GCT634: Musical Applications of Machine Learning **Name:** *Vanessa Tan - 20225640*
HW2: Music Auto Tagging (Multi-Label Classification)
October 16, 2022

1 Introduction

With the recent growth of using applications like Spotify and Apple Music, Music Information Retrieval (MIR) researchers should have a basic foundation on how these systems work. One of the tasks developed for these systems is automatic music tagging. This task is a multi-label classification which predicts different tags for a given song [1]. This document discusses the improvements done in the automatic music tagging baseline.

2 Data Analysis

A song can be composed of different instruments (e.g. piano, guitar, violin, etc.) or sections (intro, chorus, bridge). A section may not contain all the instruments in the given song, hence, it is hard to create a dataset in which all instruments or tags have the same distribution. As an example the MagnaTagATune (MTAT) dataset which is used in this experiment is highly imbalanced as shown in fig. 1 [1, 2]. The guitar tag appears most of the time in the dataset while the choral tag appears the least. One of the techniques for handling imbalanced datasets is data augmentation where different audio effects are applied to the dataset to compensate for the audio with least samples. Improvements of the architecture will be further discussed in the next sections.

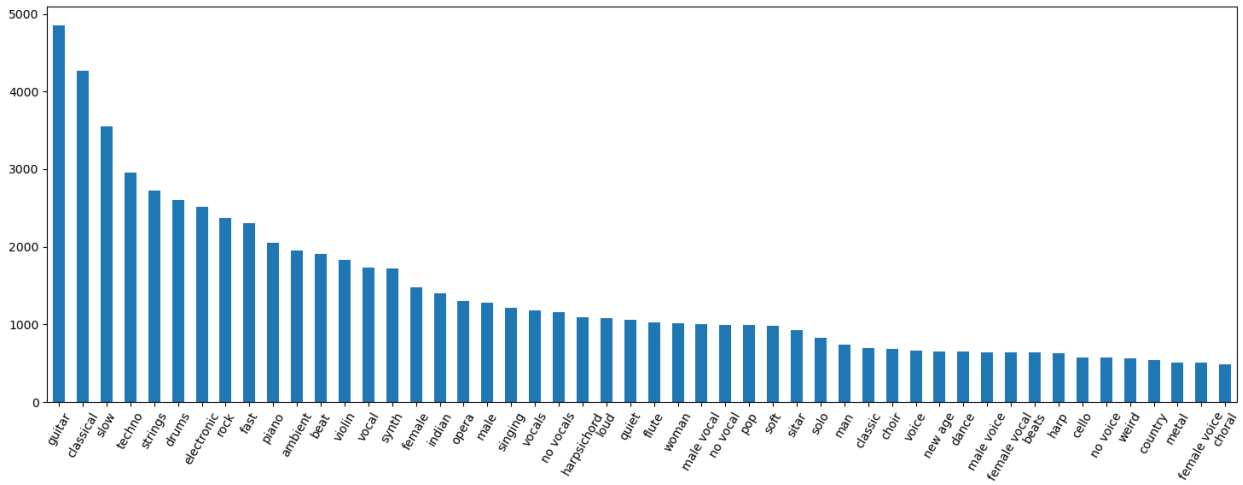

Figure 1: Dataset Distribution

3 Methodology

The baseline architecture is given by table 1. It is a simple 2D convolutional architecture with different pooling parameters. The metrics used to evaluate this architecture is the ROC-AUC and PR-AUC. For this baseline architecture, the ROC-AUC and PR-AUC are 0.7217 and 0.2036 respectively. Different parameters and experiments were done to the baseline. However, only the improvements will be discussed in these sections.

Table 1: New Architecture

Layer	Output Size	Details
input	B x 1 x sample rate * duration	batch x channel x samples
mel spec	B x 1 x 96 x 188	batch x channel x freq x time
conv + maxpool	B x 64 x 24 x 47	out=64, kernel=3, pooling=(4,4)
conv + maxpool	B x 128 x 8 x 15	out=128, kernel=3, pooling=(3,3)
conv + maxpool	B x 128 x 2 x 5	out=128, kernel=3, pooling=(3,3)
conv + maxpool	B x 64 x 1 x 1	out=64, kernel=3, pooling=(2,5)
classifier	B x 50	-

3.1 Optimizers

Optimization algorithms reduce the model error by calculating and modifying the parameter values[3]. Two popular optimization algorithms for deep learning are Stochastic Gradient Descent (SGD) and Adam [4, 5]. These algorithms make the model generalize better, train faster, and achieve optimal minima. Another optimization algorithm is NAdam which incorporates SGD's momentum concept into Adam [6]. The evaluation of these optimizers are shown in table 2. Results show that NAdam improved the baseline, hence, showing that optimization algorithms could greatly affect the model performance.

Table 2: Optimizer Results

Optimizer	ROC-AUC	PR-AUC
SGD (Baseline)	0.7217	0.2035
Adam	0.8485	0.3265
NAdam	**0.8516**	**0.3281**

3.2 Loss Function

Loss functions also play an important role in model performance. The most common loss function for classification in binary cross-entropy. Another variation of cross-entropy is focal loss. Focal

loss was used in dense object detection to solve class imbalances [7]. It uses the training weights to penalize easy samples and focus training on the hard samples. The evaluation for this loss function can be seen in the table below. The focal loss improves the model performances and tries to balance the accuracy for each class as shown in fig. 2 and 3.

Table 3: Loss Function Results

Loss Function	ROC-AUC	PR-AUC
Binary Cross-Entropy (Baseline)	0.8516	0.3281
Focal Loss	**0.8576**	**0.3372**

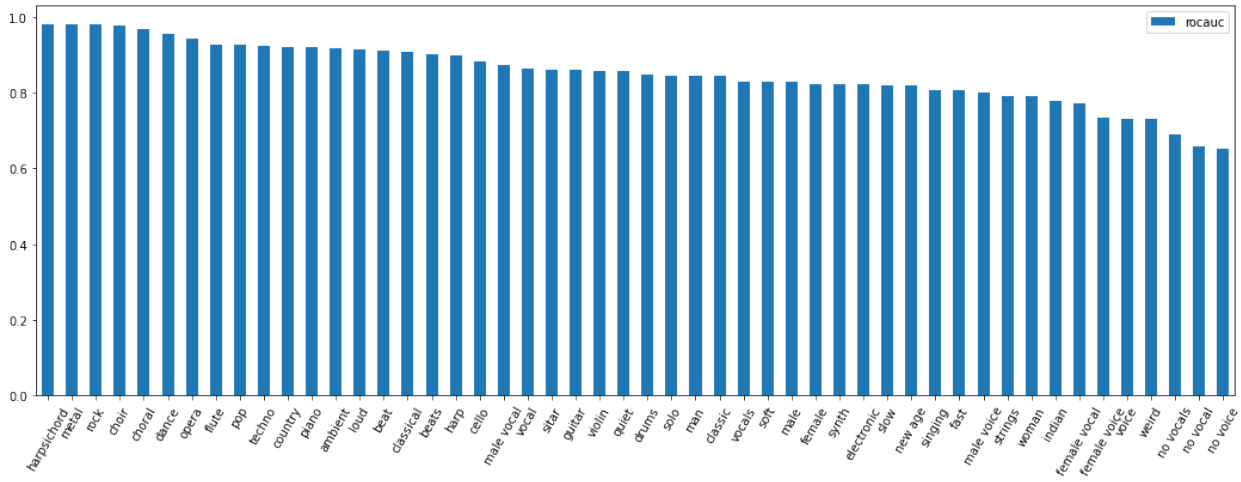

Figure 2: ROC-AUC Results: Architecture with Binary Cross-Entropy

3.3 Activation Functions

Another parameter explored in this experiment is the activation functions. The most common activation function is the Rectified Linear Unit. Other non-linear activation functions like the Exponential Linear Unit (ELU) and Gaussian Error Linear Units (GELUs) are explored in this experiment. As shown in table 4, GELUs improved the model performance.

Table 4: Activation Results

Activation	ROC-AUC	PR-AUC
ReLU (Baseline)	0.8576	0.3372
ELU	0.8605	0.3421
GELU	**0.8635**	**0.3474**

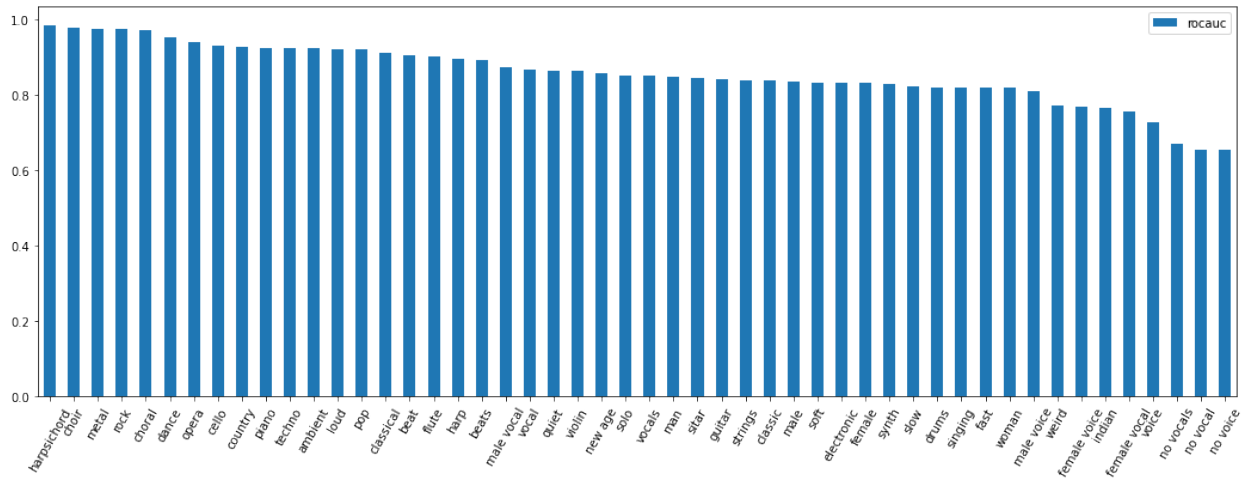

Figure 3: ROC-AUC Results: Architecture with Focal Loss

3.4 New Model Architecture

Inspired by Short-chunk CNN, a new model architecture was implemented as shown in the table below. Similar to Short-chunk CNN, simple 2D CNN with 3x3 filters and 2x2 max pooling layers were used [8]. However, instead of a 7-layer CNN, only a 6-layer CNN and 96 mel bins were used in the experiment. A dropout layer is also added after conv + maxpool layers to avoid overfitting.

Table 5: New Architecture

Layer	Output Size	Details
input	B x 1 x sample rate * duration	batch x channel x samples
mel spec	B x 1 x 96 x 188	batch x channel x freq x time
conv + maxpool	B x 128 x 48 x 94	out=128, kernel=3, pooling=(2,2)
conv + maxpool	B x 256 x 24 x 47	out=256, kernel=3, pooling=(2,2)
conv + maxpool	B x 256 x 12 x 23	out=256, kernel=3, pooling=(2,2)
conv + maxpool	B x 128 x 6 x 11	out=256, kernel=3, pooling=(2,2)
conv + maxpool	B x 128 x 3 x 5	out=128, kernel=3, pooling=(2,2)
conv + maxpool	B x 128 x 1 x 1	out=128, kernel=3, pooling=(2,3)
classifier	B x 50	-

As shown in table 7, more layers and increasing the number of filters improved the model performance. Additionally, similar with Short-chunk CNN's claim, a simple 2D CNN with small kernel and pooling size can achieve good performance [8].

3.4.1 Augmentation

Another technique to resolve imbalanced datasets is to use data augmentation. Data augmentation is the process of synthesizing data to increase the amount of samples in the dataset [9].

Table 6: New Model Results

Model	ROC-AUC	PR-AUC
Previous Model	0.8635	0.3474
New Model	**0.8788**	**0.3785**

This experiment uses the torchaudio augmentation which offers the following audio transforms: frequency filter, delay, pitch shift, reverb, gain, noise, and polarity inversion [10]. These transformations were experimented but only the polarity inversion, gain, and pitch shift helped in improving the model. The polarity inversion helped in removing unwanted noise in the audio while gain increases the volume of the signal [9]. An important transformation is the pitch shift since the samples for the vocals particularly the female vocals are very little. As shown in fig. 5, augmentation helped in increasing the accuracy of the classes particularly for the vocals. However, this model has a hard time with tags with no vocals. A timbre masking audio augmentation where the separation of vocals and instruments could be done. Another technique is to apply voice activity detection augmentation which could help in increasing the performance for vocal and non-vocal tags.

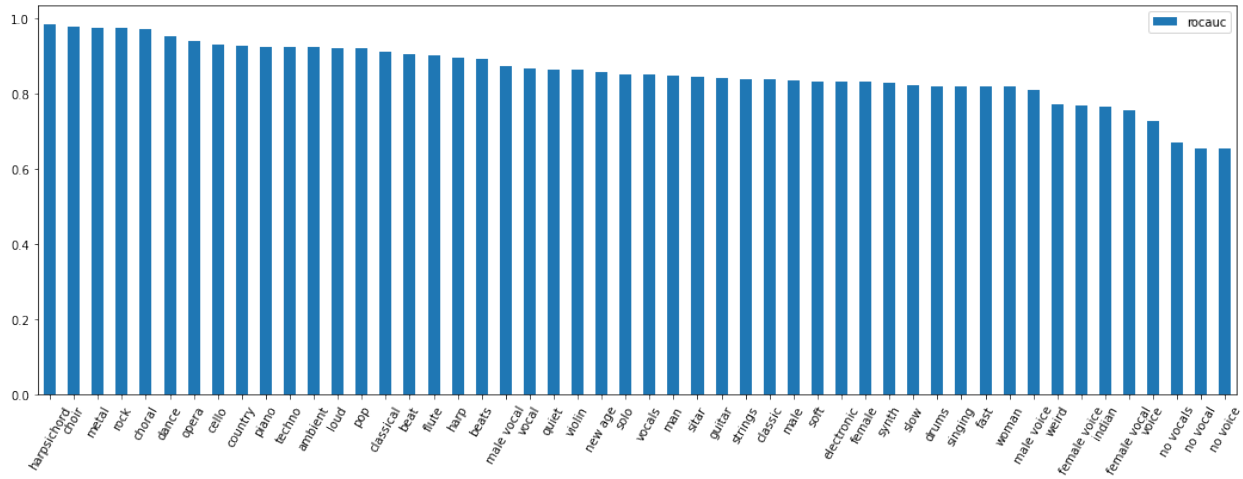

Figure 4: ROC-AUC Results: No Augmentation

To summarize, the final model uses augmentation and the parameters (optimizer, loss function, and activation functions) discussed in the previous sections. Also, the following hyperparameters were utilized: dropout = 0.1, learning rate = 0.001, batch size = 16 running for 50 epochs with early stopping.

4 Conclusion

This experiment demonstrates automatic music tagging using deep learning. One of the problems for automatic music tagging is having imbalanced datasets. Solutions to help solve imbalanced

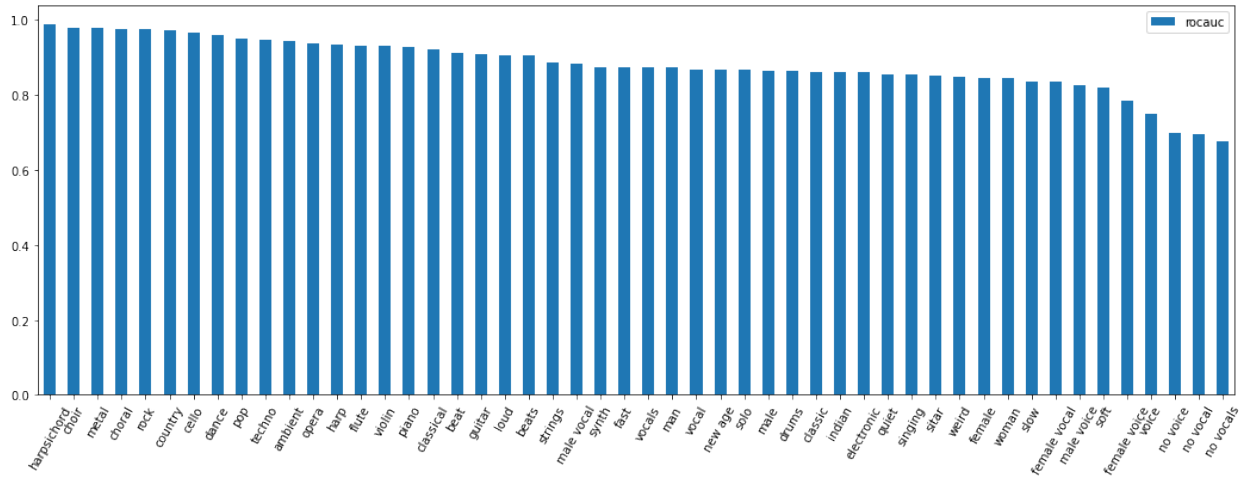

Figure 5: ROC-AUC Results: With Augmentation

Table 7: Augmentaion Results

Data	ROC-AUC	PR-AUC
No Augmentation	0.8788	0.3785
With Augmentation	**0.8815**	**0.3860**

datasets are applying data augmentation or modifying loss functions. The different model parameters play a big role in the model performance, hence, different experiments are needed. In this experiment, the NAdam optimizer, focal loss function, GELU activation, increasing the number of layers and filters, and data augmentation helped improve the model performance.

References

[1] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," 2020.

[2] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *ISMIR*, 2009.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[4] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," 2015.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[6] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

[8] M. Won, S. Chun, O. Nieto, and X. Serrc, "Data-driven harmonic filters for audio representation learning," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 536–540, 2020.

[9] M. Won, J. Spijkervet, and K. Choi, *Music Classification: Beyond Supervised Learning, Towards Real-world Applications*. https://music-classification.github.io/tutorial, 2021.

[10] J. Spijkervet, "Spijkervet/torchaudio-augmentations," 2021.