Comparative Study of Multilayer Perceptron Models for Music Genre Classification

Your Name
Student Number: 12345678

Abstract

This report explores several deep learning methods for music genre classification applied to the GTZAN dataset [1]. We implement and compare a baseline MLP (multilayer perceptron) with handcrafted audio features, a tuned MLP (using Optuna for hyperparameter optimization), a deeper MLP (adding layers and regularization), and an autoencoder-based model that learns latent feature representations [4]. We present the architecture, training setup, and results from each of the models in detail. In experimentation on the GTZAN dataset, the baseline MLP achieved approximately 90.99% accuracy, the tuned MLP had approximately 92.3% accuracy, and the deep MLP had approximately 92.5% accuracy. The autoencoder approach had approximately 87.0% accuracy. We comment on these results concerning model complexity, accuracy, and generalization performance.

1 Introduction

Music genre classification is a core aspect of music information retrieval (MIR) which forms the basis of applications from personalized music recommendation to large scale organization and archiving of digital audio collections. Traditional techniques often rely on hand- crafted audio descriptors (i.e., mel-frequency cepstral coefficients, chroma features, spectral contrast, tonnetz features) and classical machine learning classifiers (i.e., k-nearest neighbors, support vector machines, or Gaussian mixture models). While these methods worked well in previous work, the reliance on engineered features may limit their ability to map the complex and nonlinear relationships present in audio signals. Recently, deep learning has made a huge impact on MIR by allowing for a fully end-to-end approach to learn hierarchical feature representations from raw, or minimally processed data and achieving state-of-the-art results in genre classification, instrument recognition, and audio tagging [3]. Fully-connected neural networks (i.e., multilayer perceptrons [mlps]) provide a simple but flexible architecture for supervised learning with fixed-length feature vectors. In this project, we will focus on mlp-based approaches using handcrafted feature inputs. We started with a basic baseline MLP, enhanced it through systematic hyperparameter optimization (tuned MLP), deepened its architecture via the use of added layers and regularization (deep MLP), and also considered using an autoencoder + MLP pipeline for unsupervised pretraining followed by latent-space classification. This series of developments serves to allow us to observe how model complexity, as

well as representation learning, impact classification accuracy and generalization performance in the GTZAN benchmark.

2 Related Work

The GTZAN dataset [1, 2] was used, containing genres of blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock (100 tracks per genre). The data were divided into training, validation, and test sets (approximately 70%/15%/15%) with stratified sampling to ensure genre distributions remained unchanged. From each audio track, standard audio features (e.g., MFCCs, spectral contrast) were extracted to create a fixed-length feature vector. Features were normalized to have zero mean and unit variance (z-score) prior to training.

3 Experimental Setup

3.1 Dataset and Preprocessing

The dataset used for this study was in the form of the standard GTZAN dataset [1, 2], containing genres of blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock (100 tracks each genre). The data was split into training, validation, and test sets (roughly 70%-15-% 15%) with stratified sampling to maintain the genre distributions. For each audio track, a fixed-length feature vector can be obtained by extracting the standard audio features (e.g., MFCCs, spectral contrast). The features were normalized to achieve mean 0 and variance 1 (z-score) prior to training.

3.2 Baseline MLP

The baseline MLP model is a simple two–hidden–layer MLP., The network consists of an input layer followed by two hidden layers of 256 and 128 units, respectively. Each hidden layer applies batch normalization [7], a ReLU activation, and dropout (rate = 0.3) for regularization [6]. The final output layer has 10labels denoting to one per genre with a softmax activation being used.

The Adam optimizer was used for training across 100 epochs with a batch size of 128 and a learning rate of 0.001 and cross-entropy loss. On the GTZAN test set, the baseline MLP's accuracy was roughly 90.99%.

3.3 MLP with Hyperparameter Tuning

To improve the baseline model, Optuna [5] was used for hyperparameter optimization. The search space includes the two hidden layer sizes, dropout rate, learning rate, and batch size. Hidden layer sizes were chosen in ranges 128–1024 and 64–512 (step 64), dropout in [0.1,0.5], learning rate in $[10^{-4}, 10^{-2}]$, and batch size in {64, 128, 256}. The validation accuracy was optimized over 30 trials. The best trial had hidden layer sizes 512 and 448, dropout ≈ 0.277, learning rate ≈ 0.00493, and batch size 256.Training was then performed on the tuned MLP with these parameters for 100 epochs. In our experiments, this model achieved about 92.33% test accuracy, a 1.3% improvement over the baseline.

3.4 Deep MLP

A deeper MLP was developed to investigate the impact of added layers and regularization. The deep model contains five hidden layers with 1024, 512, 256, 128, and 64 units, respectively (Figure 1). Each hidden layer is followed by batch normalization [7], a ReLU activation, and dropout (with dropout rates decreasing from 0.3 to 0.2) [6]. The network was trained in 100 epochs with the AdamW optimizer (learning rate = 0.0008 and weight decay = 104) using a batch size of 128. The deeper MLP was able to achieve approximately 92.53% test accuracy, slightly better than the tuned MLP, which suggests that the benefit of additional depth was small for this task.

3.5 Hyperparameter Tuning

Optuna [5] was used for hyperparameter optimization to improve the baseline MLP.The sizes of the two hidden layers, dropout rate, learning rate, and batch size were tuned. All hidden layers retained batch normalization [7] and dropout. Hidden layer sizes were sampled from the ranges 128–1024 and 64–512 (in steps of 64), dropout was sampled from the interval 0.1–0.5, learning rate was sampled from 0.0001–0.01, and batch size was chosen from {64, 128, 256}.The validation accuracy has been fine tuned across 30 trials. The optimal trial hidden layer sizes were 512 and 448 with a dropout rate of around 0.277, which is a learning rate of around 0.00493, and a batch size of 256. The next step was to take the tuned MLP using these hyperparameters to train for 100 epochs. The resulting tuned model had a test accuracy of around 92.33%, which is 1.3 % greater than the untuned baseline.

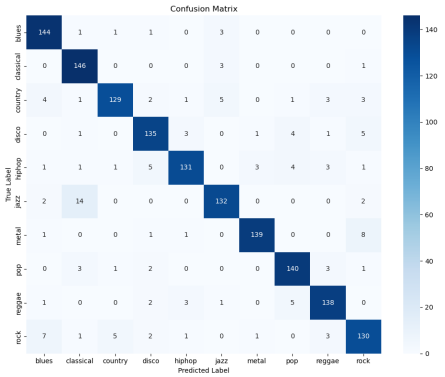
The trained deep MLP achieved approximately 92.53% accuracy on the test set upon training.

3.6 Autoencoder-based Model

A fully-connected autoencoder was constructed: the encoder mapped the original 58-dimensional feature vector to a 128-dimensional bottleneck via an intermediate layer of 256 units, and the decoder mirrored this structure to reconstruct the input. The autoencoder was trained in an unsupervised manner using mean squared error loss with the Adam optimizer (learning rate = 0.001) for 100 epochs and a batch size of 128. After training, all data were encoded to obtain 128-dimensional embeddings. A separate classifier MLP was then trained on these embeddings, comprising two hidden layers of 128 and 64 units, each with batch normalization and dropout for regularization. This autoencoder–MLP pipeline achieved approximately 87.0% test accuracy on the GTZAN dataset, significantly lower than the other models.

4 Results

Table 1 Summarizes the test accuracies of all models. The baseline MLP reached about 90.99%. After hyperparameter tuning, accuracy improved to 92.33%. The deep MLP reached 92.53%. The autoencoder-based feedforward model reached 87.0%. In summary, the tuned and deep models ranked higher than the baseline MLP, although differences are relatively small (0.2–1.3%). The autoencoder version is performing worse than the baseline due to the loss of information from the compression bottleneck in the autoencoder process.

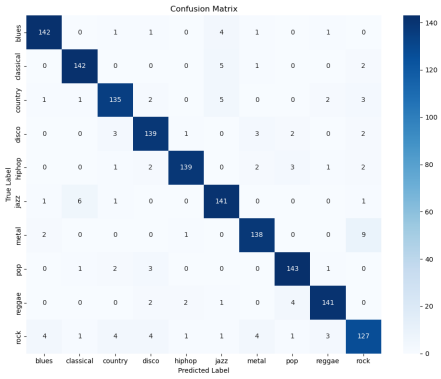


(a) Confusion matrix for Base MLP Model

```
Accuracy: 0.9099

Classification Report:
              precision    recall  f1-score   support

       blues     0.9000    0.9600    0.9290       150
   classical     0.8690    0.9733    0.9182       150
     country     0.9416    0.8658    0.9021       149
       disco     0.9000    0.9000    0.9000       150
     hiphop     0.9357    0.8733    0.9034       150
        jazz     0.9167    0.8800    0.8980       150
       metal     0.9653    0.9267    0.9456       150
         pop     0.9091    0.9333    0.9211       150
      reggae     0.9139    0.9200    0.9169       150
        rock     0.8609    0.8667    0.8638       150

    accuracy                         0.9099      1499
   macro avg     0.9112    0.9099    0.9098      1499
weighted avg     0.9112    0.9099    0.9098      1499
```

(b) Accuracy metrics for Base MLP Model

Figure 1: Base MLP Model Results



(a) Confusion matrix for Base MLP Model

📋 Classification Report:

	precision	recall	f1-score	support
blues	0.9467	0.9467	0.9467	150
classical	0.9404	0.9467	0.9435	150
country	0.9184	0.9060	0.9122	149
disco	0.9085	0.9267	0.9175	150
hiphop	0.9653	0.9267	0.9456	150
jazz	0.8981	0.9400	0.9186	150
metal	0.9262	0.9200	0.9231	150
pop	0.9346	0.9533	0.9439	150
reggae	0.9463	0.9400	0.9431	150
rock	0.8699	0.8467	0.8581	150
accuracy			0.9253	1499
macro avg	0.9254	0.9253	0.9252	1499
weighted avg	0.9254	0.9253	0.9252	1499

(b) Accuracy Metrics for Deep MLP Model

Figure 2: Deep MLP Model Results

Model	Test Accuracy (%)	Model Details
Baseline MLP	90.99	2 hidden layers (256,128), ReLU, dropout 0.3
Tuned MLP	92.33	2 hidden layers (512,448), dropout 0.277, LR=0.00493
Deep MLP	92.53	5 hidden layers (1024→512→256→128→64), dropout [0.3–0.2]
Autoencoder+MLP	87.00	AE bottleneck=128 + 2-layer MLP on embeddings

Table 1: Test accuracies of different models on the GTZAN genre classification task.

5 Discussion

By comparing the models, we can find the trade-off between complexity and performance. The baseline MLP is the simplest model, and has already achieved about 91% accuracy. Through hyper-parameter tuning, we were able to achieve a further 1.3% gain without changing the layer depth as on the deeper MLP, which suggests we had reasonable choices regarding layer sizes, learning rate, and dropout. The deep MLP had many additional parameters, and layers, but only modestly improved from the tuned MLP (0.2%), which suggests diminishing returns on these improvements. The deeper model also took longer to train and required additional care with appropriate regularization (dropout [6], and batch norm [7]).

The autoencoder-based model has a significantly different structure (combined encoder and classifier), which could allow it to potentially capture the important features; however, its accuracy was lower (87%) which suggests this redundancy forced it to lose some genre-specific information during compressing the data (into a 128-dimensional bottleneck). It probably learned good representations of the general characteristics of the signals [4], but not all discriminative characteristics.

Overall, all models used can benefit from dropout [6] and batch normalization [7], regardless of the net complexity, to generalize; these techniques contributed to reducing overfitting and stabilizing the training, particularly in larger networks.

6 Conclusion and Future Work

Based on the comparisons of MLP-based approaches to abstract the features of music genres on the GTZAN dataset that were completed here, it appears that the baseline MLP, the tuned MLP, and the deep MLP can achieved respectable accuracy (91-93%). However, the autoencoder-augmented performance was poor. The results seemed to suggest that for this dataset and for the features used, tuning and moderately deep networks were sufficient performance, whereas overly aggressive compression could limit performance. Future work might focus on the refinement of models to improve performance targeting centered and soundly explained architectures, data augmentation, repeatedly using glimpsical data, either like GTZAN that have known issues, to build more robust approaches for target generalizability. For example, under the guidance of neural temporal learning observations, convolutions may be a viable option in successive numerical processes to sequentially extract invariant information within listening (the perception of music over time) using spectrograms (an option beyond what is trivially listed in GTZAN [2]). Similarly, future work with potentially different architectures or combinations could appeal custom crafting and geo-caging observations with augmentations to much larger or even well-constructed datasets like GTZAN, to improve generalizability. Alternatively, exploring even more audio features or other ensemble approaches could be a follow-on extension from this worked discussion.

References

[1] M. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, 2002.

[2] B. L. Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," *Journal of Machine Learning Research*, vol. 14, pp. 267–305, 2013.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, 2019.

[6] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2015, pp. 448–456.