A Survey On Audio Signal Classification Using Transfer Learning: CNN and VGGish implementation on the GTZAN Dataset

Emir Uncu
Dept. of Electrical And Electronics Eng.
Middle East Tech. Univ.
e273968@metu.edu.tr
GitHub Repo: https://github.com/emiiruncu10/CENG562_Project.git

ABSTRACT

This paper investigates the application of transfer learning techniques for music genre classification using the GTZAN dataset. Two deep learning approaches are implemented and compared: a convolutional neural network (CNN) trained from scratch using Mel-spectrogram features, and a transfer learning model employing VGGish embeddings as input to a classification head. While the scratch-based CNN achieved higher accuracy, it required extensive training time and computational resources. In contrast, the VGGish-based model offered competitive performance with faster convergence, even when trained on a CPU. The results demonstrate the viability of using pretrained audio representations in low-resource scenarios and emphasize the importance of model selection, dataset quality, and adaptation strategies. The paper concludes with a discussion on current challenges and proposes future directions, including fine-tuning modern audio embeddings and improving data preprocessing for enhanced model generalization.

1 Introduction

Audio classification plays a significant role in various real-world applications, including automatic speech recognition, environmental sound classification, speaker identification, and music classification. Traditional machine learning techniques in this domain often rely on handcrafted feature extraction, where audio signals are transformed into spectrograms or feature representations such as Mel-frequency cepstral coefficients (MFCCs) before classification, using machine learning algorithms. However, with the advancement of deep learning, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have demonstrated superior performance in automatically extracting relevant features from raw audio data. This paper mainly focusing on music genre classification, aiming to classify audio signals into predefined music genres using the transfer learning techniques. Music-genre labels (rock, jazz, hip-hop,…) help streaming apps build playlist and help researchers study how music is organized. Teaching a computer to hear a song and guess its genre is therefore useful for both industry and science. Performing this task is quite challenging, because musical style depends on rhythms, melodies and instruments not just words or simple sounds. Modern neural networks can learn these patterns, but they usually need millions of labelled songs. The popular GTZAN dataset gives us only 1000 short clips (100 per genre 30 sec long), which is suitable for implementing simple models needing small amount of data [1]. Its modest size makes it attractive for fast experiments but also renders end-to-end CNN training prone to overfitting. In order to overcome these challenges while still using the GTZAN dataset, we will refer to the "transfer learning" technique, which is emerged as a powerful approach that enables the reuse of knowledge from pre-trained models. By leveraging pre-trained architectures, transfer learning allows deep learning models to generalize better, particularly in scenarios where labeled data is scarce. Transfer learning lets us fine-tune only a small fraction of weights, achieving high accuracy with limited data and computation [2], [3]. This paper surveys TL techniques for small-scale audio classification and implements a representative pipeline around:

- Input Representation: Log-mel spectrogtams, which transform one-dimensional waveforms into two-dimensional images suitable for CNNs.
- Base Model: "VGGish" a compact CNN pre-trained on AudioSet, chosen for its public weights, stable embedding interface, and ability to run-on laptop-class GPUs and CPUs [4]
- Adaptation Strategy: All convolutional blocks are frozen; only the classifier head(and optionally the top convolutional block) is fine tuned to map AudioSet features onto the ten GTZAN genres.

GTZAN serves as the "toy problem" for empirical evaluation. Although subsequent studies have noted duplicates and occasional mis-labels[5], its ten-class structure and long research lineage allow straightforward baseline comparisons. This paper is structured as follows: Section 2 Surveys classical machine-learning and recent deep-learning approaches to genre recognition, highlighting where transfer learning has provided measurable gains. Within section 3 preliminary literature review is given, which Summarizes the prior work on genre classification, emphasizing where transfer learning improved robustness or accuracy. In section 4 methodology and implementation details will be presented, including the use of fully frozen VGGish model as a feature extractor, without fine-tuning its internal layers. The extracted features are then used to train a custom classifier for the audio classification task. Additionally, this section will compare two CNN-based architectures: one designed and trained from scratch, and the other built using a transfer learning approach that leverages the VGGish model. The comparative evaluation highlights the strengths and limitations of both methods in terms of performance and training efficiency. Within section 5 results and discussions will reviewed such as specifying implementation details, hyper-parameters optimizer settings, and evaluation metrics and justifying the baseline model used for comparison. In section 6, the main challenges and open problems encountered throughout this study will be discussed. This includes difficulties in training deep CNN architectures from scratch due to the limited dataset size and high computational cost, as well as challenges related to genre overlap and label ambiguity in music classification. The section will also address the limitations of using fully frozen pre-trained model like VGGish, including domain mismatch issues and the lack of flexibility in adapting to task-specific nuances. These discussions aim to highlight areas where further research and improvement are needed to enhance the robustness and generalizability of transfer learning approaches in audio classification tasks. Finally, in section 7, a summary of the key findings will be presented, highlighting the comparative performance on the scratch-based CNN and the transfer learning approach using VGGish. The section will also reflect on the benefits and limitations observed during implementation and suggest directions for future improvements in audio classification using transfer learning techniques.

2 Background and Problem Definition

Audio classification is the task of automatically categorizing audio signals into meaningful classes (e.g. music genres, environmental sounds etc.). Traditional approaches rely on carefully engineered features to represent audio content. One of the most established features are Mel-frequency cepstral coefficients(MFCCs), which compactly represent the short-term spectral envelope of the sound. MFCCs are widely used in speech and audio analysis, as they model the human ear's frequency response by using a nonlinear mel scale. To compute MFCCs, one typically performs a short-time Fourier transform (STFT) on the signal, applies a bank of mel-spaced filters to the power spectrum, takes the logarithm of each filters energy and then applies a discrete cosine transform (DCT). This yields a set of decorrelated coefficients (often ~12-13) that summarizes the spectral shape. STFT is a technique to analyze how the frequency content of the signal changes over time. First the audio signal is divided into short frames (e.g. 20-40 ms), for each frame the Fast Fourier Transform (FFT) is applied in order to extract frequency information. These steps provides time-frequency representation: A spectrogram (frequency content vs. time). Since the audio signals are non-stationary (they change over time), they have to be analyzed within small chunks. After the STFT, the power spectrum is obtained for each frame. After passing through a filter bank(consist of a set of overlapping triangular filters), it is observed that each filter sums up energy in its frequency band (having one energy value per filter). This step compresses the high-resolution frequency spectrum into a more perceptually meaningful set of frequency bands. The mel-filter energies logarithms are taken in order to compress the dynamic range(loud and soft sounds), make the representation more similar to human loudness perception (logarithmic), help the features match better with Gaussian distributions used in classifiers. After these processes the log-mel energies are still correlated (neighboring filter outputs are similar). The DCT decorrelates them. Turning overlapping, redundant filter values into a compact, orthogonal set of MFCCs. Usually, only the first 12-13 DCT coefficients are kept, as they represent the overall spectral shape. The Fig1 is the simple diagram showing the full MFCC pipeline.

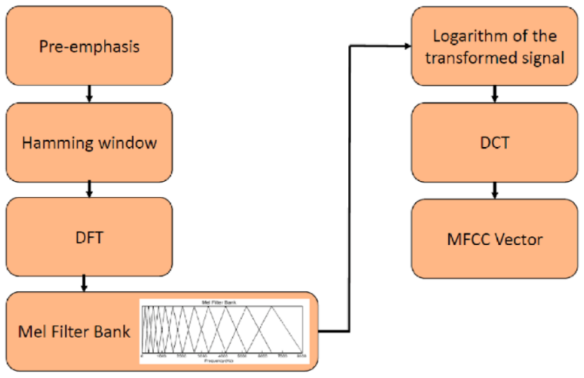


Fig 1: simple MFCC pipeline diagram

These coefficients have proven effective in speech recognition and classification tasks, as they capture perceptually important spectral characteristics in a compact form. However, MFCCs are a highly compressed representation and may discard fine details that could be relevant for complex audio classification tasks. In contrast to using hand-crafted features like MFCCs, modern approaches often use richer time-frequency representations such as spectrograms. A spectrogram is a visual representation of sound typically showing time on the x-axis, frequency on the y-axis, and amplitude (or power) as intensity (often on a logarithmic scale). A common choice is the log-mel spectrogram, which is a time-frequency plot after mapping frequencies to the mel scale and taking a log of the amplitude. This representation retains more detailed frequency information than a small set of MFCCs while still incorporating perceptual scaling. In the field of audio recognition, the log-mel spectrogram is generally regarded as one of the most powerful input features because it aligns with human auditory perception and provides a 2D time-frequency map of the sound. By preserving the temporal evolution of frequency content, spectrograms allow learning algorithms to exploit local patterns in both time and frequency. Using spectrogram-based features as input has enabled the rise of data-driven methods for audio classification. In particular, (CNNs) can be applied to spectrograms in a similar manner as they are to images, by treating time-frequency representations as 2D grids. **Convolutional Neural Networks (CNNs)** are a class of deep learning models particularly effective for processing data with a grid-like topology, such as images and spectrograms. In the context of audio classification, spectrograms (visual representations of the frequency spectrum of audio signals over time) serve as input to CNNs, enabling the model to learn and extract hierarchical features pertinent to different audio classes. A typical CNN architecture comprises several types of layers, each serving a specific function:

- **Input layer:** Receives the input data, such as spectrogram image representing the audio signal.
- **Convolutional Layers:** Apply filters (kernels) to the input data to detect local patterns, such as edges or textures. Each filter produces a feature map highlighting the presence of specific features in the input.
- **Activation Functions:** Introduce non-linearity into the model, allowing it to learn complex patterns. The rectified linear unit (ReLU) is commonly used.
- **Pooling Layers:** Reduce the spatial dimensions (width and height) of the feature maps, thereby decreasing computational load and controlling overfitting. Max pooling is a prevalent technique, selecting the maximum value within a pooling window.
- **Fully Connected Layers:** Flatten the output from the convolutional and pooling layers and pass it through one or more dense layers to perform high-level reasoning and classification.
- **Output Layer:** Produces the final classification output, typically softmax activation function to generate probabilities for each class.

Fig2 showing the simple CNN architecture for a basic animal classification problem:

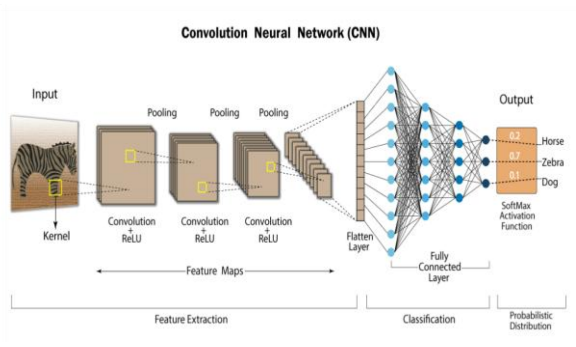


Fig2 : CNN architecture for a basic animal classification problem

CNNs automatically learn to detect useful time-frequency patterns (such as timbral textures or harmonic structures) from the spectrogram, alleviating the need for manual feature engineering. Indeed CNN, architectures have demonstrated a strong ability to capture salient audio features across time and frequency. The ability of CNNs to learn hierarchical representations, from low-level spectral edges up to high-level audio object patterns makes them well-suited to audio classification. As a result, modern audio classification systems often involve a CNN operating on a spectrogram or related representation, either trained from scratch or using a pretrained model via transfer learning. While deep CNNs can learn rich audio features, training them from scratch typically demands a large amount of labeled data. In many audio domains (music, environmental sounds, bioacoustics, etc.), dataset sizes are relatively small, which can lead to overfitting when training a complex model solely on the limited data. **Transfer learning** addresses this issue by leveraging knowledge from a source task with abundant data to improve performance on a target task with sparse data. In general, transfer learning for audio involves using a model that has been pretrained on a large audio dataset and adapting it to the task of interest. The pretrained model's learned parameters serve as a rich initialization or as a fixed feature extractor, carrying over learned auditory patterns that can benefit the target task. This approach mirrors the success of transfer learning in computer vision, where CNNs trained on millions of images (e.g. ImageNet) are re-used to boost performance on smaller image dataset. Fig3 shows us the simple Transfer learning idea within an animal classification example.

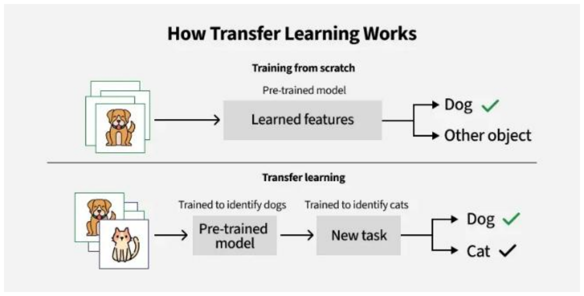


Fig3: Transfer Learning Example

By leveraging representations learned from the source task, the model can adapt to the target task with greater efficiency, resulting in accelerated convergence and improved predictive accuracy. Moreover, transfer learning mitigates the risk of overfitting by incorporating generalized feature representations that enhance the model's ability to generalize across tasks. Transfer learning is methodological approach that repurposes knowledge acquired from a pre-trained model to enhance performance on a different but related target task. The process typically involves the following stages:

Pre-trained Model: The foundation of transfer learning is a model previously trained on a large-scale dataset for a source task. This model has developed a rich set of feature representations that capture fundamental patterns, often transferable across domains.

Base Model Architecture: The pre-trained model, referred to as the base model, comprises a hierarchical architecture. Early layers learns low-level features (e.g. edges, textures), while deeper layers capture more abstract and task-specific representations.

Transferable Feature Layers: Layers that encode general-purpose features commonly located in the earlier to intermediate stages of the model are identified to reuse. These layers provide a transferable knowledge base, applicable to the target task.

Fine-Tuning: To tailor the model to the new task, selected layers are fine-tuned using data specific to the target domain. This process allows the model to preserve beneficial pre-learned features while adapting its parameters to task-specific nuances, thereby enhancing generalization and predictive performance.

In the context of transfer learning, the model architecture is typically divided into two distinct components to facilitate effective adaptation to a new task. **Frozen layers** are inherited from the pre-trained model, which is kept fixed during the fine-tuning process. They encapsulate generalized feature representations learned from the source task, such as edges, textures, or shapes, which are broadly applicable across various domains. By preserving these weights, the model leverages stable and transferable knowledge while reducing computational overhead and the risk of overfitting. **Trainable Layers** are selectively updated during the fine-tuning phase using data from the target task. These layers are responsible for learning task-specific features that address the unique characteristics and requirements of the new domain. Adjusting these layers enables the model to specialize and achieve improved performance on the target application.

3 Preliminary Literature Review

Transfer learning has become a widely adopted strategy in music genre classification to address data scarcity and improve model generalization. Instead of training models from scratch, transfer learning leverages feature representations learned from large-scale datasets. This particularly valuable for audio classification tasks where labeled data such as the GTZAN dataset is limited. **VGGish,** a CNN model pre-trained on YouTube's AudioSet, is a commonly used feature extractor in audio research. Studies have shown that using embeddings from VGGish significantly improves classification accuracy across various tasks, including environmental sound recognition and music tagging [3][6]. In genre classification, VGGish has been used as a fixed feature extractor or fine-tuned with smaller datasets to boost accuracy and convergence speed [7]. In a seminal study, Choi et al.(2017) demonstrated the effectiveness of transfer learning by training a CNN on a large music tagging dataset(244k tracks) and reusing the learned features for genre classification on GTZAN, achieving up to 89.9% accuracy substantially higher than traditional MFCC-based methods [8]. Other studies explored OpenL3 and SoundNet embeddings, often outperforming models trained from scratch on datasets like FMA and MagnaTagATune [9]. Researchers have also proposed hybrid approaches such as neural model reprogramming, enabling cross-domain transfer from speech to music tasks with promising results[10]. In summary, transfer learning becomes a cornerstone of modern audio classification. Thanks to deep learning and large-scale datasets, we regularly use pre-train models and fine-tune them for new tasks with significantly better results. Models like Choy et al. convolutional network for music tagging, Google's VGGish and more recent PANNs have proven that learned audio representations can be effectively transferred to new domains. Just like in computer vision or natural language processing, the audio field now benefits from rich, pre-trained models. Leveraging them lead to %5-%10 absolute improvements in accuracy, better generalization across domains, and much faster development cycles. Despite these advances, challenges remain: domain mismatch, fine-tuning complexity, and limited generalization across datasets. Nonetheless, the literature supports the use of CNN-based transfer learning (especially VGGish) as an effective and practical solution for genre classification with constrained data.

4 Methodology and Implementation

Method-1: The first implemented method for this study is a Convolutional Neural Network(CNN) designed to classify music genres based on audio-derived features. CNNs are a class of deep learning models well-suited for grid-like data such as images and spectrograms, due to their ability to automatically learn spatial hierarchies through local receptive fields and shared weights. In this implementation, audio signals are first transformed into **Mel-spectrograms** (shown in "Fig5"), which are two-dimensional representations capturing both frequency and time information, structured similarly to gray-scale images. These spectrograms are then used as input to the CNN, which consists of multiple convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The network is trained using the categorical cross-entropy loss function, with the Adam optimizer employed to update the weights based on the backpropagated gradients. The final layer uses a softmax activation function to output a probability distribution over the set of predefined music genres. This end-to-end learning approach enables the model to discover genre-discriminative features directly from the input data without need for handcrafted features.

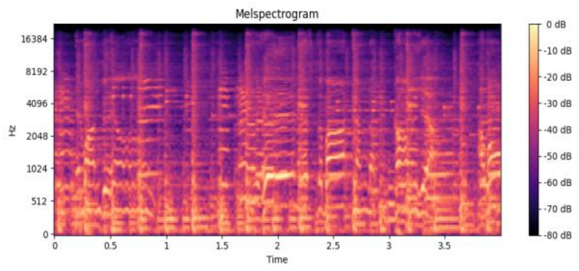


Fig5: Example of mel-spectrogram

Dataset Choice: For the implementation, the **GTZAN** dataset was selected as the benchmark dataset for music genre classification. GTZAN is widely used and publicly available collection of audio samples, consisting of 1,000 audio tracks, each 30 seconds long, equally distributed across 10 music genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. This makes it a K>2 class classification problem, where the goal is to assign each audio clip to one of the predefined genre labels. The dataset is particularly suitable for evaluating deep learning-based approaches such as CNNs due to its relatively balanced class distribution and moderate size, which allows for efficient experimentation and prototyping without excessive computational overhead. The musical diversity and genre separability in GTZAN make it an appropriate toy problem to test the efficacy of spectrogram-based CNN architectures for audio classification tasks.

Preprocessing Steps and Data Augmentation: The audio data from the GTZAN dataset was first loaded using **Librosa** at a sampling rate of 44.1 kHz. Each audio file was segmented into shorter frames to enhance model generalization and to create more training samples. For feature extraction, Mel-spectrograms were computed using Short-Time fourier transform (STFT), followed by conversion to the Mel scale with a logarithmic transformation. These spectrograms, treated as 2D images, served as the input for the convolutional model. The data was normalized to standardize feature values, facilitating faster convergence. No artificial data augmentation techniques were explicitly applied in this version.

Implementation Details-1: The model was implemented in **Python,** utilizing **TensorFlow** and **Keras** as the core deep learning frameworks. For audio processing and feature extraction, **Librosa** was employed to convert audio signals into Mel-spectrograms. The architecture of the baseline model is a deep convolutional neural network consisting of three convolutional layers with increasing filter sizes, each followed by ReLU activations and MaxPooling layers to downsample feature maps. After flattening, two fully connected (dense) layers are applied, followed by a Dropout layer for regularization. The final classification layer uses Softmax activation to predict the class probabilities. The model was trained using the

Adam optimizer and categorical cross-entropy as the loss function. Training proceeded over a fixed number of epochs with early stopping based on validation accuracy. Key hyperparameters such as learning rate (0.0001), batch size (32), and dropout rate (0.3) were selected empirically. Due to a preprocessing error encountered while handling the **Jazz** genre files, one class had to be excluded, resulting in a 9-class classification task instead of the originally planned 10. Despite multiple attempts, corrupted or unreadable files within the Jazz category led to imbalance and inconsistency in the dataset, and the genre was removed to maintain the integrity of the training.

Method-2: In this method, a transfer learning approach is adopted by utilizing the VGGish model(a pretrained convolutional neural network originally trained on AudioSet). Instead of training a CNN from scratch, this approach leverages the general audio representations already learned by VGGish, which can capture meaningful low and mid level features from input audio. VGGish processes log-mel spectrogram patches and outputs a **128-dimensional embedding vector** for each input frame. These embeddings serve as compact, informative feature descriptors for downstream classification tasks. Importantly, the pretrained VGGish model is used in a fully frozen state, meaning its internal weights are not updated during training. This enables the model to retain robust general audio features without overfitting to the relatively small target dataset.

Implementation Details-2: The VGGish-based model was implemented using Python, leveraging the TensorFlow, Keras, and TensorFlow Hub libraries. Audio was preprocessed using Librosa to extract log-mel spectrograms that match VGGish input requirements. These were passed through the frozen VGGish network to generate 128-dimensional embeddings for each sample. The extracted embeddings were then used to train a shallow neural network comprising one or more fully connected (Dense) layers. A ReLU activation function was applied between hidden layers, followed by a Dropout layer for regularization. The final layer employed a Softmax activation to classify the samples into one of 10 music genres. The model was trained using the Adam optimizer, categorical cross-entropy loss, and early stopping based on validation accuracy.

No automated hyperparameter tuning techniques were applied for both models. Models performance was evaluated using Accuracy, Loss, recall, precision and confusion Matrix metrics.

5 Results and Discussion

The scratch CNN model was trained over 30 epochs using a batch size of 32, with the training and validation progress monitored via loss and accuracy metrics. The final validation accuracy reached approximately %91, indicating that the model successfully learned discriminative features for classifying the 10 music genres from Mel-spectrogram inputs. Training loss decreased steadily, while validation loss stabilized in later epochs, suggesting good convergence without significant overfitting. The training process was visualized through the following plots:

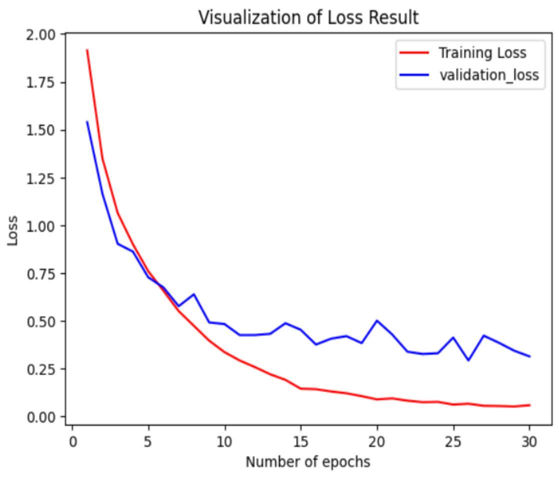


Fig6: Visualization of Loss Result

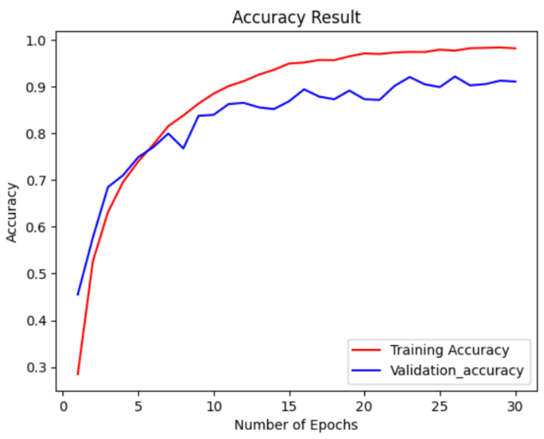
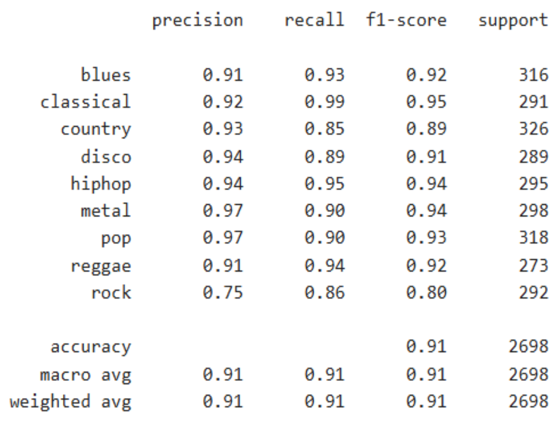


Fig7: Visualization of Accuracy Result

These plots indicate that both training and validation metrics followed a similar trajectory, reinforcing the generalizability of the learned features. Beyond accuracy, the evaluation included a detailed analysis using precision, recall, and F1-score(shown in fig8), computed via classification_report from sklearn.metrics. The report indicated that most genres were predicted with high precision and recall, with F1-scores above 0.85 for the majority of classes, suggesting balanced performance across all genre categories. A confusion matrix(shown in fig9) was also plotted to visualize class-wise performance. It revealed that most misclassifications occurred between genres with overlapping acoustic features(e.g., rock vs metal, or pop vs disco), which is consistent with observations in related studies. The matrix indicated strong diagonal dominance, affirming that the model accurately captured genre-defining patterns in the spectrograms.

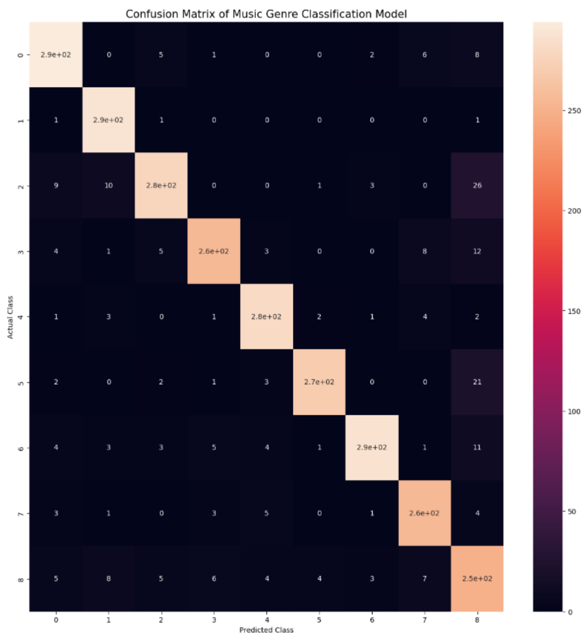


```
              precision    recall  f1-score   support

       blues       0.91      0.93      0.92       316
   classical       0.92      0.99      0.95       291
     country       0.93      0.85      0.89       326
       disco       0.94      0.89      0.91       289
      hiphop       0.94      0.95      0.94       295
       metal       0.97      0.90      0.94       298
         pop       0.97      0.90      0.93       318
      reggae       0.91      0.94      0.92       273
        rock       0.75      0.86      0.80       292

    accuracy                           0.91      2698
   macro avg       0.91      0.91      0.91      2698
weighted avg       0.91      0.91      0.91      2698
```

Fig8: Precision, recall, f1-score metrics



	0	1	2	3	4	5	6	7	8
0	2.9e+02	0	5	1	0	0	2	6	8
1	1	2.9e+02	1	0	0	0	0	0	1
2	9	10	2.8e+02	0	0	1	3	0	26
3	4	1	5	2.6e+02	3	0	0	8	12
4	1	3	0	1	2.8e+02	2	1	4	2
5	2	0	2	1	3	2.7e+02	0	0	21
6	4	3	3	5	4	1	2.9e+02	1	11
7	3	1	0	3	5	0	1	2.6e+02	4
8	5	8	5	6	4	4	3	7	2.5e+02

Fig9: Confusion Matrix

On the other hang the VGGish-based model also showed promising performance in the music genre classification task. Utilizing the pretrained VGGish network as a frozen feature extractor, the model achieved a validation accuracy of %82 with relatively low computational cost.

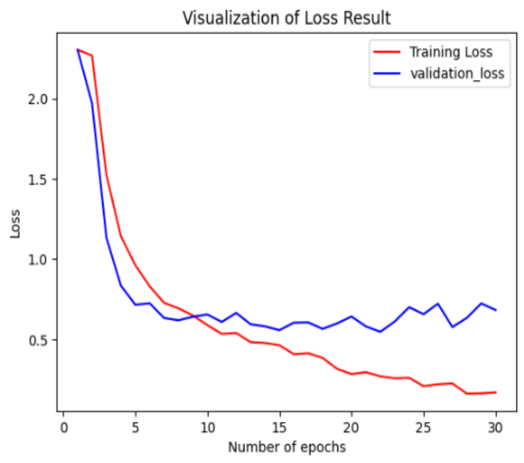


Fig10: Visualization of Loss Result VGGish

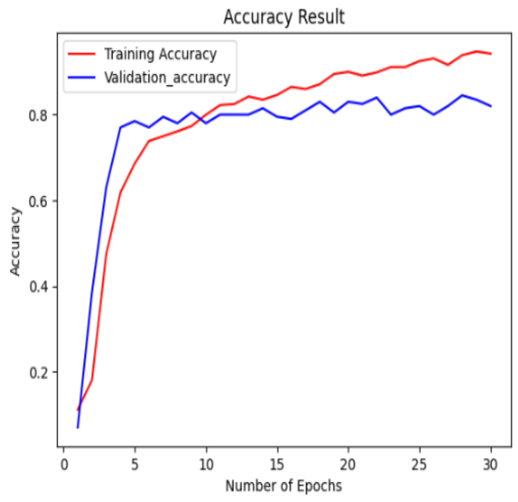
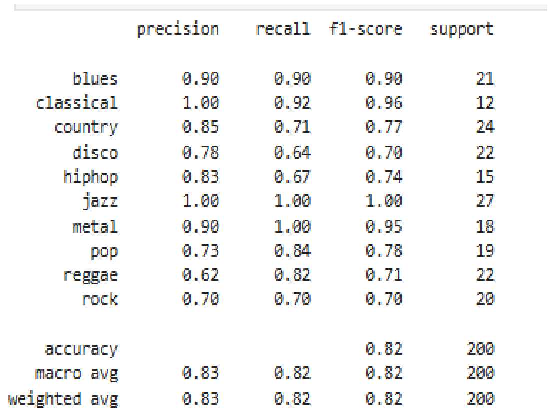


Fig11: Visualization of Accuracy Result VGGish

The training and validation curves converged smoothly, and early stopping was triggered at an optimal point without signs of overfitting.



```
              precision    recall  f1-score   support

       blues       0.90      0.90      0.90        21
   classical       1.00      0.92      0.96        12
     country       0.85      0.71      0.77        24
       disco       0.78      0.64      0.70        22
      hiphop       0.83      0.67      0.74        15
        jazz       1.00      1.00      1.00        27
       metal       0.90      1.00      0.95        18
         pop       0.73      0.84      0.78        19
      reggae       0.62      0.82      0.71        22
        rock       0.70      0.70      0.70        20

    accuracy                           0.82       200
   macro avg       0.83      0.82      0.82       200
weighted avg       0.83      0.82      0.82       200
```

Fig12: Precision, recall, f1-score metrics

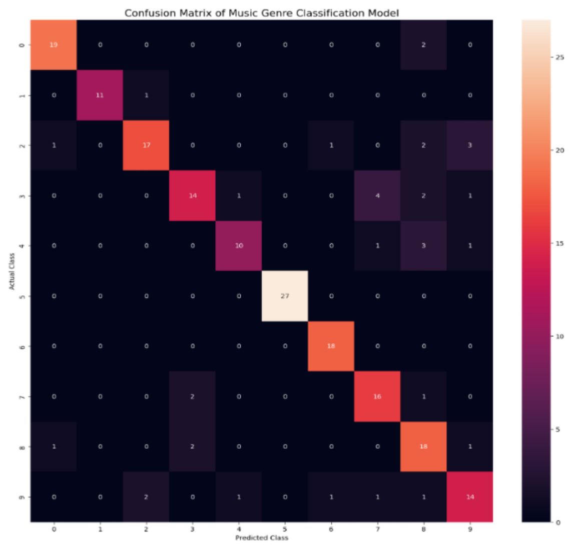


Fig13: Confusion Matrix

The VGGish-based model achieved a solid overall accuracy, with a macro average F1-score of 0.82, reflecting balanced performance across most of the ten music genres. Several genres such as classical, jazz and metal stood out with exceptionally high score. Classical reached a precision of 1.00 and F1-score of 0.96, jazz achieved perfect precision and recall (1.00), and metal followed closely with an F1-score of 0.95. These results indicate that genres with distinct harmonic and timbral features were well captured by the VGGish embeddings. However, performance declined on genres like reggae (F1-score: 0.71), rock (F1-score: 0.70) and hiphop (F1-score 0.74), which may be attributed to overlapping acoustic characteristics or insufficient representation in the feature space. Country and disco also had moderately lower F1-scores (0.77 and 0.70 respectively), suggesting these genres were harder to distinguish with fixed features from the pretrained model. Despite these variations, the model maintained consistent generalization and robust classification across dataset without overfitting, a key benefit of using VGGish as a frozen feature extractor. When we compare both of these models we can clearly see that, the scratch-based CNN outperforming the VGGish-based model. However, this came at the cost of significantly higher training time, increased model complexity, and a greater risk of overfitting. The model required GPU acceleration to reach convergence in a reasonable timeframe, and even then, it required careful hyperparameter tuning, such as dropout adjustment and early stopping, to maintain generalization. In contrast, the transfer learning approach offered a far more efficient pipeline. The VGGish-based model trained on CPU was faster to implement, less sensitive to hyperparameter changes, and inherently more stable due to the frozen pretrained layers. While it may underperform slightly in terms of peak accuracy, its reliability, training efficiency, and scalability make it an attractive alternative for projects constrained by data or compute. In summary, while training a CNN from scratch can yield better accuracy under ideal conditions, transfer learning with a model like VGGish provides a practical balance between performance and resource constraints, making it particularly suitable for real-world applications or rapid prototyping in audio classification tasks.

6 Challenges and Open Problems

The task of music genre classification faces persistent challenges, especially when applying transfer learning in this domain. A fundamental issue is data scarcity and quality. The widely-used GTZAN dataset introduced by Tzanetakis & Cook, contains only 1000 audio excerpts and is rife with faults like duplicate and mislabeled tracks, making it a noisy, small-scale benchmark that undermines model generalization. Transfer learning is appealing here because it can leverage models trained on larger datasets to compensate for limited music data. However, the effectiveness of this approach is often limited by a misalignment between the source and the target domains. The implemented pretrain model VGGish provide rich features, but these features may not be optimal for music genres. In fact, representations learned for one audio task can fail to capture information crucial for another task like musical genre recognition, a phenomenon known as negative transfer where using out-of-domain knowledge can hurt performance. Furthermore, many studies initially employ such pretrained networks as frozen feature extractors without fine-tuning; this practice can cap the performance, since fixed embeddings often omit genre-discriminative cues and can be outperformed by models tuned to the specific task. On a small and noisy dataset like GTZAN, these issues are amplified, making it hard to reliably evaluate transfer learning models. Recent research suggests several directions to address these open problems. One approach is to adopt more domain-relevant pretrained models: for example, modern audio embeddings like **OpenL3** and **PANNs(**Pretrained Audio Neural Networks**)** have been shown to outperform older VGGish features on sound classification tasks, thanks to training on large-scale audio data (OpenL3 on AudioSet's music subset, PANNs on 5800 hours of AudioSet). Fine-tuning such pretrained models- rather than relying on completely frozen features – further allows to transferred knowledge to better align with genre-specific characteristics, often boosting classification accuracy. In addition, ensemble methods that combine multiple models can leverage complementary strengths, and data augmentation techniques (e.g. pitch shifting, time stretching) can expand effective training size and improve robustness on limited dataset. Finally, improving the dataset itself remains crucial: curating larger, more diverse and correctly labeled genre collections (or cleansing existing ones like GTZAN) would provide a stronger foundation for both training and evaluating transfer learning approaches. Choosing more suitable pretrained models, adapting them to the music domain, and mitigating data limitations is key to advancing transfer learning performance in music genre classification.

7 Conclusion

This study explored the effectiveness of deep learning and transfer learning for music genre classification using the GTZAN dataset. Two primary approaches were implemented and compared: a scratched-trained CNN leveraging Mel-spectrograms, and a transfer learning pipeline utilizing pretrained VGGish embeddings. While the scratch CNN achieved superior accuracy, it required more computational resources and careful tuning. The VGGish-based model, although slightly lower in peak performance, offered efficient training and robust generalization, making it attractive for resource-constrained scenarios. The comparative analysis underscored the trade-offs between training complexity and model performance. Furthermore, it highlighted that frozen audio embeddings, while powerful, may fail to fully capture task-specific nuances such as genre overlaps. These findings reinforce the importance of model selection, dataset quality, and adaptation strategy in transfer learning workflows. Future work should investigate fine-tuning modern pretrained models like PANNs or OpenL3, experimenting with ensemble approaches, and curating cleaner, larger datasets to overcome domain mismatch and data scarcity issues. Overall, the study confirms that transfer learning holds significant promise for small-scale audio classification tasks and serves as a practical foundation for further research in this domain.

8 References

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

[2] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] H. Hershey *et al.*, “CNN architectures for large-scale audio classification,” in *Proc. ICASSP*, 2017, pp. 131–135.

[4] A. Van Den Oord *et al.*, “Transfer learning with VGGish for audio event and scene classification,” *arXiv:1807.09902*, 2018.

[5] B. L. Sturm, “The GTZAN dataset: Its contents, faults, their effects on evaluation, and its future use,” in *Proc. ISMIR*, 2013, pp. 7–12.

[6] Q. Kong *et al.*, “PANNs: Large-scale pretrained audio neural networks for audio pattern recognition,” *IEEE TASLP*, 2020.

[7] E. Koh and S. Dubnov, “Comparison of deep audio embeddings for music emotion recognition,” *AAAI Workshop*, 2021.

[8] K. Choi *et al.*, “Transfer learning for music classification and regression tasks,” *Proc. ISMIR*, 2017.

[9] J. Salamon *et al.*, “Learning audio embeddings for music classification,” *IEEE TASLP*, 2019.

[10] Y. Hung *et al.*, “Low-resource music genre classification with neural model reprogramming,” *Proc. ICASSP*, 2023.

GitHub repo: https://github.com/emiiruncu10/CENG562_Project.git