Music Genre Classification with FMA: CMSC472 Project Report

Mark Jung
B.S. Computer Science
University of Maryland, College Park
markjung@umd.edu

Levi Lutz
B.S. Computer Science
University of Maryland, College Park
levi@umd.edu

Raymond James McGhee
B.S. Computer Science
University of Maryland, College Park
rjmcghee@umd.edu

Yashaswi Sharma
B.S. Computer Science
University of Maryland, College Park
ysharma2@terpmail.umd.edu

1 Introduction

Music Classification is a sub-problem of Computer Audition. Given the recent explosion of music data being stored on the web-based applications such as YouTube, Spotify, etc. the need for classification systems has grown exponentially as well. Genre tags for audio files can be used as keys for safer databases, recommendation systems, and playlist sorting.

Our group implemented a UNet encoder for this unique audio classification problem. We converted our audio files into spectrogram images which we then fed into our network to produce one-hot probability vectors where the probabilities represented the likelihood that this audio file belongs to a particular class. We had 8 classification genres: Electronic, Experimental, Folk, Hip-hop, Instrumental, International, Pop and Rock. Our final test classification accuracy came out to 42

2 Data

2.1 Free Music Archive [2]

The data were taken from the Free Music Archive (FMA) which is a high-quality, interactive library of audio tracks from WFMU - it was released in 2017. It provides associated metadata for each file, including genres, artists, dates, and more.

The full dataset includes 106,574 tracks across 161 unbalanced genres. However, due to memory constraints, we decided to use the `fma_small` dataset, which which consists of 8,000 audio tracks of 30 seconds across 8 balanced top-level genres. It is similar to the GTZAN dataset released in 2002, though more complete and better quality. The 8 top-level genres are: "Electronic", "Experimental", "Folk", "Hip-Hop", "Instrumental", "International", "Pop", "Rock".

3 Related Work

As mentioned in our project proposal, there have been a wide number of music classification Machine Learning systems made thus far. Defferrard et. al. provides us with several baseline models for genre recognition, such as logistic regression (LR), KNN, SVMs, etc. Defferrard et. al. achieved a maximum performance of 63% with a RBF-kernel SVM using the largest possible feature set.

There currently are no research papers (on Arxiv.org) that detail the usage of convolutional neural networks in music genre classification. However, the FMA authors hosted a 2018 classification

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

competition which spawned several papers which delved into the intricacies of audio classification using deep learning. A notable paper, published by Kim et. al., used transfer learning for audio classification. This method achieved a 63% classification accuracy. Another interesting paper released by Choi et. al. [7] explored the use of zero-shot learning, achieving results of about 73% test accuracy. Both papers were explorations in the potential usage of pre-existing ML infrastructure on audio data, specifically music data.

4 Our Approach

Our general approach is outlined as follows:

- Split each audio file into 3 Mel-spectrograms.
- Design a CNN to capture hierarchical features within the waveforms.
- Evaluate on training and validation set (provided by FMA).
- Evaluate best performing model on test set.

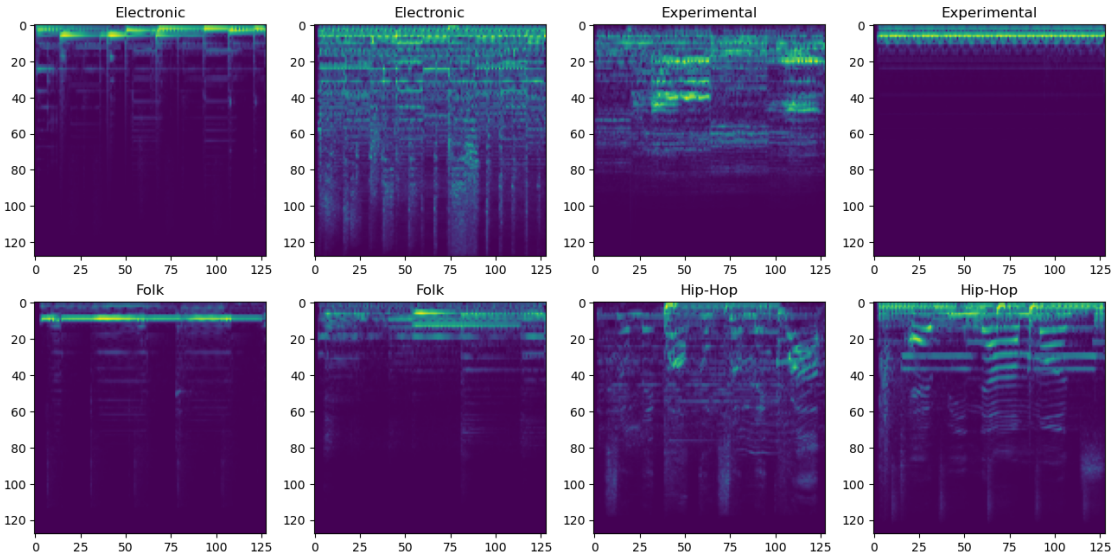
4.1 Data Preprocessing

Training, validation, and test splits for `fma_small` were provided by the researchers, which we used as well. They split the data into 80% training (6400 samples), 10% validation (800 samples), and 10% test.

Compressing 30 seconds of audio into a single spectrogram for learning is not very effective. So, we divided each audio file into three spectrograms using the `librosa` library [3] and parameters `sample_rate: 44100, window_length: 1024, hop_length: 512, n_mels: 128, time_steps: 127` to achieve an image size of 128x128. Each of these spectrograms was mapped to a corresponding one-hot vector of its genre classification.

While we had the thought of generating more spectrograms, which would potentially overlap, it turned out that this was infeasible for us computational-wise. There were some errors opening audio files in the training split. We skipped processing these results, leading to a final dataset with the following splits:

	Spectrograms	Labels
Training	(19184, 1, 128, 128)	(19184, 8)
Validation	(2400, 1, 128, 128)	(2400, 8)
Test	(2400, 1, 128, 128)	(2400, 8)



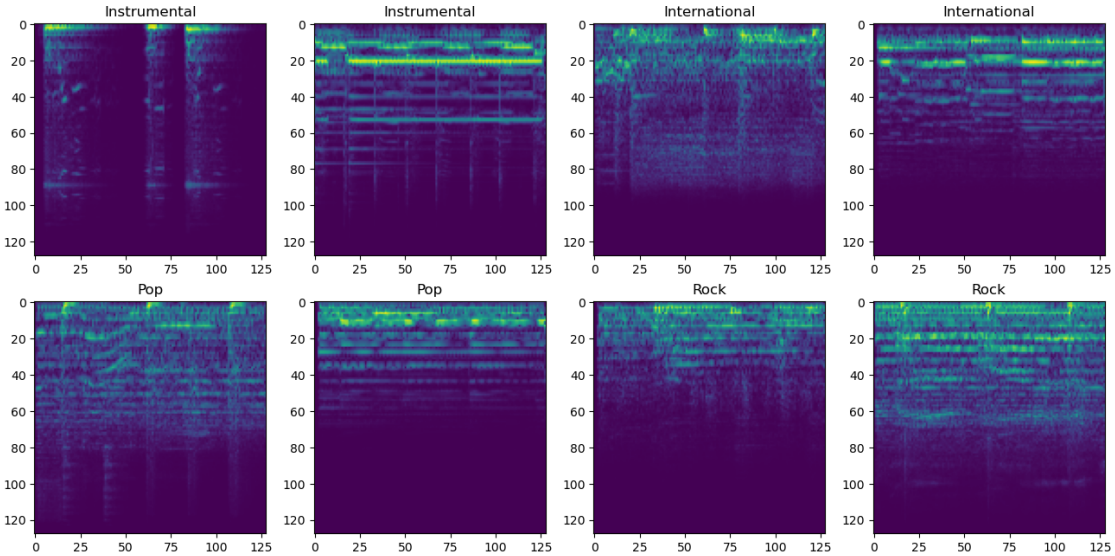


Figure 1. Selected spectrograms for each genre.

4.2 Architecture

A 2-D CNN is the most common type of CNN. We decided to design a 2-D CNN based on the encoder portion of the UNet architecture [4]. This includes four double convolutions, each including a 3x3 convolution, followed by batch normalization, and followed ReLU activation, repeated twice. After each double convolution, max pooling was performed to reduce the dimensions and dropout was applied to reduce overfitting. Finally, the last convolution was fed into a fully connected layer which output a length-8 vector of logits.

Additionally, weights were initialized using Xavier initialization and data were normalized by subtracting the training mean and dividing by training standard deviation.

4.3 Development and Training

Development was performed on Google AI Notebooks. Training was submitted as a Google AI Platform job using a single GPU to speed up computation. Due to a large dataset and GPU usage as well as 100% AI Notebook machine up-time, credit usage became an issue in the later stages of training.

5 Experiments and Results

5.1 Training and Validation

A first hyperparameter-tuning job was submitted to the Google AI Platform with the goal of minimizing validation loss. We allowed for 30 trials with early stopping. Some results are shown in the table below:

	Batch Size	Learning Rate	LR Scheduler	Validation Loss
1	39	0.0105	OneCycleLR	2.0842
2	40	0.0096	OneCycleLR	1.8806
3	30	0.0100	ReduceLROnPlateau	2.2423
4	64	1.0000	None	2.2228
5	41	0.0103	OneCycleLR	1.7441
6	43	0.0094	OneCycleLR	1.7051

Later, we realized that some models that did not train would receive lower validation loss than models that did train (because of overfitting). Because of this, our second hyperparameter-tuning job was submitted with the goal of maximizing validation accuracy. Additionally, we found that the OneCycleLR (CITE) performed the best, and batch sizes of around 42 were also the most successful,

so we used these in our second job, which only tuned for learning rate. The best results are shown below (out of 30 trials):

	Learning Rate	Validation Accuracy
1	0.00025	0.46333
2	0.00028	0.47000
3	0.00037	0.46416
4	0.00030	0.47875
5	0.00040	0.4888
6	0.00026	0.4892

5.1.1 Performance and Analysis

We took the two best performing models in Section 5.1.1 (batch size: 42, LR scheduler: OneCycleLR) and tested them, with the following results:

Model	Test Accuracy
5	0.4104
6	0.4271

To interpret our results, we plotted a confusion matrix on the test set (Fig. 2). Our best classifications were for the electronic (64.3%) and hip-hop (80.7%) genres. However, we performed poorly on experimental (23.3%) and pop (13.0%) music. Our performance on experimental music seems due to the definition (emphasis added): "a general label for any music that pushes existing boundaries and *genre* definitions" [5]. Our performance on pop music may be due to its varying definition, which is essentially all music that is popular. Thus, it may share many features with hip-hop and rock music, tripping up the model.

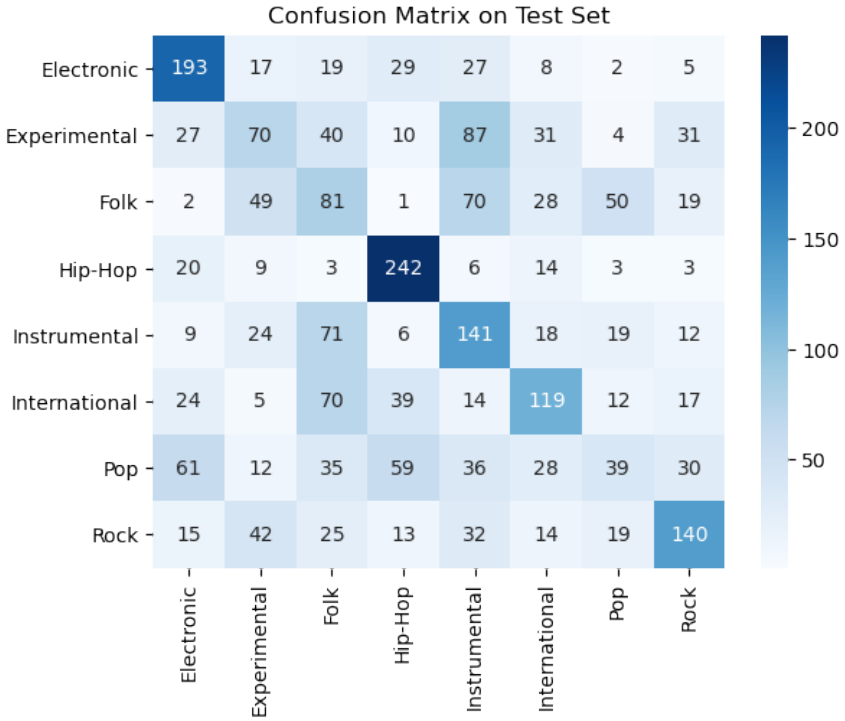


Fig 2. Confusion matrix of model predictions versus ground truth.

6 Conclusion and Discussion

Looking at our final performance with respect to the test set, we achieved a relatively high accuracy. Our model scored a 42% accuracy on the test set. This is higher than the baseline convolutional

neural networks referenced in the FMA data analysis notebooks, which achieved between 17% to 35%.

We ran into issues with our computational resources. Initially, we trained our model with 256x256 high-resolution spectrograms, which occupies a lot of memory. After much trial and error, we realised that we could reduce the resolution of our spectrogram and still get similar results which optimized computation time and available memory. This allowed us to reduce our training time from taking overnight (8+ hrs on GPU) to about 43 minutes, giving us room to use Google's Hypertune API [6] which improved our performance significantly.

We also ran into errors regarding our loss function for quite a few days. We misunderstood the PyTorch documenation and accidentally added a softmax activation at the end of our network while using Cross-Entropy Loss, which already computes softmax internally. This led to our model not training and a long debugging process.

7 Future Work

If we had 2 more weeks for this project, we would have tried to re-design our network as the UNet encoder implementation was not performing particularly well. We would try 1-D convolutions (such as 1x7 convolutions) because it is popular in this space. We would also enhance our features (via augmenting the spectrograms by speeding up/slowing down, varying pitches, and shifting the audio clips). This would increase the size of our data as well as improve the generalizing power of our model.

If we had 2 more months for this project, we would piggy back to changing our model by using an RNN instead as that would allow us to better capture the time domain. We would then compare the RNN's performance as opposed to our enhanced CNN's performance to analyze the underlying data better.

References

[1] Kim, J., Won, M. Serra, X., &; Liem, C. C. (2018). Transfer learning of artist group factors to musical genre classification. Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18. doi:10.1145/3184558.31918234

[2] Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2017). FMA: A Dataset For Music Analysis. doi:1612.01840

[3] Librosa. (n.d.). https://librosa.org/.

[4] Ronneberger, O., Fischer, P., &; Brox, T. (2015, May 18). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv.org. https://arxiv.org/abs/1505.04597.

[5] Wikimedia Foundation. (2021, May 14). Experimental music. Wikipedia. https://en.wikipedia.org/wiki/Experimental_music.

[6] cloudml-hypertune. PyPI. (n.d.). https://pypi.org/project/cloudml-hypertune/.

[7] Choi, J., Lee, J., Park, J., &; Nam, J. (2020, March 19). Zero-shot Learning for Audio-based Music Classification and Tagging. arXiv.org. https://arxiv.org/abs/1907.02670.