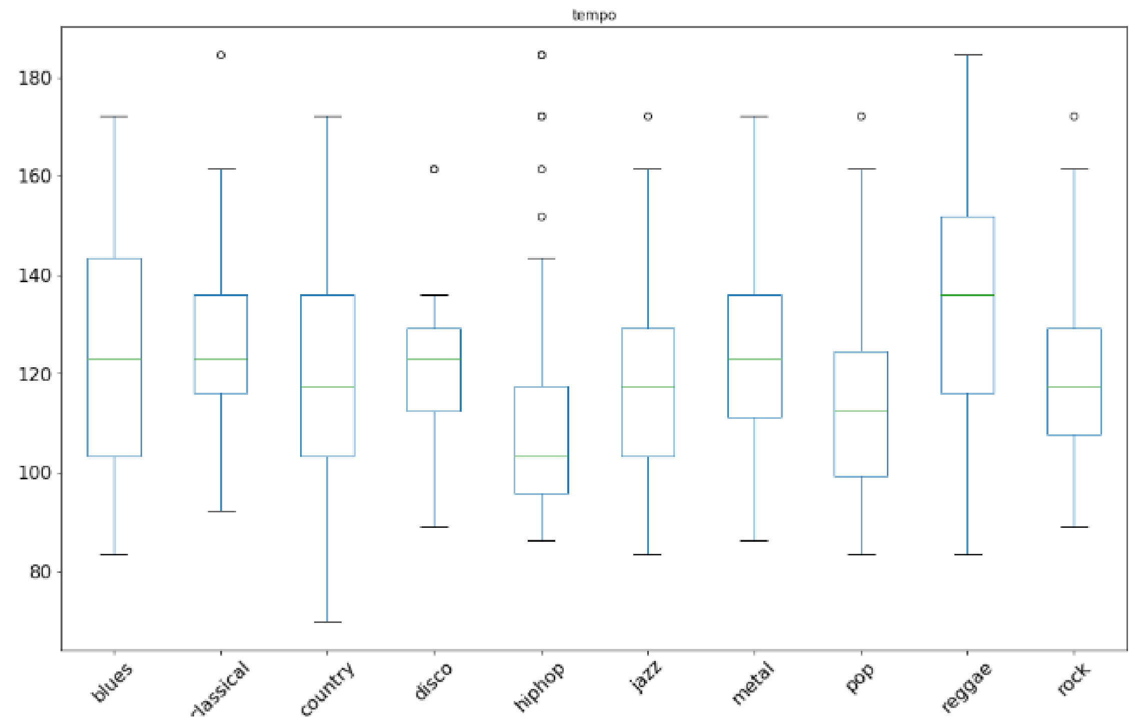
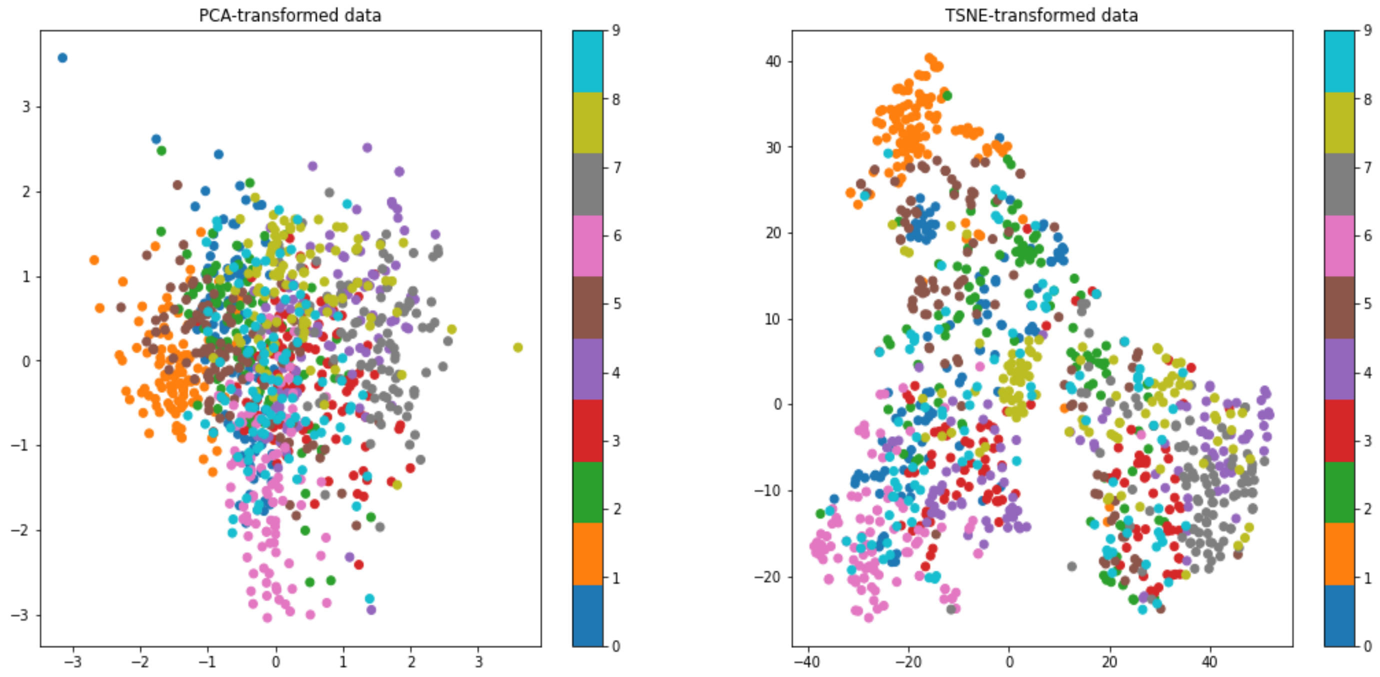


Fig. 2.3 – PCA and t-SNE on Tabular Data, Color-Coded by Genre

```
{'blues': 0, 'classical': 1, 'country': 2, 'disco': 3, 'hiphop': 4, 'jazz': 5, 'metal': 6, 'pop': 7, 'reggae': 8, 'rock': 9}
```



Classical ML Approach:

All numerical features were scaled with scikit-learn's StandardScaler() and an 80-20 train-test split was performed on the dataset. For each of the classical algorithms evaluated, GridSearchCV() was used to perform 5-fold cross validation for hyperparameter optimization. The tuned models were then evaluated on the hold-out set. The summarized metrics for these tested models can be seen in Figure 3.1 and are saved as 'Classical_ML_Metrics.csv'. The results show that the best-performer for the classical approaches was XGBoost's Classification algorithm. The 'Train AUC' of 1.0 indicates overfitting, though these models still perform relatively well on new data. This may simply indicate that the training data is representative of the test data in terms of genre-defining statistics.

Fig. 3.1 – Classical ML Metrics Summarized

	accuracy	cv best f1	test macro avg f1	Train AUC	Test AUC
KNearestNeighbors	0.615	0.603849	0.608323	0.965472	0.912458
LogisticRegression	0.700	0.685577	0.691909	0.979939	0.949750
SVM	0.705	0.710727	0.698152	0.999988	0.956889
XGBoost	0.725	0.686891	0.719013	1.000000	0.951889
GradientBoostingClassifier	0.700	0.687605	0.690294	1.000000	0.951528

Custom CNN Approach:

For the following image-recognition models, a 90-10 train-test split was performed. This is due to the fact that 1000 images is actually quite a small dataset for training a CNN. In the aim of optimizing model accuracy after convergence, the train split comprised as much data as possible without drastically reducing validation precision during training.

A custom CNN was built using TensorFlow which consisted of a rescaling layer, 5 convolutional blocks, and 5 L2-regularized dense layers which end with a 10-class decision layer utilizing the 'softmax' activation function. Compiled with the 'Adam' optimizer, the loss function calculated sparse categorical cross entropy and 'Accuracy' was chosen as the monitored metric. This should be appropriate, as our data is evenly distributed across all 10 genres. The custom CNN was fit to 85% of the training data and evaluated on the other 15%, with EarlyStopper and ReduceLROnPlateau for callback functions. This configuration resulted in convergence after 50 epochs and an accuracy of ~67% on the test data. This is a reasonable accuracy when considering the lack of available training data, and one could expect this number to rise simply with the addition of more training samples. See Figure 4.1 for a visualization of the training process.

Fig. 4.1 – Monitoring Accuracy and Loss during Custom CNN Training

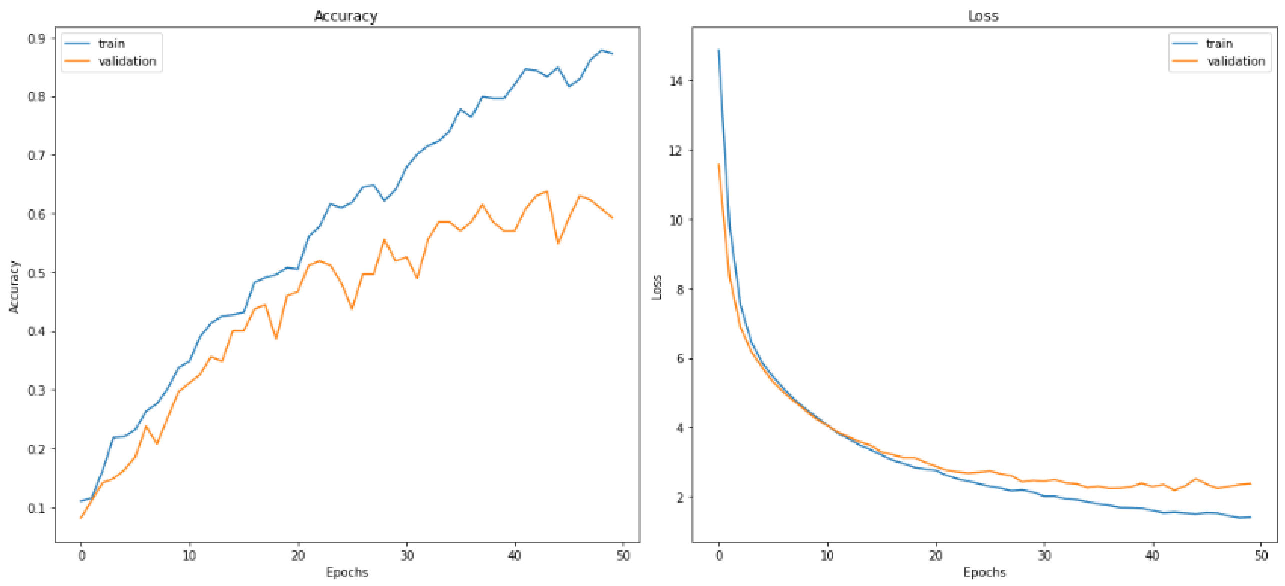
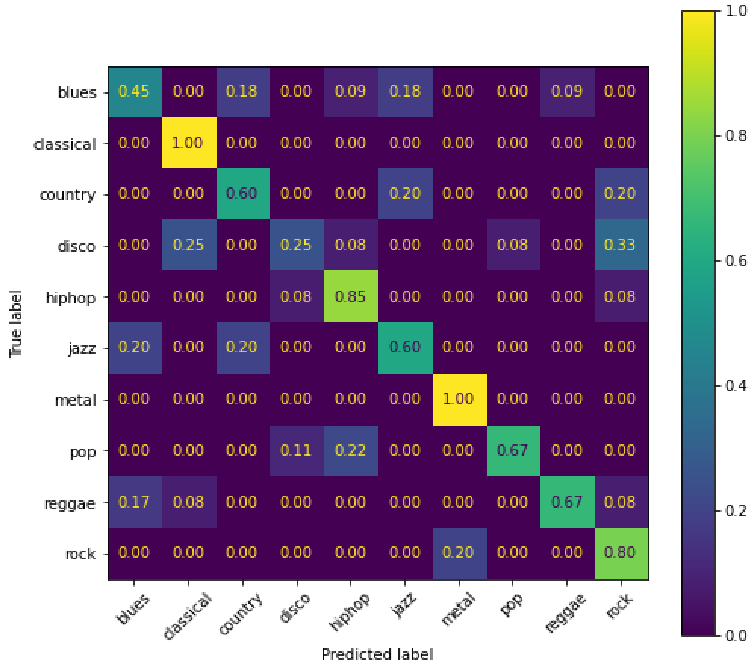


Figure 4.2 shows the normalized classification report for this CNN on the hold-out set. The results show our custom CNN is very good at classifying songs in the 'classical' and 'metal' genres, but seems to struggle with 'disco' and 'blues'.

Fig. 4.2 – Normalized Classification Report for Custom CNN



Transfer Learning Approach:

Three pre-trained networks were evaluated for transfer learning: ResNet50, MobileNetV2, and VGG16.

The ResNet50 architecture was frozen between an input layer and a global average pooling layer, with an added 10-class dense decision layer at the top. The training for the added layers was run for 80 epochs, appearing to converge with an accuracy of ~71% on the hold-out set. For fine-tuning, the ResNet50 layers were unfrozen, the model was re-compiled with a learning rate of 0.0001, and was trained for an additional 10 epochs. After fine-tuning, the accuracy dropped to ~70%. The observed volatility during fine-tuning is likely due to the small amount of available data. See Figure 5.1 for a visualization of the training process for the ResNet50 model. Figure 5.2 shows the ResNet50 model does very well with the genres of 'classical', 'rock', 'hiphop', 'jazz' and 'reggae', while struggling with 'pop', 'country', and 'disco'.

The MobileNetV2 architecture was frozen and followed by a global average pooling layer, a light dropout layer, and the 10-class dense decision layer. The same 80-epoch training and 10-epoch fine-tuning process was followed, yielding a final testing accuracy of ~70%. Figure 6.1 visualizes the training process for the MobileNetV2 model. Further, Figure 6.2 shows that the MobileNetV2 model is performing well with the 'classical', 'rock', 'hiphop' and 'jazz' genres, while struggling with 'pop', 'country', 'disco', and 'metal'.

Finally, the VGG16 'feature extractor' was frozen and followed by a flattening layer and a few dense, L2-regularization layers with one dropout layer. Once again, the output layer is a

10-class dense layer with the ‘softmax’ activation function. This time, however, the model was trained over 15 epochs before fine-tuning. This was done because the model was clearly converging very quickly, not requiring the same 80-epoch training process as the other transfer learning models. As with the other models, however, the fine-tuning process was carried out over 10 epochs. See Figure 7.1 for the visualization of the VGG16 model’s training process. The model achieved a testing accuracy of ~63% after fine-tuning. As with the other models, VGG16 performs well on some genres, and not-so-well on others. Figure 7.2 shows this model does well with the genres of ‘metal’, ‘hiphop’, ‘blues’, and ‘jazz’, while struggling with ‘pop’, ‘country’, and ‘classical’ songs.

Fig. 5.1 – ResNet50: Monitoring Accuracy and Loss During Training

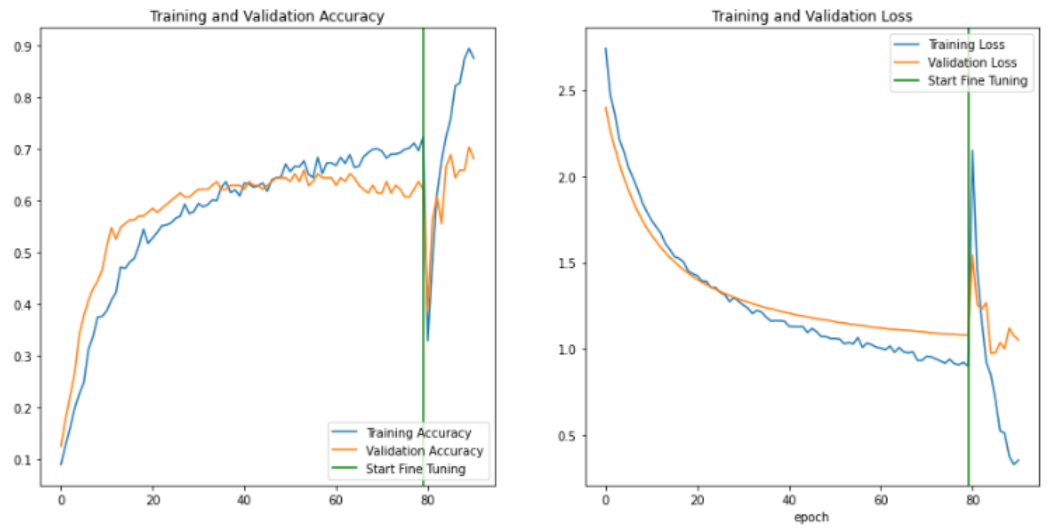


Fig. 5.2 – ResNet50: Normalized Classification Report

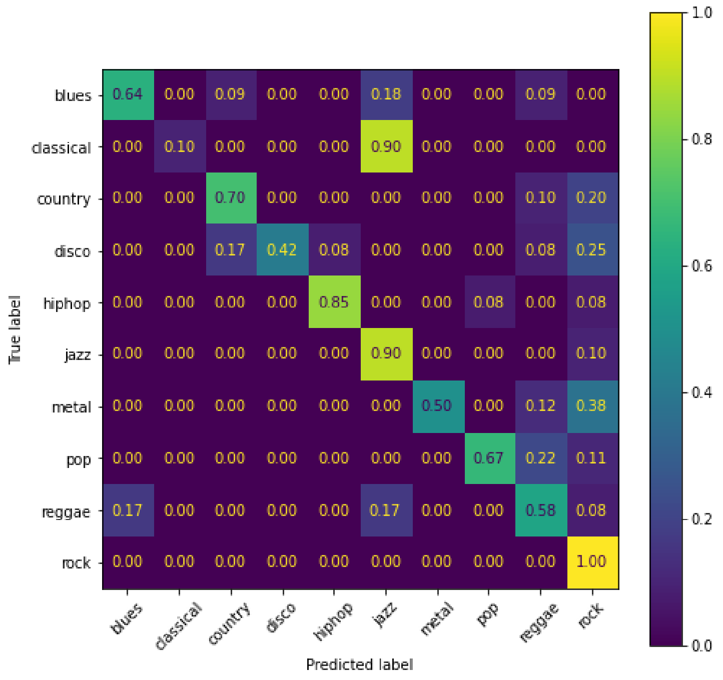


Fig. 6.1 – MobileNetV2: Monitoring Accuracy and Loss During Training

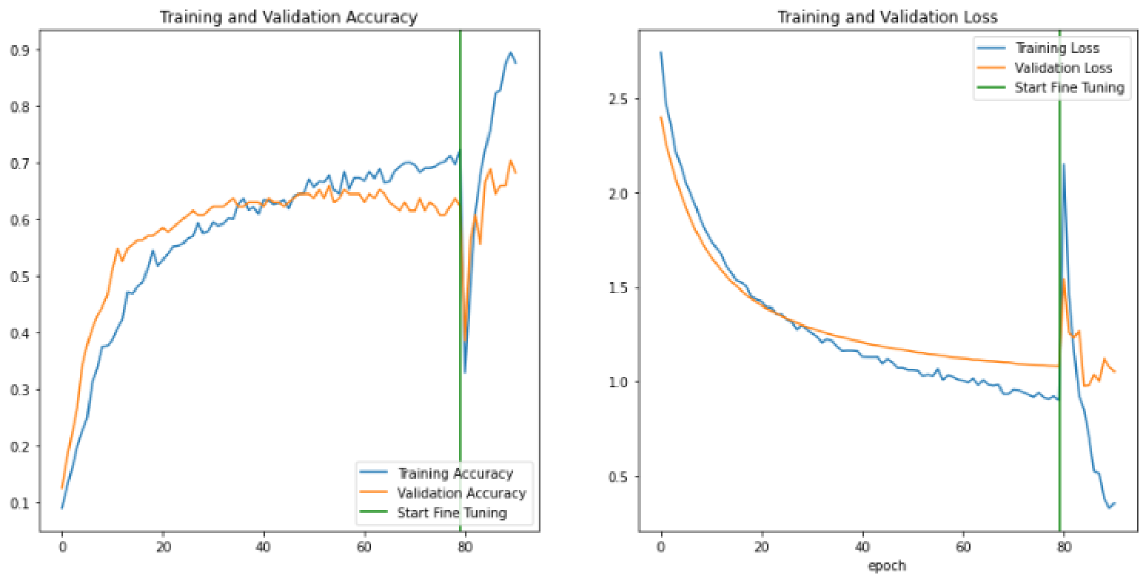
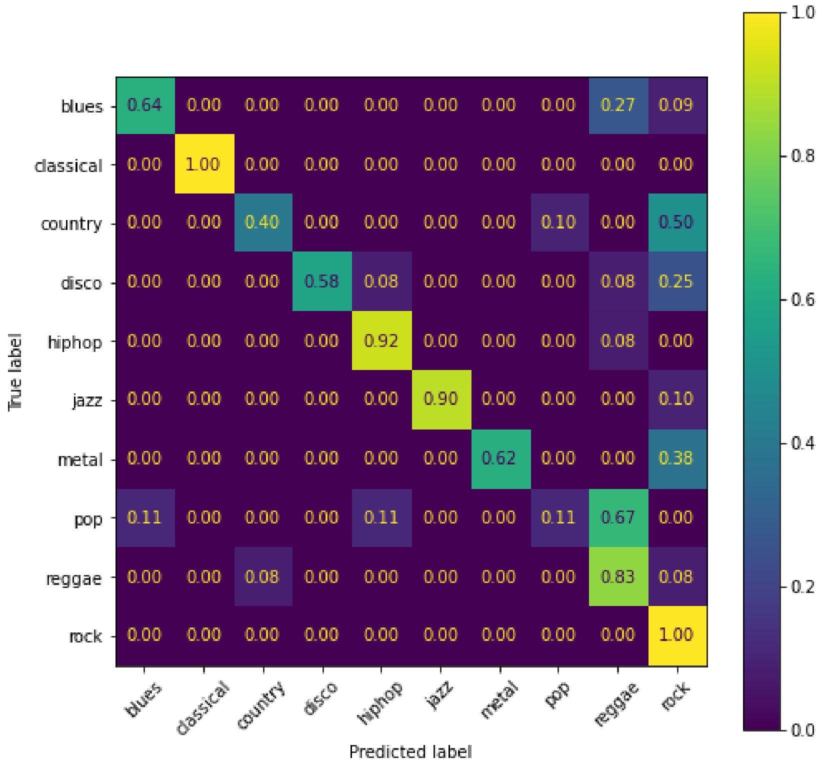


Fig. 6.2 – MobileNetV2: Normalized Classification Report



	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	0.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.09
classical	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
country	0.00	0.00	0.40	0.00	0.00	0.00	0.00	0.10	0.00	0.50
disco	0.00	0.00	0.00	0.58	0.08	0.00	0.00	0.00	0.08	0.25
hiphop	0.00	0.00	0.00	0.00	0.92	0.00	0.00	0.00	0.08	0.00
jazz	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.10
metal	0.00	0.00	0.00	0.00	0.00	0.00	0.62	0.00	0.00	0.38
pop	0.11	0.00	0.00	0.00	0.11	0.00	0.00	0.11	0.67	0.00
reggae	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.83	0.08
rock	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Fig. 7.1 – VGG16: Monitoring Accuracy and Loss During Training

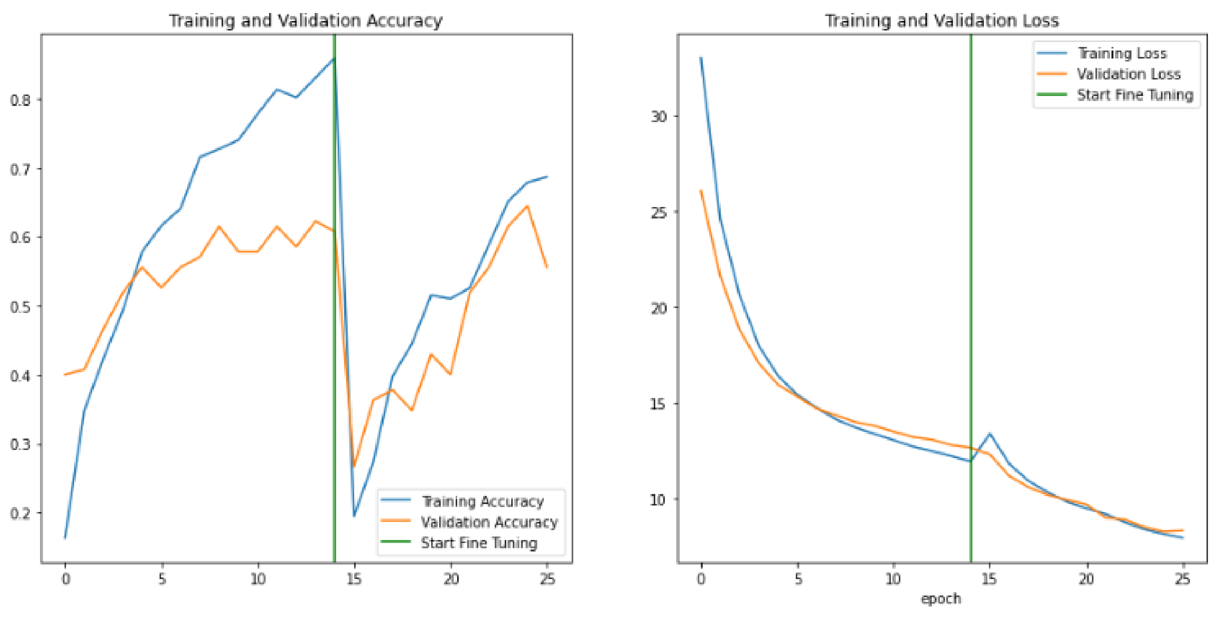
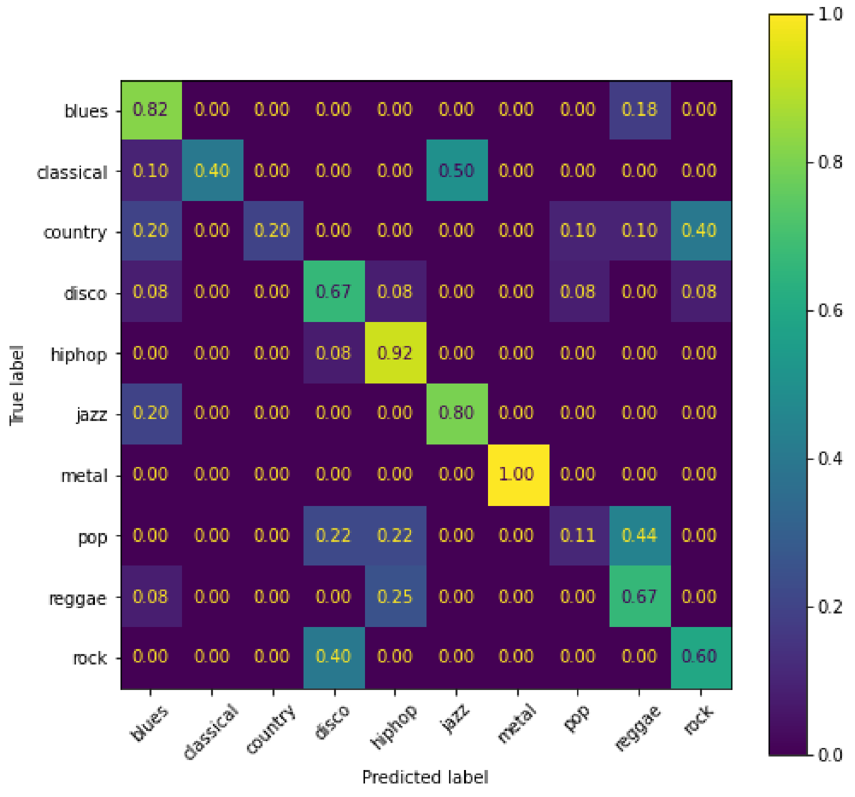


Fig. 7.2 – VGG16: Normalized Classification Report

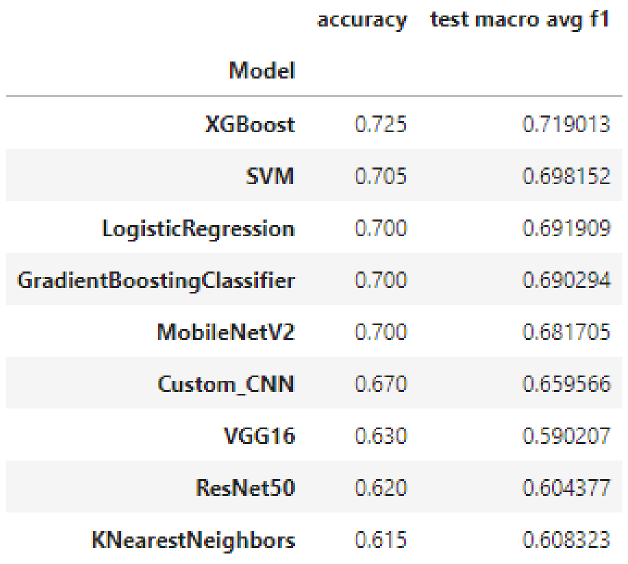


Conclusions & Recommendations:

The final testing results were compiled in a table and sorted by accuracy for easy comparison (Fig. 8.1). It should be noted here that the results seen in Figure 8.1 are approximately consistent with other runs of the training and testing stages for the deep learning models. For example, some runs of ResNet50's training achieved a testing accuracy of >72%, but in others it would yield ~62%, as seen in Figure 8.1. The lack of data is likely responsible for the volatility in results, especially in regard to the fine-tuning stages of the transfer learning models. With the addition of more data, one could expect these results to become much more stable.

On average, the deep learning models performed a little worse than the best classical models, with some transfer learning models occasionally testing in the 70-75% accuracy range. MobileNetV2 and ResNet50 were frequently jumping up in testing accuracy and would therefore be my main suggestions for models to train with more data. The custom CNN was performing more consistently than the others, due to the lack of a fine-tuning stage as with the transfer learning models. The accuracy of the custom CNN would therefore probably continue to improve steadily with the addition of more data and may prove to be a helpful addition to an ensemble approach, depending on results. The XGBoost and Support Vector Machine classifiers should be trained on more data as well, in order to serve as a baseline for comparison with deep learning models as they are developed. Finally, the deployed model(s) should be continually updated, training on new data when available.

Fig. 8.1 – Final Metrics Summary for All Models



	accuracy	test macro avg f1
Model		
XGBoost	0.725	0.719013
SVM	0.705	0.698152
LogisticRegression	0.700	0.691909
GradientBoostingClassifier	0.700	0.690294
MobileNetV2	0.700	0.681705
Custom_CNN	0.670	0.659566
VGG16	0.630	0.590207
ResNet50	0.620	0.604377
KNearestNeighbors	0.615	0.608323

Future Refinement:

Certainly the main restriction in performance for these models is the lack of available data. Particularly for the neural networks, 1000 samples is not enough data to train on to develop a comprehensive model which can sufficiently classify songs across 10 genres. In the 20+ years since the GTZAN dataset was collected, the amount of genre-labeled music available online has increased exponentially. If a music-streaming platform were to develop deep learning models for this purpose, they would have access to *millions* of songs, rather than 1000. So, in practice, the CNN or transfer learning approaches would probably prove most effective. Specifically, if a number of models are trained, each performing better than the others on certain genres, an ensemble method which classifies by a weighted majority vote across models would most likely be the optimal solution.

Aside from gathering more data, the project could be improved simply by splitting the data we do have into smaller samples which would be trained on, then classified by majority vote in prediction. One way of doing this would be to split the 30-second song clips into smaller, 10-second clips in preprocessing stages. This process would triple the amount of available samples to train and predict on, which would almost definitely result in more accurate predictions.

Sources:

[1] Tim Ingham, *"OVER 60,000 TRACKS ARE NOW UPLOADED TO SPOTIFY EVERY DAY. THAT'S NEARLY ONE PER SECOND.",* musicbusinessworldwide.com, Feb. 20, 2021*[https://www.musicbusinessworldwide.com/over-60000-tracks-are-now-uploaded-to-spotify-daily-thats-nearly-one-per-second/]*

[2] Andrada Olteanu, *"GTZAN Dataset - Music Genre Classification Audio Files | Mel Spectrograms | CSV with extracted features",* kaggle.com, *[https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification]*