Music Genre Classification Using Enhanced Feedforward Neural Networks and Hyperparameter Optimization

Yuze Jiang
Department of Statistics
University of Michigan – Ann Arbor
Ann Arbor, USA
jyzjyz@umich.edu

***Abstract*—Music genre classification is crucial for organizing and recommending music on digital platforms. This paper presents a comprehensive machine learning approach that classifies music genres based on audio features using Python and Hugging Face resources. An enhanced feedforward neural network (FFNN) incorporating batch normalization and dropout layers is developed. Extensive data preprocessing and exploratory data analysis are conducted, including principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE). Additionally, class imbalance is addressed using Synthetic Minority Over-sampling Technique (SMOTE). Hyperparameter optimization is performed using Optuna to fine-tune the model. Experimental results show that before hyperparameter tuning, the model achieved an accuracy of 0.5514, a precision of 0.5409, recall of 0.5514, F1 score of 0.5329, and AUC-ROC of 0.9115. After tuning, these metrics improved by approximately 7% (e.g. accuracy to 0.5952) and the AUC-ROC increased to 0.9338. These findings demonstrate the effectiveness of the proposed methods and highlight future avenues of improvement.**

***Index Terms*—Music genre classification, feedforward neural network, hyperparameter optimization, Optuna, Hugging Face, machine learning**

I. Introduction

With the exponential growth of digital music content, efficient methods of organizing and recommending music genres are essential to enhance the user experience. Music genre classification helps in managing large music libraries and improves recommendation systems by categorizing songs based on their characteristics [1].

Traditional approaches typically involve processing raw audio signals or spectrograms through complex models, including convolutional neural networks (CNNs) [2], recurrent neural networks (RNNs) [3], as well as hybrid and attention-based architectures [4], [5]. While these advanced models can achieve high performance, they frequently require substantial computational resources, extensive data preprocessing, and large datasets.

In contrast, a simpler model like a feedforward neural network (FFNN) can be an appealing alternative. Although FFNs may not always outperform CNNs or RNNs in complex tasks, they can achieve competitive results when combined with careful feature engineering and hyperparameter tuning [6]. FFNs (or multilayer perceptrons) are relatively easy to implement, train, and deploy, and require fewer computational resources than more complex architectures. They have been shown to perform well on tabular or pre-extracted features, balancing complexity and interpretability [7]–[9]. Moreover, FFNs can be more straightforward to optimize and integrate into production pipelines [10]–[16].

This paper leverages the advantages of FFNs in the context of music genre classification. By using pre-extracted audio features, applying SMOTE for class imbalance, and performing systematic hyperparameter optimization with Optuna, we demonstrate that a well-optimized FFNN can achieve improved performance while maintaining efficiency and interpretability.

II. Literature Review

Music genre classification has been extensively studied. The early methods relied on handcrafted features and traditional classifiers such as support vector machines (SVMs) and k-nearest neighbors (KNN) [1]. Deep learning introduced CNNs and RNNs for learning features directly from raw audio data, achieving state-of-the-art performance [2], [3]. Hybrid architectures, attention mechanisms, and domain adaptation techniques have further pushed the boundaries [4], [5].

However, these methods often come at a high computational cost. Simpler architectures like FFNs can yield competitive results when used with pre-extracted features [6]–[16], particularly if systematic optimization techniques are employed. SMOTE [17] effectively handles class imbalance, while hyperparameter optimization frameworks like Optuna [20] streamline the search for optimal model parameters.

III. Methodology

A. Dataset Description

An open-source dataset with over 50,000 songs is used, each annotated with a genre label. The dataset includes acousticness, danceability, duration, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, key, mode, and popularity. Genres include Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical, and Hip-Hop.

B. Data Preprocessing

Data preprocessing ensures data quality and effective modeling:

- **Missing Values and Encoding**: Missing values are handled, and categorical variables (e.g., key, mode) are encoded.
- **Outlier Treatment**: Anomalies in features like duration are addressed.
- **Log Transformations**: Skewed features (e.g., instrumentalness, liveness, speechiness) are log-transformed to normalize their distributions.
- **Scaling**: Numerical features are standardized using *StandardScaler*.

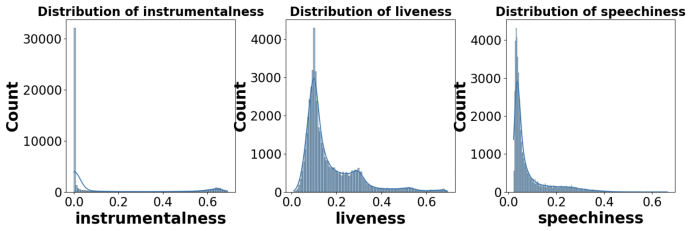
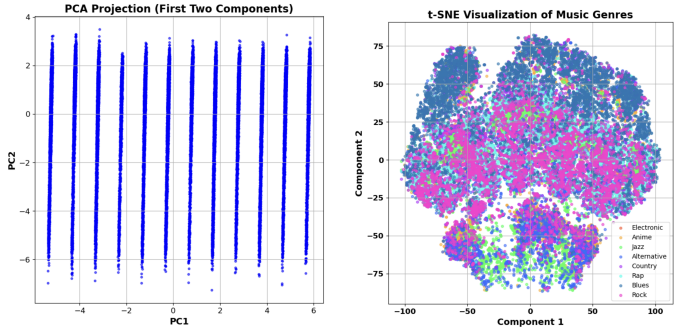


Fig. 1: Distributions of 'instrumentalness', 'liveness', and 'speechiness' before log transformation.

C. Exploratory Data Analysis

PCA and t-SNE are used to visualize data structure:

- **PCA**: The first two principal components explain about 68% of the variance. While this indicates a substantial dimension reduction, overlapping genre clusters highlight the intrinsic difficulty of separating certain genres (Fig. 2a).
- **t-SNE**: The t-SNE plot (Fig. 2b) shows clusters of points representing genres, but overlaps suggest some genres share similar feature spaces, challenging the classification task.



(a) PCA Projection (First Two Components).

(b) t-SNE Visualization of Music Genres.

Fig. 2: Visualizations of Music Genre Overlap using PCA and t-SNE.

D. Data Splitting and Balancing

Data is split into training (80%), validation (10%), and test (10%) sets. SMOTE [17] is applied to the training set to address class imbalance, creating a more balanced dataset.

E. Model Architecture: Why a FFN?

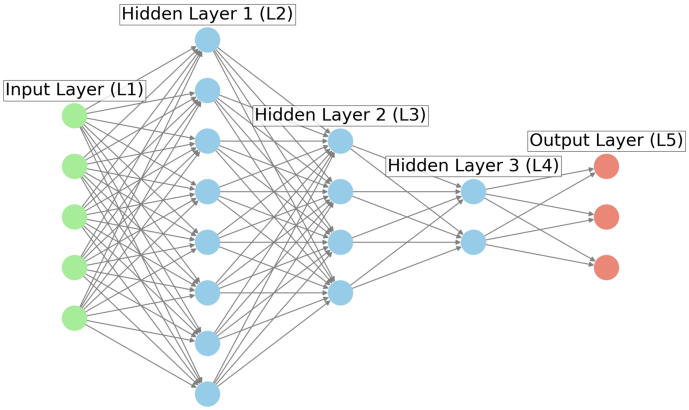


Fig. 3: The structure of a feedforward neural network with an input layer, three hidden layers, and an output layer, designed for classification tasks.

A feedforward neural network was chosen over more complex models due to its simplicity, lower computational cost, and easier interpretability. While CNNs or RNNs might achieve higher performance with raw signal inputs, a well-optimized FFN can approach competitive results on pre-extracted features without extensive resource usage [7], [8]. This trade-off makes FFNs suitable for scenarios where computational efficiency and deployment considerations are critical.

The FFN architecture includes:

- **Input Layer**: 13 features.
- **Hidden Layers**:
  - Initially: Layer 1 (256 neurons), Layer 2 (128 neurons), Layer 3 (64 neurons).
  - ReLU activation, batch normalization, and dropout in each layer.
- **Output Layer**: Softmax activation with units equal to the number of genres.

F. Training Configuration

The model is trained with:

- **Optimizer**: Adam with a tunable learning rate.
- **Loss Function**: Cross-entropy with class weights.
- **Batch Size**: Tuned via hyperparameter optimization.
- **Epochs**: Around 10-15. item **Evaluation Strategy**: Validate at the end of each epoch.

G. Hyperparameter Optimization with Optuna

Optuna [20] is used to explore hyperparameters, including learning rate, dropout rate, hidden layer sizes, and batch size. The best hyperparameters found are:

- Dropout Rate: 0.1251
- Hidden Layer Sizes: (512, 128, 64)
- Learning Rate: 9.41×10^{-4}
- Batch Size: 32

IV. Results

A. Initial Performance

Before tuning, the model achieved:

- Accuracy: 0.5514
- Precision (Weighted): 0.5536
- Recall (Weighted): 0.5514
- F1-Score (Weighted): 0.5392
- AUC-ROC (Avg): 0.9115

B. Optimized Performance

After hyperparameter tuning:

- Accuracy: 0.5952
- Precision (Weighted): 0.5979
- Recall (Weighted): 0.5952
- F1-Score (Weighted): 0.5937
- AUC-ROC (Avg): 0.9338

These metrics represent approximately a 7% improvement over the initial model in accuracy, precision, recall, and F1-score, and an increase in AUC-ROC from 0.9115 to 0.9338.

C. Loss Curves and ROC Analysis

Loss curves show smoother convergence and reduced overfitting after tuning (Fig. 4). ROC curves (Fig. 5) indicate that while some genres are easily distinguishable, others remain challenging due to overlapping features. The improved AUC-ROC suggests more consistent discriminative power across genres.

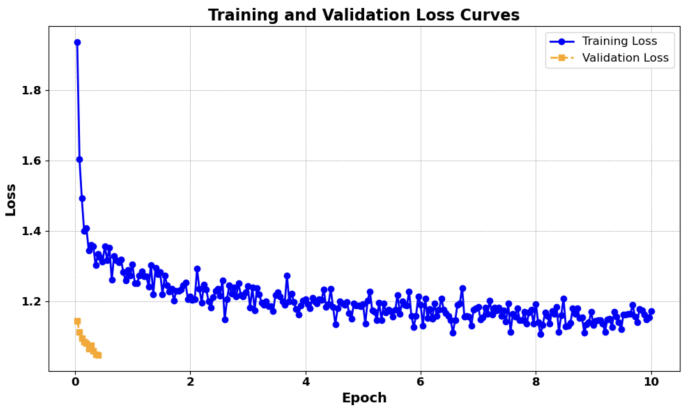


Fig. 4: Training and Validation Loss Curves for the Optimized Model.

D. Interpretation of Results

The hyperparameter optimization improved the model's capacity to leverage pre-extracted features effectively. The PCA and t-SNE analyses help explain persistent difficulties in distinguishing certain genres due to intrinsic feature similarities. However, the 7% gain across multiple metrics and improved AUC-ROC underscore the value of systematic tuning, class balancing, and a well-chosen architecture like an FFN.

V. Discussion

This work demonstrates that a designed and tuned FFN can achieve competitive results in music genre classification without resorting to more complex architectures. The combination of SMOTE, feature preprocessing, and Optuna-based hyperparameter optimization significantly improved model performance.

The results indicate that, while FFNs can close much of the performance gap, certain intrinsic challenges persist, such as genre overlap. Future research could explore the incorporation of additional features, the use of attention mechanisms, or the use of transformer-based models to further enhance classification capabilities.

VI. Conclusion

An enhanced FFN-based approach to music genre classification, combined with robust preprocessing and hyperparameter optimization, yields notable performance improvements. The model's accuracy, precision, recall, and F1-score each improved by around 7%, and the AUC-ROC increased from 0.9115 to 0.9338 after tuning. These findings highlight the utility of optimizing simpler models in resource-constrained or rapidly deployable scenarios.

Future work includes exploring more advanced architectures, integrating raw audio signals, and refining feature extraction techniques. The demonstrated approach presents a robust baseline for the development of more sophisticated systems, or alternatively, a practical and efficient solution for music genre classification tasks.

Data and Code Availability

The code and trained models are available on GitHub (https://github.com/StephanJ911/Music$_G$enre$_C$lassification.git) and on Hugging Face (https://huggingface.co/JyzJiang/music-genre-classifier).

References

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

[2] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396.

[3] S. Sigtia and S. Dixon, "Improved music feature learning with deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6959–6963.

[4] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968.

[5] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[6] H. Zhang, Y. Zhang, and Q. Yang, "Improved deep learning baseline for music classification," in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6.

[7] R. Caruana *et al.*, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 161–168, 2006.

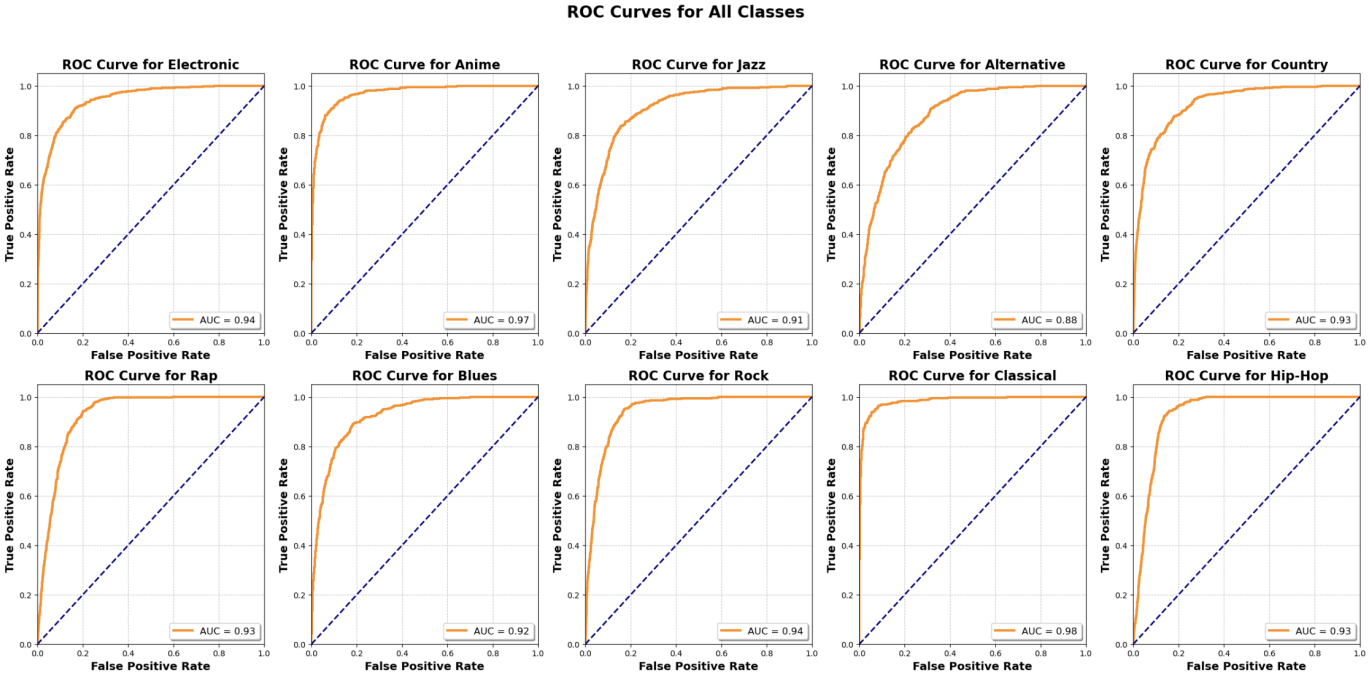


Fig. 5: ROC Curves for Each Genre.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

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

[10] E. Jordaan, "Feedforward neural networks and feedback networks," in *Handbook of Neural Computation*, IOP Publishing Ltd, 2002.

[11] W. Wang *et al.*, "A comparative study of MLP and CNN in raw data classification," in *2019 IEEE Symposium on Computational Intelligence*, pp. 1–7.

[12] T. He *et al.*, "Bag of tricks for image classification with convolutional neural networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 558–567.

[13] A. Maas *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[15] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[16] K. Clark *et al.*, "Semi-supervised sequence modeling with cross-view training," in *2018 EMNLP*, pp. 1914–1925.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[19] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.