## ****Troy Sabia****

## ****Executive Summary: Automated Metal Genre Classification for Music Streaming****

### Problem Statement

A startup streaming service targeting metal music fans will require a system to automatically classify incoming audio content as either "metal" or "non-metal." Manual tagging is error-prone and does not scale with a large or constantly growing music catalog. Furthermore, genre metadata from third-party sources is often inconsistent or missing. To maintain genre purity and avoid user churn, a programmatic solution will be essential.

### Customer Summary

The primary customer is a startup building a niche streaming platform focused exclusively on metal music. Their value proposition is genre-specific curation and authenticity. Their technical infrastructure is not yet finalized, but they will need backend tools that can be integrated into ingestion pipelines or run as batch preprocessing jobs. The customer expects this tool to handle hundreds to thousands of music files automatically and to be expandable for more refined classification in the future (e.g., subgenres).

### Existing System Analysis

Currently, no automated system exists within the organization. Manual vetting is used or outsourced, which is not viable long term. While third-party metadata services exist, they rely on subjective tagging and lack audio analysis capabilities. No open-source or commercial solution is known to provide reliable genre classification specifically tuned for metal. This project will serve as the first purpose-built tool in the company’s toolchain for genre enforcement.

### Data

The training dataset will consist of two labeled classes: “metal” and “non-metal.” Source material will include the GTZAN dataset (30-second clips across 10 genres) and publicly available MP3s curated from metal and non-metal collections.

The raw audio will be resampled to 22.05 kHz mono and clipped to 15 seconds. Mel Frequency Cepstral Coefficients (MFCCs) with 16 channels will be extracted using a 512 FFT size and 128 hop length. Only these MFCC tensors (not raw audio) will be used for training. All data will be stored locally in .pt files with labels, paths, and padding to uniform time dimensions for batch training.

### Project Methodology

The solution will be built using PyTorch. The pipeline will consist of:

1. **Preprocessing**: Convert raw WAVs to MFCC tensors using a consistent configuration.
2. **Dataset Handling**: Pad MFCCs to a consistent temporal shape; store them with labels in a structured cache directory.
3. **Model Design**: A CNN with three convolutional blocks (non-square kernels) and a classifier head with dropout, batch normalization, and LeakyReLU activation.
4. **Training**: Use BCEWithLogitsLoss with class balancing and StepLR for scheduling. Track performance on validation and test sets.
5. **Evaluation**: Assess training and validation loss and iteratively select the best model for deployment.

### Project Outcomes

* A trained binary classifier that accepts a 15-second clip and returns a metal vs. non-metal prediction.
* Evaluation metrics that exceed 90% accuracy and demonstrate generalization across unseen data.
* A reusable pipeline to expand to multi-class subgenre classification in the future.
* Batch classification support for new audio ingestion.

### Implementation Plan

The model will be trained locally or on a GPU-enabled cloud instance using PyTorch. Once trained, it will be saved to disk along with test results. An inference script will allow batch classification of new files. The tool will be packaged into a CLI utility that can be run on Linux servers or integrated into a larger ETL system. No external services are required at this stage.

### Evaluation Plan

The model’s success will be evaluated using:

* Accuracy, precision, recall, and F1 score on a held-out test set
* ROC AUC score and curve visualization
* Consistency between validation and test metrics (≤5% difference)
* Manual sampling of borderline cases to understand edge behavior

### Resources and Costs

| Resource | Description | Cost Estimate |
| --- | --- | --- |
| GPU Rental | Cloud GPU (e.g., AWS p3/Google Colab Pro) | $300 |
| Developer Labor | 10 hours @ $120/hr equivalent | $1,200 |
| Miscellaneous Storage | Cloud object storage (optional) | $0–$50 |
| **Total** |  | **$1,500** |

No proprietary software licenses or API fees are expected. All software dependencies are open-source (PyTorch, torchaudio, scikit-learn, matplotlib).

### Timeline and Milestones

| Week | Milestone |
| --- | --- |
| 1 | Finalize dataset and preprocessing pipeline |
| 2 | Train initial model; perform error analysis |
| 3 | Tune architecture and scheduler; retrain |
| 4 | Evaluate model; package tool; write report |