**Music Genre Classification using**

**Audio-Spectrogram-Transformer**

Mini-Project report submitted in partial fulfilment of the

Requirements for the degree of

**Master of Engineering**

**ME (Big Data Analytics)**

By

**Nikhil M (241058020)**

**Nikhil S G (241058024)**

**Vinayashree M Shet (241058030)**



**MANIPAL SCHOOL OF INFORMATION SCIENCES**

**(A Constituent unit of MAHE, Manipal)**

# Acknowledgement

It is great pleasure to thank the people behind the success of this Mini-Project activity. I owe my deepest gratitude to all those who guided, inspired, and helped me to complete this project.

I would like to take this opportunity to express my gratitude and heartiest thanks to my panel members, **Prof. Mr. Raghudathesh G P**, Associate Professor, Manipal School of

Information Sciences for his inspirational support and guidance throughout my project period.

My heartful gratitude to **Dr Keerthana Prasad** Professor & Director Manipal School of Information Sciences for his full support and encouragement during the Mini-Project activity.

I would like to thank all sources mentioned in the references.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Table No** | **Figure Title** | **Page No** |
| 1 | AST Architecture | 6 |
| 2 | Diagram of Model workflow | 8 |
| 3 | Diagram of Spectrogram | 9 |
| 4 | Diagram of Spectrogram Generated | 10 |
| 5 | Training and Validation loss | 11 |
| 6 | Snapshot of Result1 | 13 |
| 7 | Snapshot of Result2 | 13 |
| 8 | Snapshot of Result3 | 14 |
| 9 | Snapshot of Result4 | 14 |
| 10 | Snapshot of Result5 | 14 |

**Contents**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | Page No |
| Acknowledgement | | | | |  | i |
| List of Figures | | | | |  | iii |
|  | | | | | | |
|  | | | | **ABSTRACT** | | 1 |
|  | | | | | | |
| **Chapter 1 INTRODUCTION**  2 | | | | | | |
|  | | | | | | |
| **Chapter 2** | | | | **LITERATURE SURVEY** | |  |
|  | 2.1 | U-Net: Convolutional Networks for Biomedical Image Segmentation | | | | 7 |
|  | 2.2 | Singing voice separation with deep U-Net convolutional networks | | | | 8 |
|  | 2.3 | U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications | | | | 5 |
|  | | | | | | |
| **Chapter 3** | | | | **METHODOLOGY** | |  |
|  | 3.1 | U-Net Architecture | | | | 10 |
|  | 3.2 | Model workflow | | | | 12 |
|  | 3.3 | Short-Time Fourier Transform (STFT) | | | | 13 |
|  | 3.4 | Working Procedure | | | | 14 |
|  | | | | | | |
| **Chapter 4** | | | | **RESULTS AND CONC;USION** | | **17** |
|  | | | | | | |
|  | | | | | | |
|  | | | **REFERENCES** | | | 21 |
|  | | | | | | |

# Abstract

This project explores a deep learning approach to music genre classification using the Audio Spectrogram Transformer (AST), a transformer-based architecture originally pre-trained on the large-scale Audio Set for general audio classification. The primary goal is to leverage transfer learning to adapt this model to music genre classification with limited labelled data. By converting audio files into Mel-spectrograms—which capture the temporal and frequency-based features of sound in a format aligned with human auditory perception—the model can effectively learn genre-specific characteristics. To enhance performance and reduce overfitting, various data augmentation techniques such as time masking and frequency masking are applied during fine-tuning. This allows the model to generalize better across diverse and overlapping music genres. The proposed method demonstrates that pre-trained audio transformers, when fine-tuned with minimal data and supported by augmentation strategies, can achieve high accuracy in genre classification. The AST model’s ability to capture complex audio patterns makes it well-suited for this task, outperforming traditional CNN or RNN-based models in both efficiency and generalization. Overall, this project highlights the effectiveness of transformer-based models in advancing automated music analysis and supporting personalized music recommendation systems.

# Chapter 1 Introduction

The increasing demand for personalized user experiences in music streaming platforms has led to the development of systems that predict users' preferences and recommend music based on the tastes of others with similar profiles. A critical aspect of this recommendation system is music genre classification. Recent advances have focused on automating this task, as it plays a significant role in delivering accurate suggestions. However, supervised learning models for music genre classification face challenges due to the need for large, labelled datasets to handle the high diversity of audio samples.

To address this issue, techniques like self-supervised learning, transfer learning, and data augmentation have been proposed to reduce the size of the labelled data required for training, making the models more efficient. Among various audio representations, the Mel-spectrogram is a widely used feature, as it captures frequency information in a manner similar to human hearing over time. This makes it an ideal input for deep learning models aimed at classifying music genres.

In this project, we propose using an audio spectrogram transformer, which has been pre-trained on a large dataset of everyday sound classifications. We will leverage transfer learning and sound augmentation techniques to fine-tune the model on a smaller music genre dataset. Finally, the generalization performance of the trained model will be evaluated across different music genres, showcasing its ability to adapt to new data with minimal labelled examples.

# Chapter 2 Literature Survey

**2.1 Mel-spectrogram**

**Abstract:**

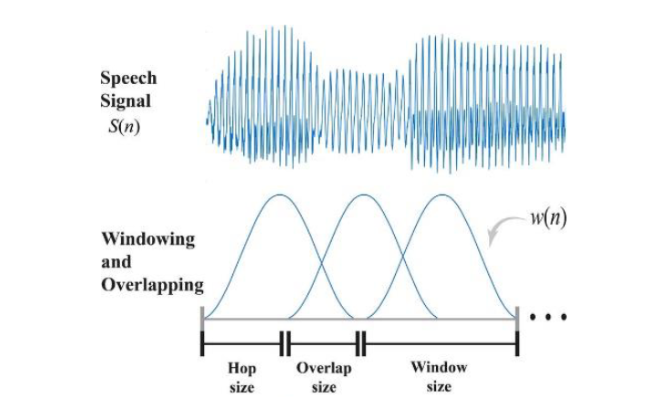
The Mel-spectrogram is a powerful audio representation that enables deep learning models to effectively interpret and classify audio signals, particularly for tasks like music genre classification. By simulating the way humans perceive sound frequencies, the Mel-spectrogram offers a perceptually meaningful way to visualize and analyse audio data over time. This technique is particularly useful in distinguishing between different genres, as each genre tends to exhibit unique frequency and temporal patterns.

**Methodology:**

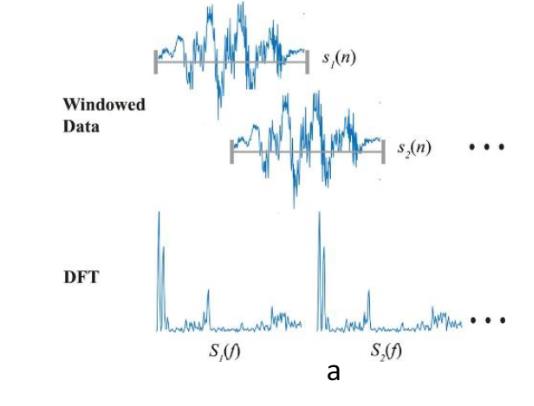
The process begins with applying the Short-Time Fourier Transform (STFT) to an audio signal. Unlike a standard Fourier Transform, STFT analyses overlapping short time intervals, preserving the temporal dynamics of the sound. For each window, a frequency vector is extracted, and by stacking these vectors in chronological order, a spectrogram is formed. However, since human hearing perceives frequency changes logarithmically rather than linearly, the frequency axis of the spectrogram is transformed using the Mel scale. This results in a Mel-spectrogram, where the frequency axis better aligns with human auditory perception. Only the magnitude (in decibels) of the frequency components is retained for analysis.

**Outcome:**

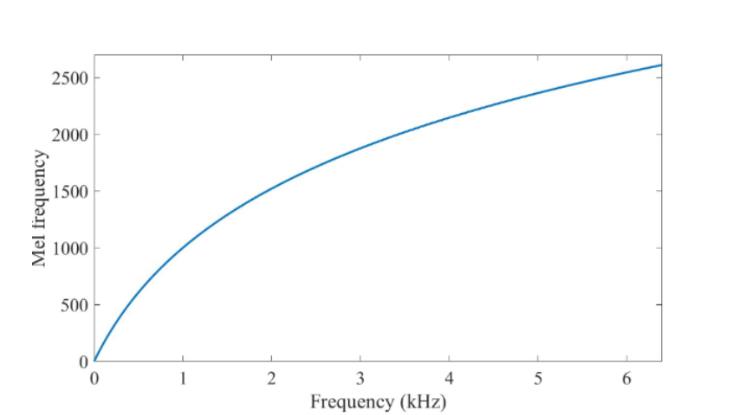
The resulting Mel-spectrograms vary significantly across genres. For example, reggae music, characterized by relaxed rhythms and lower-frequency components, shows sparse activity in the lower frequency range. In contrast, metal music, known for its aggressive and high-energy sound, results in denser and more complex spectrograms with activity spread across higher frequencies. These distinctive patterns make Mel-spectrograms highly suitable for training classification models to distinguish between music genres with improved accuracy and perceptual relevance.



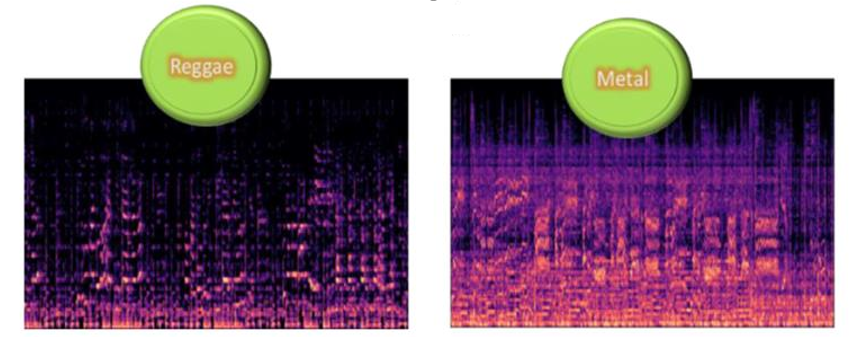
**Fig 1.** The image illustrates the windowing and overlapping process in STFT, a key step in generating Mel-spectrograms by converting time-domain speech signals into time-frequency representations.



**Fig 2**. The image illustrates how windowed segments of a signal are transformed using DFT to obtain their frequency components for time-frequency analysis.



**Fig 3**. This plot shows the nonlinear relationship between frequency in kHz and Mel frequency, highlighting how the Mel scale compresses higher frequencies to better match human auditory perception.



**Fig 4**. The image compares Mel-spectrograms of Reggae and Metal music, showing distinct frequency and energy patterns unique to each genre.

**2.2 AST: Audio spectrogram transformer**

**Abstract:**

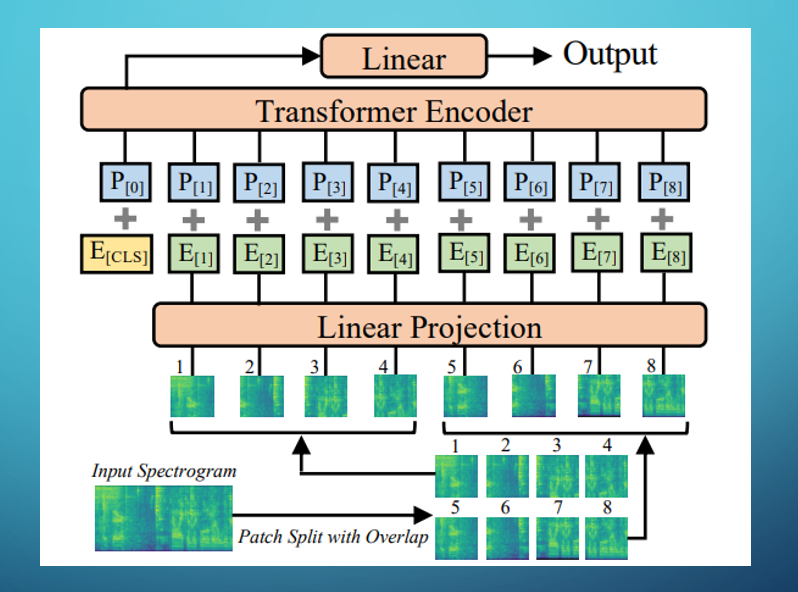
The Audio Spectrogram Transformer (AST), introduced by Yuan Gong et al., adapts the architecture of the Vision Transformer (ViT) for audio classification tasks. By treating audio spectrograms similarly to visual data, AST leverages the power of transformer-based models to learn rich, hierarchical representations of sound. The model’s design allows it to capture both temporal and frequency-based patterns from audio, making it highly effective for audio classification tasks, including music genre recognition.

**Methodology:**

The AST model processes an input Mel-spectrogram by first dividing it into 16×16 overlapping patches using a stride of 10. Each patch is flattened and projected into a 768-dimensional token. A learnable class token (CLS) is appended to the sequence of tokens, and learnable positional encodings are added to preserve spatial information. This sequence is then passed through a transformer encoder, which models complex inter-token dependencies. The final representation of the class token is projected to a score vector representing class probabilities. Initially, the model reuses the weights of a ViT pretrained on ImageNet (which processes 224×224 RGB images) and adapts them to audio input with spectrogram dimensions of 128×T (T being the number of time frames). The model is then fine-tuned on Audio Set, a large-scale dataset containing labelled audio clips from diverse sources such as music, speech, and environmental sounds.

**Outcome:**

By repurposing a ViT architecture and training it on audio data, AST achieves strong performance in sound classification tasks. Its ability to model both local and global patterns in spectrograms allows it to distinguish between complex audio classes with high accuracy. AST’s success demonstrates the versatility of transformer models in domains beyond vision and highlights the effectiveness of transfer learning from large visual datasets to audio tasks. This foundational architecture is now being leveraged in this project to classify music genres with minimal labelled data.



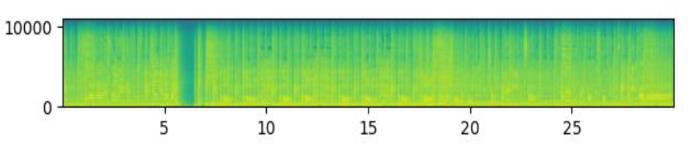
**Fig 5**. The Audio Spectrogram Transformer (AST) converts audio into spectrogram patches and uses a transformer to learn patterns for classification.

# Chapter 3 Methodology

**3.1 U-Net architecture**

One of the major challenges in training deep learning models on small datasets is the risk of overfitting, where the model learns to perform well on training data but fails to generalize to unseen samples. To mitigate this, we employed audio data augmentation strategies aimed at enriching the diversity of the training data without requiring additional labeled samples. These augmentations simulate real-world variations in music playback and recording conditions, ensuring that the model can better generalize to different acoustic environments and signal distortions.

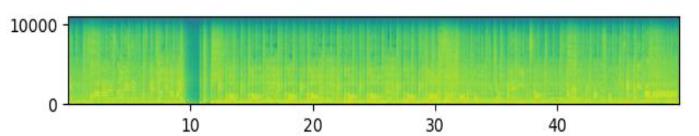
All augmentations were implemented using **TorchAudio’s AudioEffector**, which utilizes **FFmpeg** under the hood for efficient and reproducible audio manipulation. The augmentations were applied to every sample in the training set, with random parameterization to ensure variety. Below are the detailed descriptions of the applied augmentation techniques:



**Fig 6.** Mel spectrogram of a 30s track of the original sample

**Change of Pace (Tempo Variations)**

In real-world listening experiences, songs may be played at slightly faster or slower tempos without affecting their genre identity. To simulate such variability, we applied tempo modifications to the audio waveform. The pace of each training sample was randomly adjusted by a factor less than or greater than 1.0. For instance, a tempo value <1.0 resulted in a slow-paced version of the track, whereas a value >1.0 sped up the rhythm. This helps the model focus on genre-defining patterns like harmonic structure and timbre, rather than being misled by tempo-related artifacts.



**Fig 7.** Slow paced original sample. Tempo < 1

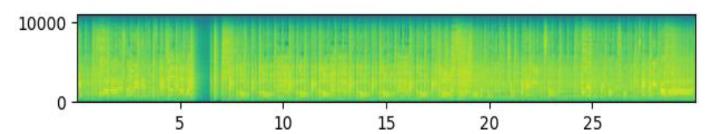
**Frequency Filtering (High-Pass and Low-Pass Filters)**

Frequency content plays a crucial role in distinguishing music genres, but real-world recordings often differ in how certain frequency ranges are emphasized or attenuated. To enhance the model’s robustness across different spectral distributions, we applied both high-pass and low-pass filters to the audio.

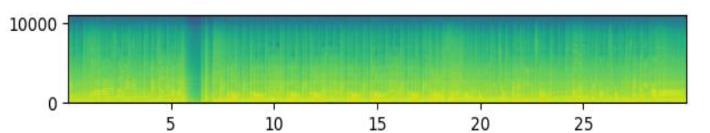
High-pass filters removed lower frequencies and emphasized higher tones, mimicking scenarios like distant or tinny recordings.

Low-pass filters suppressed high frequencies and accentuated bass, simulating environments where the music is muffled or heard through walls or barriers.

These filters force the model to learn genre features that persist even when parts of the frequency spectrum are altered or removed.



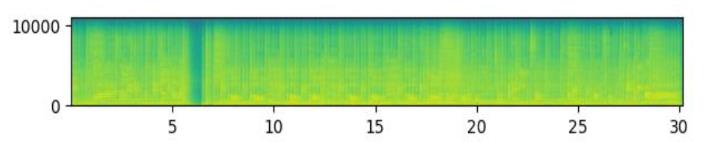
**Fig 8.** High-pass filter on the original sample



**Fig 9.** Low-pass filter on the original sample

**Echo Effect (Simulated Reverberation)**

To simulate the acoustic conditions of live recordings, open spaces, or concert halls, we applied an echo effect to selected audio samples. This augmentation adds delayed copies of the original signal to itself, creating a reverberant version of the track. The echo delay and decay factors were chosen randomly to create realistic variations. This prepares the model to handle tracks recorded in acoustically reflective environments and ensures it does not overfit to only clean, studio-quality audio.



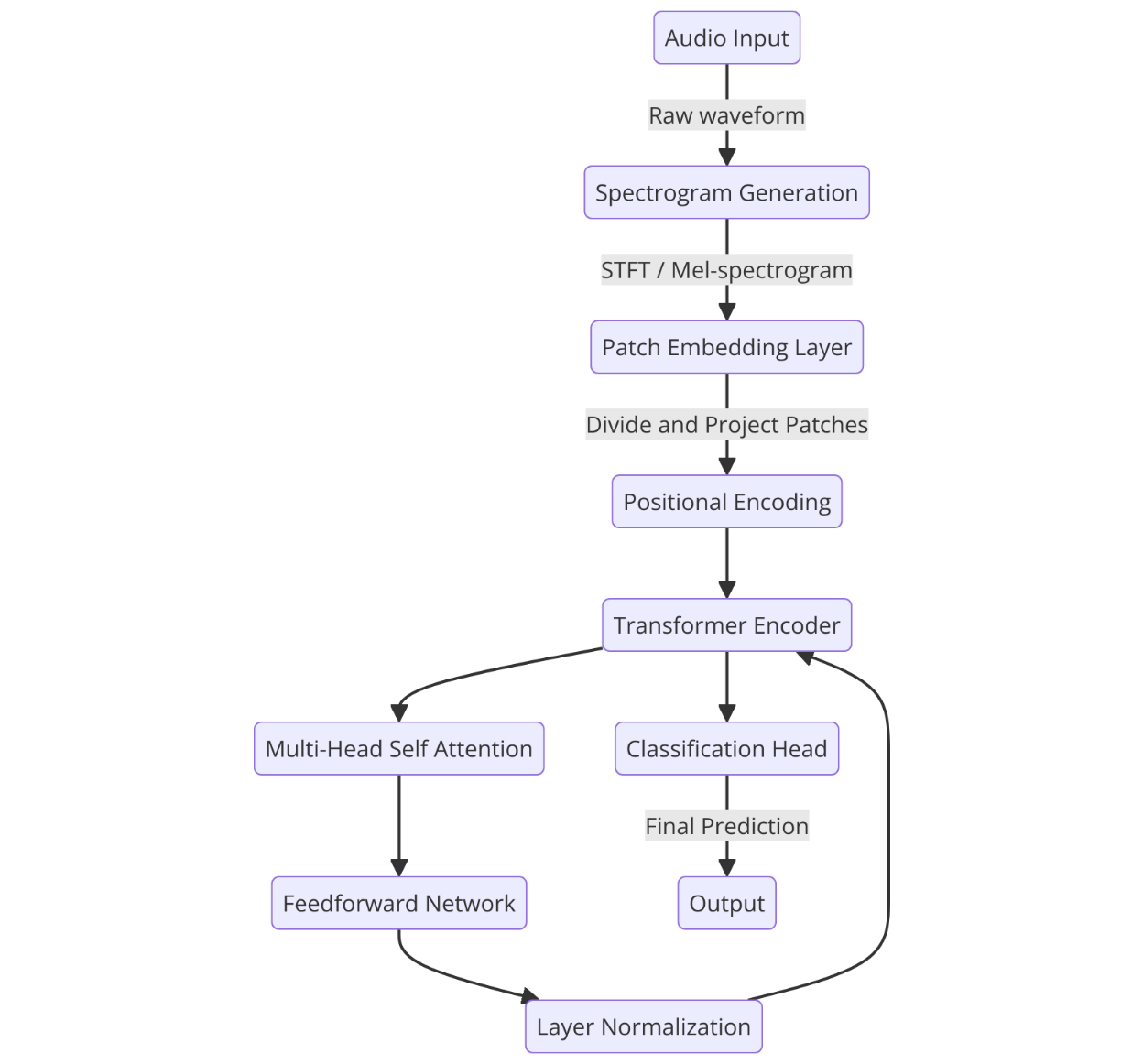
**Fig 10.** Echo effect applied on the original sample

**Bundle Effect (Combined Augmentations)**

To further enhance variability, we created a bundle augmentation by combining multiple effects—such as tempo changes, frequency filtering, and echo—on a single sample. Each augmentation was applied with randomly selected parameters, and in different sequences, to produce highly diverse variations. In cases where the augmentation resulted in a longer audio duration (e.g., due to echo or slowed tempo), we randomly cropped a fixed-length segment from the extended audio to ensure input consistency. This practice avoids introducing bias from fixed cropping positions.

After generating these augmented versions, we incorporated them into the original training set, effectively increasing its size and diversity. These transformations help the model develop a more invariant and generalized understanding of genre characteristics, improving its performance on real-world, noisy, and varied audio inputs.

## 3.2 Model Workflow



**Fig 11.** Diagram of Model workflow

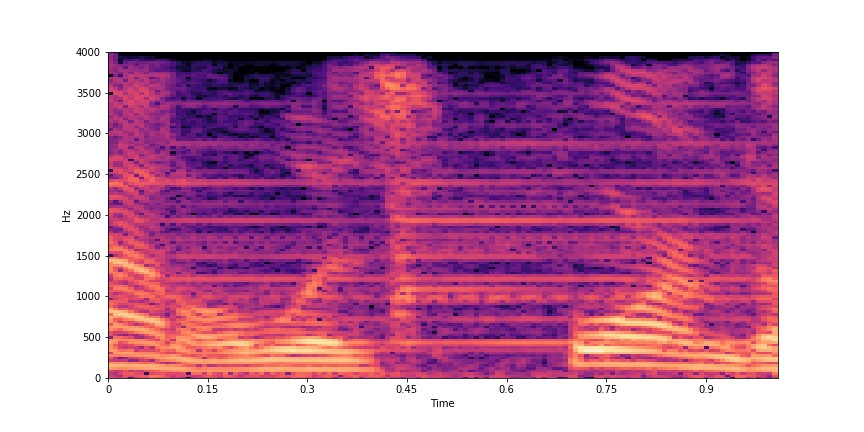
The diagram illustrates the workflow of an Audio Spectrogram Transformer (AST) model for audio classification tasks. The process begins with the audio input, which is first converted into a raw waveform. This waveform undergoes spectrogram generation, typically using Short-Time Fourier Transform (STFT) or Mel-spectrogram techniques, to convert the audio signal into a time-frequency representation. The resulting spectrogram is then divided into smaller patches, which are projected into a higher-dimensional space through a patch embedding layer. Positional encoding is added to these patches to retain the sequential order of the audio data. The encoded patches are fed into a transformer encoder, which processes the data using multi-head self-attention mechanisms and feedforward networks, with layer normalization applied to stabilize and accelerate training. The output from the transformer encoder is passed to a classification head that produces the final prediction, which is then output as the model’s result. This workflow leverages the transformer’s ability to capture long-range dependencies and contextual information in audio signals, making it highly effective for audio classification tasks

**3.3 Short-Time Fourier Transform (STFT)**

Short-Time Fourier Transform (STFT): This technique divides the signal into short overlapping segments and applies the Fourier transform to each segment. The result is a spectrogram, a two-dimensional representation with time on one axis, frequency on the other, and intensity (amplitude) represented by color or grayscale.

**What is a Spectrogram?**

To process and improve audio, we need a way to represent sound visually. One effective method is using spectrograms. Think of a spectrogram as a picture of sound over time



**Fig 12.** Diagram of Spectrogram

**Axes**: In a spectrogram, the horizontal axis represents time, and the vertical axis represents frequency (pitch).

**Brightness**: The brightness or intensity at any point on the spectrogram shows the strength (volume) of a particular frequency at a particular time.

## 3.4 Working Procedure

1. **Data Loading and Preprocessing**

**GTZAN Dataset:** The code uses the GTZAN music genre classification dataset containing 10 genres.

**Audio Transformations:**

Waveform Augmentations like pitch shifting, time stretching, and additive noise may be applied.

The audio waveform is converted into a log-Mel Spectrogram, which is a time-frequency representation suitable for audio transformers.

1. **Model Definition**

**AST (Audio Spectrogram Transformer):**

Based on the Vision Transformer (ViT) architecture.

Instead of image patches, it processes patches of the Mel spectrogram.

A pretrained AST model (possibly pretrained on AudioSet) is loaded.

The final layer is fine-tuned to predict 10 music genres (the number of classes in GTZAN).

1. **DataLoader and Batching**

The code sets up PyTorch DataLoaders for train, validation, and test splits.

It uses collate\_fn to pad audio inputs (since audio clips may vary in length).

Data is fed in batches to the GPU for training using DataLoader.

1. **GPU Parallelism**

If multiple GPUs are detected (e.g., two A100 GPUs), the model is wrapped in torch.nn.DataParallel or torch.nn.parallel.DistributedDataParallel.

This allows the training to be distributed across GPUs for faster computation.

**5. Training Loop**

For each epoch:

The model is set to train() mode.

Input spectrograms are forwarded through the AST model.

Loss is computed using CrossEntropyLoss.

Backpropagation is performed.

Optimizer (e.g., Adam) updates the model weights.

Training accuracy and loss are logged.

**6. Validation Loop**

After each epoch, the model is evaluated on the validation set using eval() mode.

Accuracy and loss are measured to monitor overfitting or underfitting.

The best-performing model (on the validation set) is saved.

**7. Testing and Evaluation**

The final model is tested on the held-out test set.

Performance metrics such as:

Accuracy

Confusion Matrix

Precision/Recall/F1-score

Possibly AUC or ROC curves (optional)

These help analyze how well the model generalizes.

**8. Logging and Saving**

Model checkpoints are saved during training.

Logs (losses, accuracies) can be visualized using TensorBoard or Matplotlib.

The final model and logs are saved for reproducibility.

**Prediction:**

1. **Environment and Model Setup**

* Imports necessary libraries: PyTorch, Transformers, HuggingFace datasets, NumPy, etc.
* The device (CPU or GPU) is set based on availability.
* Labels (genres) are extracted from the folder structure of the training dataset (/genres) for mapping predictions.
* Model configuration is loaded from the pretrained AST model (MIT/ast-finetuned-audioset10-10-0.4593) and updated for the number of genres (10 in GTZAN).
* Model-specific configurations like classifier head size and dropout rate are applied.
* The model weights are loaded from the fine-tuned checkpoint
* (Best\_Model/pytorch\_model.bin) and moved to the appropriate device for inference.

1. **Audio Preprocessing**

* A directory containing music files (jamendo\_music\_samples) is scanned, and each file is processed one by one.
* The datasets.Dataset and Hugging Face's Audio class are used to:
* Load the audio file.
* Resample the audio to match the expected sampling\_rate of the model (e.g., 16kHz).
* Each song is assumed to be at least 30 seconds long. If not, a warning is printed.

1. **Chunk Sampling Strategy**

* From each full-length song:
* 20 random 30-second chunks are sampled from the song's waveform.
* Each chunk is passed through the feature extractor (AutoFeatureExtractor) which converts it to a log-Mel spectrogram, the format required by the AST model.

1. **Genre Prediction**

* The preprocessed audio chunks are passed through the model in evaluation mode (model.eval()).
* For each chunk:
* The model outputs logits (raw prediction scores).
* The class with the highest score is selected using argmax.
* All 20 chunks give predictions, and majority voting is used to find the final predicted genre of the song.
* The model also calculates the average confidence score (softmax probability) of the predicted genre across the chunks that voted for the majority class.

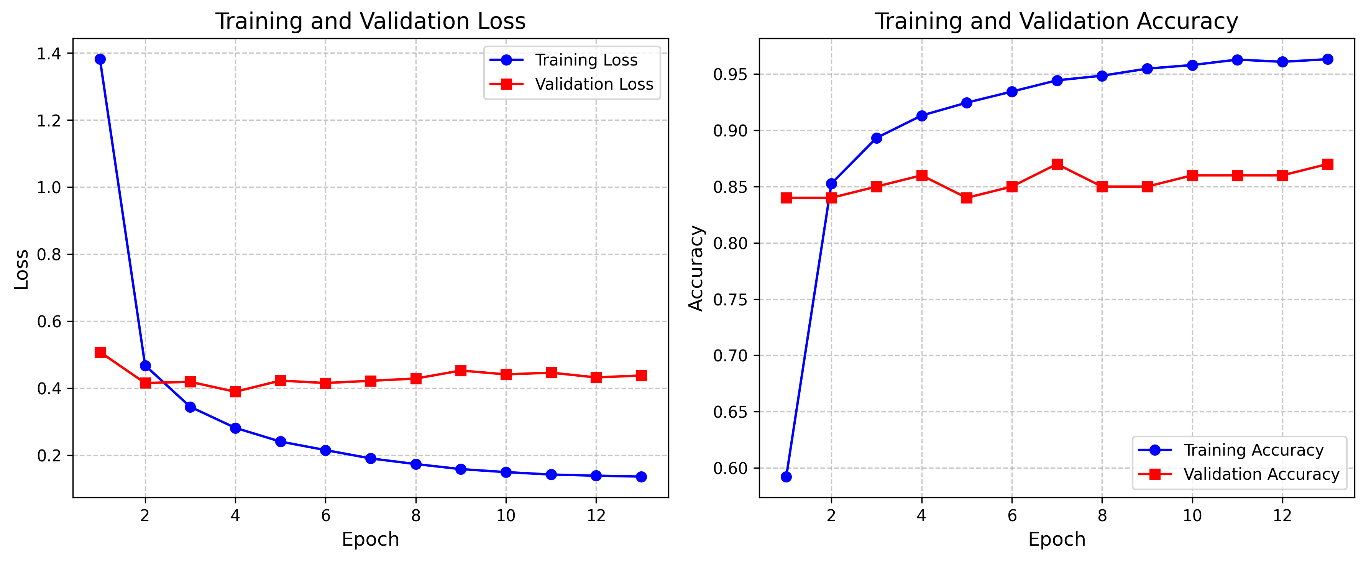
1. **Output**

For each song:

The predicted genre (e.g., “rock”, “jazz”, etc.)

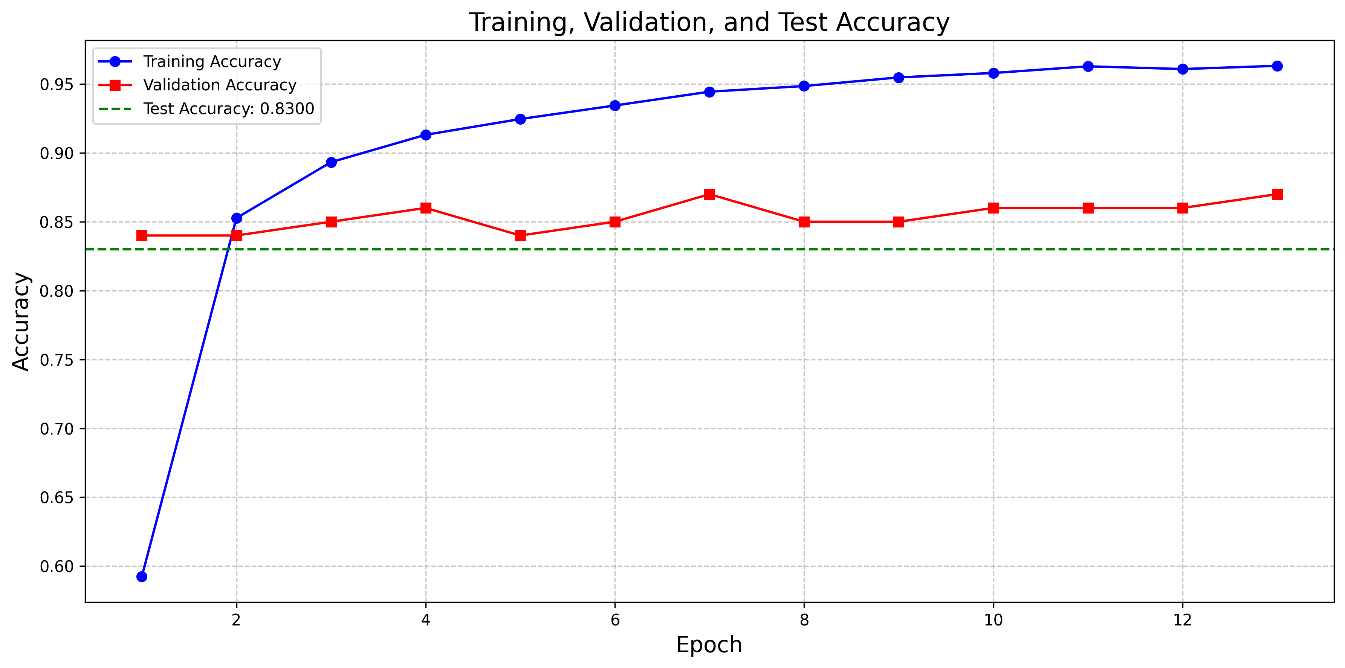
The average confidence score of the prediction is displayed.

# Chapter 4 Results and Conclusion

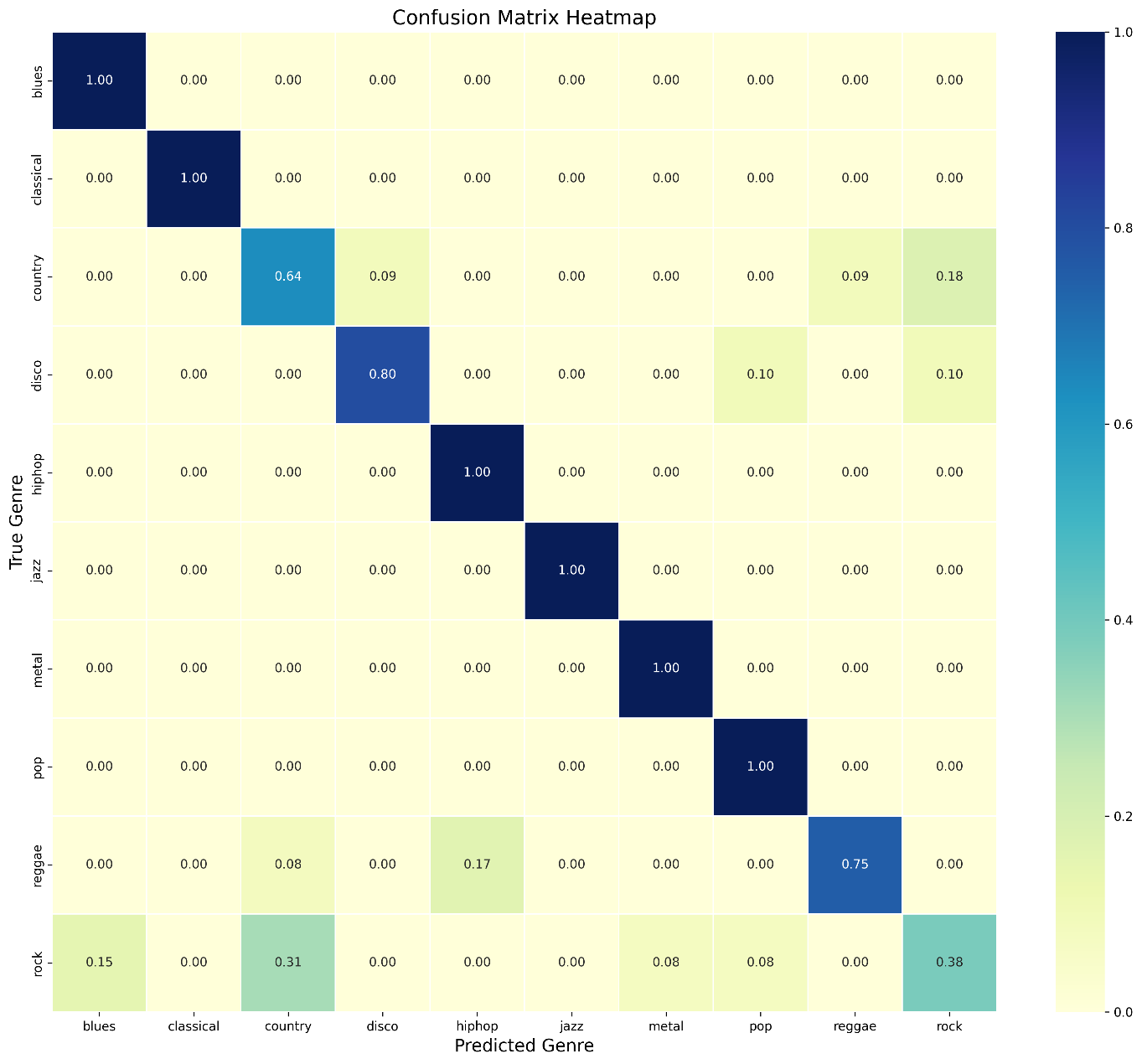


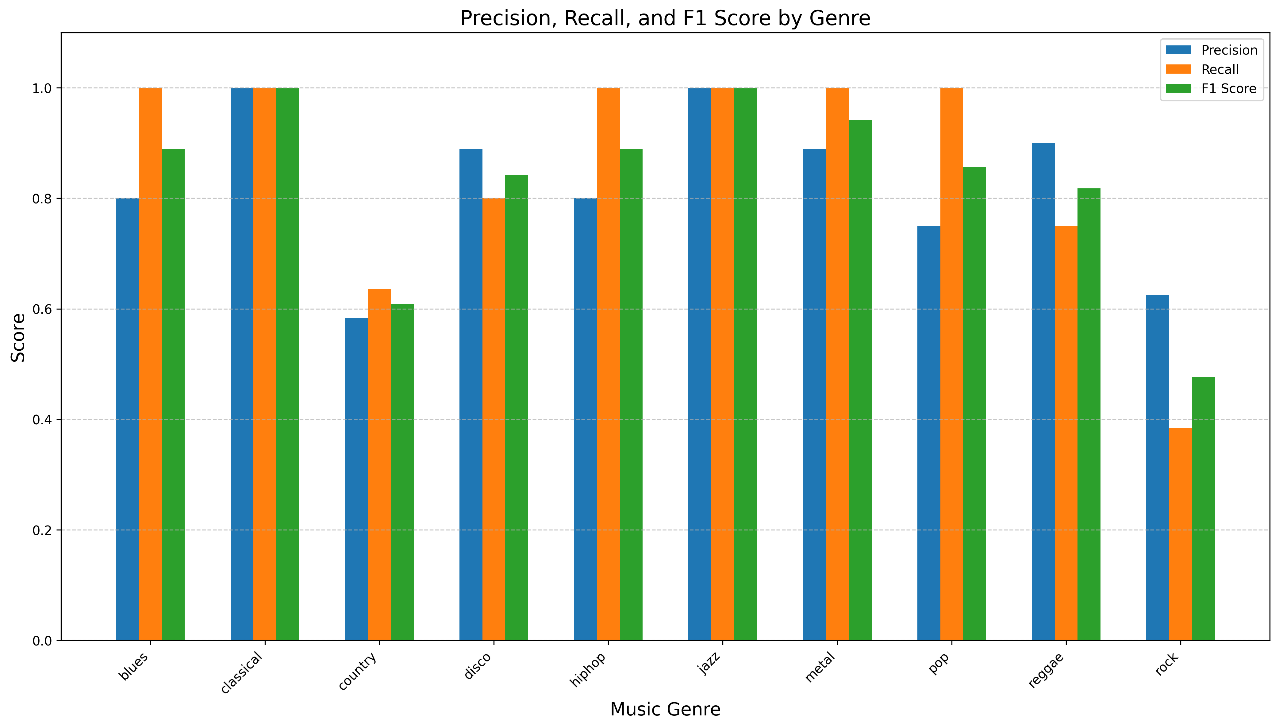
**Fig 12. Training**

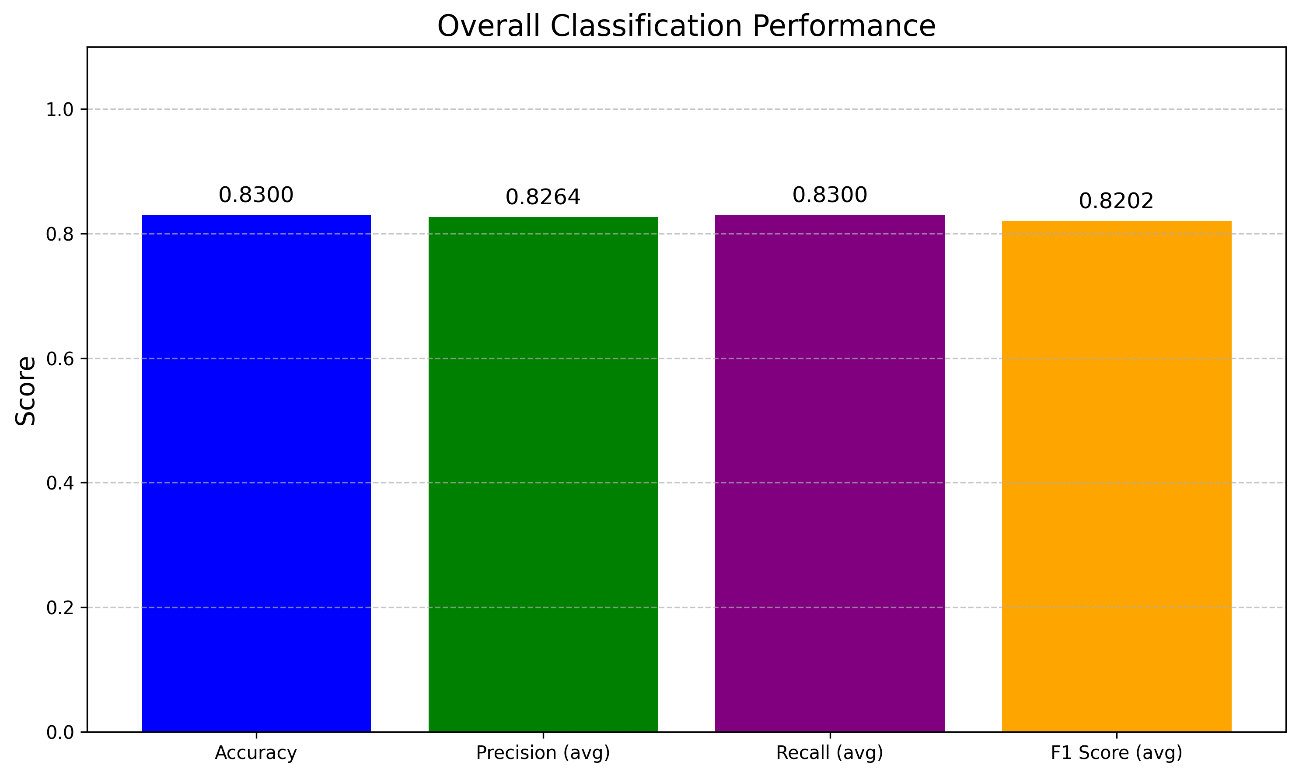
Final Test Loss: 0.3791, Accuracy: 0.8300

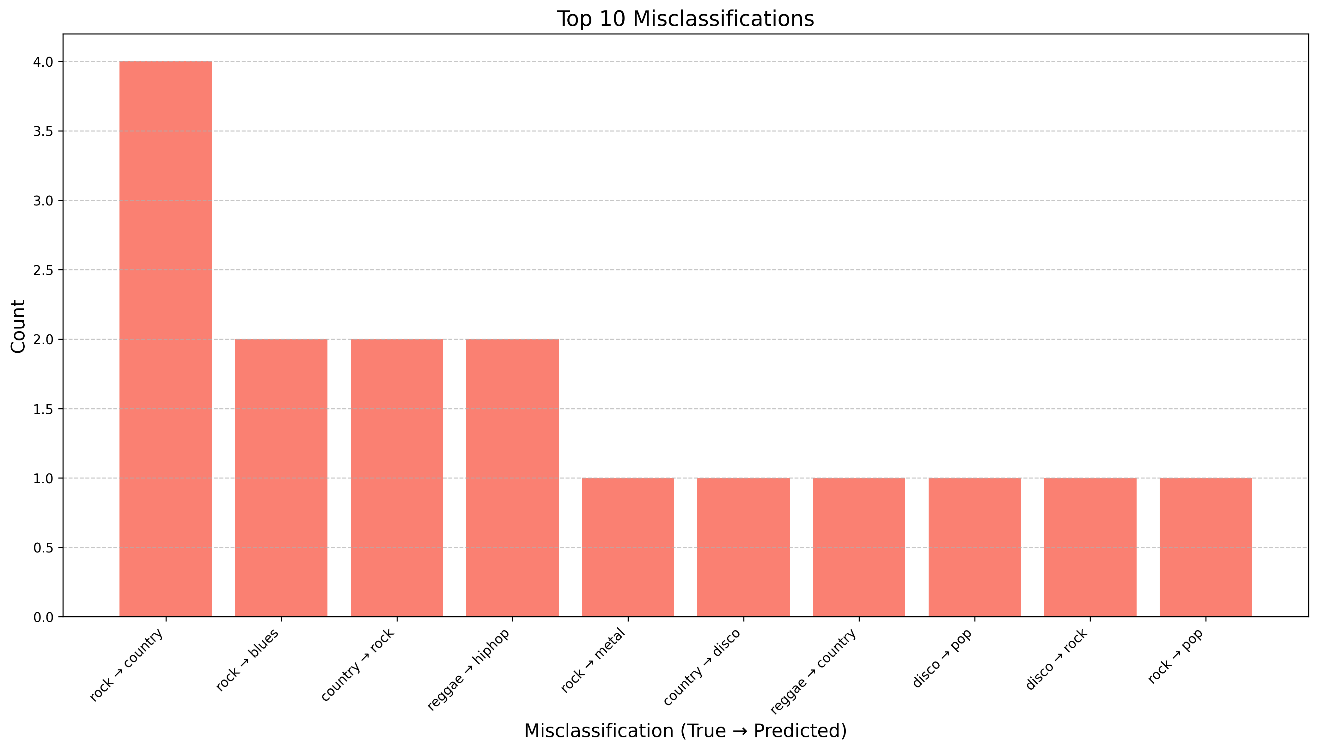
****

**accuracy\_curve**

****

****

****

****

# References

1. Yuan Gong, Yu-An Chung, James Glass, "AST: Audio Spectrogram Transformer," 2021.
2. Rodrigo Castellon,Chris Donahue,Percy Liang, "Codified Audio Language Modeling Learns Useful Representations For Music Information Retrieval," in ISMIR, 2021.
3. Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino, "Masked Modeling Duo: Learning Representations," in ICASSP, 2023.
4. Yuval Hoffman and Roee Hadar, "Music Genre Classification using Transformers," [Online]. Available: https://github.com/YuvalHoffman/Music-Genre-Classification using-Transformers.
5. Omer Cohen and Jonathan Nir Shalit, "MusicGenreClassifier," [Online]. Available: https://github.com/omercohen7640/MusicGenreClassifier.
6. Ilay Yavlovich and Amit Karp, "Automatic Music Genre Classification," [Online]. Available: https://github.com/IlayYavlovich/MusicGenreClassifier-Technion.
7. Itai Allouche and Adam Katav, "MusicGenreClassification," [Online]. Available: https://github.com/ItaiAllouche/MusicGenreClassification.
8. Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng and Min-Hung Chen, "DoRA: Weight-Decomposed Low-Rank Adaptation," in ICML, 2024.
9. George Tzanetakis, Georg Essl, Perry Cook, "Automatic Musical Genre Classification Of Audio Signals," in ISMIR, 2001.
10. Lewis Tunstall, "Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals)," [Online]. Available: https://huggingface.co/datasets/marsyas/gtzan.