August 28, 2025

C964: Audio Genre Classification

Troy Sabia

Part A: Letter of Transmittal

Letter of Transmittal Requirements

The Letter of Transmittal should convince senior leadership to approve your project. Write a brief cover letter (suggested length 1-2 pages) describing the problem, how the application (part C) applies to the problem, the practical benefits to the organization, and a brief implementation plan. Include all artifacts typical of a professional (business) letter, e.g., subject line, date, greeting, signature, etc.

The letter should be concise and target a non-technical audience. Include the following:

- *A summary of the problem.*
- *A proposed solution centering around your application.*
- *How the proposed solution benefits the organization.*
- *A summary of the costs, timeline, data, and any ethical concerns (if relevant).*
- *Your relevant expertise.*

08/27/2025
Mr. Brendon Small
Explosive Media
94 Leighton Rd
Yarmouth ME 04096

tsabia1@wgu.edu
206-666-6699
1 Herbert Dr
Scarborough ME 04074
August 27, 2025

Subject: Proposal for Automated Metal Genre Classification

Dear Mr Brendon Small,

Metal fans are loyal, but they expect authenticity. One of the greatest risks for a streaming platform built around this genre is delivering content that doesn't truly belong. Manual curation and third-party tagging are inconsistent and simply cannot keep up as the catalog grows.

I propose an automated system that programmatically classifies songs as either *metal* or *non-metal*. By using short clips of audio and a trained neural network, the tool ensures that only the right music makes it onto the platform. Early testing shows accuracy above 94%, which more than meets our threshold for reliability.

The benefits are clear: faster catalog growth without the bottleneck of manual review, reduced operational costs, and a stronger brand promise to your audience: "only the heaviest." Looking forward, the same system can be expanded to include subgenres, creating even richer curation opportunities.

The estimated cost is $1,500, covering development and training. With my background in computer science, machine learning, and audio technology, I am confident I can deliver this solution efficiently and position the platform for success.

Thank you for your consideration. I look forward to your approval to move forward.

Sincerely,

Troy Sabia

Part B: Project Proposal Plan

The project proposal should target your client's middle management. This audience may be IT professionals but have limited computer science expertise. Use appropriate industry jargon and sufficient technical details to describe the proposed project and its application. Remember, you're establishing the technical context for your project and how it will be implemented for the client. ***Write everything in the future tense.***

Project Summary

- *Describe the problem.*
- *Summarize the client and their needs as related to the problem.*
- *Provide descriptions of all deliverables. For example, the finished application and a user guide.*
- *Provide a summary justifying how the application will benefit the client.*

Project Proposal: Automated Metal Genre Classification for Explosive Media

Explosive Media is launching a streaming service with a single promise: every track will be metal. However, this is difficult to guarantee with current methods, as manual curation is slow and expensive, and outside metadata is often unreliable. Without a reliable system, non-metal songs could slip in, damaging both user trust and the brand's credibility.

This project will solve this problem by delivering an automated classification system to screen each new audio file. The application will process short clips of music, extract their acoustic features, and use a trained neural network to decide if a track is metal. This lightweight tool will run on standard infrastructure, require minimal staff training, and provide clear evaluation metrics, ensuring the accuracy that Explosive Media's leadership needs.

Deliverables include the finished application, a simple user guide, and performance reports. The model has already achieved over 94% accuracy in testing, exceeding the 90% target threshold. This not only keeps non-metal tracks out of the catalog but also reduces manual review costs and speeds up the ingestion of new music. The project is estimated to cost just $1,500, covering development and GPU training time, with no additional licensing or infrastructure expenses. This modest investment will give Explosive Media a scalable, dependable solution that protects its brand and positions the platform for future growth, including the option to classify sub-genres.

Data Summary

- *Provide the source of the raw data, how the data will be collected, or how it will be simulated.*
- *Describe how data will be processed and managed throughout the application development life cycle: design, development, maintenance, etc.*
- *Justify why the data meets the needs of the project. If relevant, describe how data anomalies, e.g., outliers, incomplete data, etc., will be handled.*
- *Address any ethical or legal concerns regarding the data. If there are no concerns, explain why.*

For this project, we'll use commercial audio files from curated music archives, supplemented by academic datasets like GTZAN. From these files, we'll extract short clips, standardize them to mono, resample them, and convert them into Mel Frequency Cepstral Coefficients (MFCCs). These MFCCs will serve as our working dataset, capturing the frequency and timing patterns that distinguish metal from non-metal music. We estimate collecting around 1,500 tracks to ensure sufficient diversity across subgenres. This variety is crucial because metal has many forms, and the model must be exposed to this range to perform reliably. We'll automatically exclude any anomalous or corrupted files during preprocessing, guaranteeing that only consistent, usable features are stored.

The dataset will be split into training, validation, and test sets to ensure an unbiased evaluation. We will use feature caching to make retraining efficient as new data is added. Since we're only storing non-reversible MFCC features, there are no ethical or legal concerns with data use during development. Explosive Media will review licensing requirements before eventual deployment, but for now, this approach is safe, ethical, and follows proper research standards. This data strategy creates a solid foundation for building an accurate, scalable classifier that supports Explosive Media's goal of delivering a trustworthy music catalog.

Implementation

- *Describe an industry-standard methodology to be used.*
- *An outline of the project's implementation plan. The focus can be the project's development or the implementation of the machine learning solution.*

This project will follow the CRISP-DM methodology, an industry standard for data mining and machine learning initiatives. This iterative cycle starts with understanding the business problem and moves through data preparation, model design, training, and evaluation before returning for refinement. By using CRISP-DM, we ensure the classifier is not only technically sound but also aligned with Explosive Media's business goals of authenticity and efficiency.

Implementation will begin with data preparation, where raw music files are standardized and transformed into Mel Frequency Cepstral Coefficients (MFCCs). These features will serve as the neural network's input, allowing it to learn the acoustic patterns that distinguish metal from non-metal music. Once the dataset is ready, we'll train the convolutional neural network (CNN) using supervised learning. We'll track performance across validation and test sets and use regularization methods like dropout and learning-rate scheduling to maintain stability and improve generalization. This ensures the system performs well on both training data and unseen tracks.

After training, the best-performing model will be packaged into a lightweight application. This application will be tested against a held-out test set to confirm an accuracy above 90%. We'll document key evaluation metrics like precision, recall, and F1 score for transparency. The final deliverables will include the working software and a user guide for IT staff. The tool can be deployed as a standalone batch process or integrated into Explosive Media's ingestion pipeline. This repeatable implementation plan allows for retraining and improving the classifier as new datasets become available.

Timeline

- *Provide a projected timeline. Include each milestone and deliverable, its dependencies, resources, start and end dates, and duration. (a table is not required but encouraged).*
- *Dates should be in the future. Write 'NA' where an item is not applicable.*

Milestone or deliverable	**Project Dependencies**	**Resources**	**Start and End Date**	**Duration**
Collect and preprocess raw audio into MFCC cache dataset	Access to internet, storage	CPU, storage	September 15 – September 21	1 week

Train baseline model, provide initial metrics	Completed dataset	GPU, Python/Pytorch development environment	September 22 – September 28	1 week
Optimize model, validate improved metrics	Baseline training results	GPU, development environment	September 29 – October 6	1 week
Final testing, documentation, delivery	Optimized training model	CPU, GPU, development and production environment	October 7 – October 13	1 week

The first week will focus on finalizing the dataset. We'll collect raw audio files, organize them into training, validation, and test sets, and process them into MFCC features. This will result in a standardized feature cache, ensuring the data is consistent and ready for model training.

Weeks two and three will be dedicated to training and refining the classifier. An initial training run will establish baseline performance, and subsequent optimization will aim to surpass the 90% accuracy threshold.

By the fourth week, the final model will be tested against the held-out test set. We'll document its performance in an evaluation report and package the model as a deployable application with a user guide. At this point, Explosive Media will have a ready-to-use solution for automated genre classification.

Evaluation Plan

- *Describe the verification method(s) to be used at each stage of development.*
- *Describe the validation method to be used upon completion of the project.*

During data preparation, automated checks will be built into the preprocessing pipeline. These checks will verify that every audio file has been successfully transcoded, clipped, and converted into MFCC features of a consistent shape. Any corrupted files or clips of unusual length will be excluded, ensuring that only valid and uniform data is used for training.

During training, we'll monitor the model's performance against the validation set. Metrics such as training loss, validation loss, and accuracy will be tracked after each epoch. We will also use precision, recall, and F1 score to gain a deeper understanding of how well the classifier is learning. We can use diagnostic tools like learning curves and confusion matrices to help identify overfitting, underfitting, or class imbalances. This continuous evaluation ensures the model is improving during training and can generalize to unseen data.

Upon completion of development, we'll validate the model using a held-out test set. This dataset provides an unbiased measure of performance since it's entirely separate from the training and validation data. Final metrics will include accuracy, precision, recall, F1 score, ROC-AUC, and a confusion matrix. Together, these metrics will confirm whether the model meets the required 90% accuracy threshold. We'll also manually review borderline cases to ensure the system behaves as expected in edge scenarios. The final results will be documented in a report, giving Explosive Media a transparent and trustworthy assessment of the classifier's effectiveness before deployment.

Costs

Include the itemized costs of the project. Include specific item names where applicable, e.g., 'PyCharm Professional Ed. 2024.3.5.'

- *Itemize hardware and software costs.*
- *Itemize estimated labor time and costs.*
- *Itemize estimated environment costs of the application, e.g., deployment, hosting, maintenance, etc.*

This project will have minimal costs compared to most large-scale IT initiatives, thanks to its use of open-source software and lightweight infrastructure.

All primary development will use freely available, open-source tools like Python 3.13.3, PyTorch 2.x, and torchaudio, so no licensing fees are required. Other necessary libraries, such as scikit-learn and matplotlib, are also open-source and free. We'll use standard IDEs for code editing, and while optional paid tools like PyCharm are available, they are not necessary. Hardware requirements are also modest, with local development possible on standard machines and cloud GPU rentals providing accelerated training.

The bulk of the project's expense comes from labor, which is estimated at ten hours of dedicated development and testing at a rate of $120 per hour, for a total of $1,200. This covers dataset preparation, model design, training, evaluation, and final packaging with documentation. The project's focused scope helps avoid unnecessary overhead while still delivering a production-ready proof of concept. The remaining cost is for GPU rental fees, estimated at $300, to cover multiple training and optimization runs. The total projected cost is $1,500, a modest investment for a high-value, scalable solution that protects the company's brand and streamlines its content ingestion process.

Item	Description	Cost Estimate
Labor	10 hours @ $120/hr	$1,200
GPU Rental	Cloud GPU (AWS p3 or Google Colab Pro)	$300
Software	Python 3.13.3, PyTorch 2.x, torchaudio, scikit-learn, matplotlib (all open-source)	$0
IDE	Visual Studio Code	$0
Deployment/Hosting/ Maintenance	Use of existing Explosive Media infrastructure	$0
Total		**$1,500**

Part C: Application

Part C is your submitted application. This part of the document can be left blank or used to include a list of any submitted files or links.

The minimal requirements of the submitted application are as follows:

1. ***The application functions as described.*** *Following the 'User Guide' in part D, the evaluator must be able to review your application on a Windows 10 machine successfully.*
2. ***A mathematical algorithm applied to data,*** *e.g., supervised, unsupervised, or reinforced machine learning method.*
3. ***A "user interface."*** *Following the 'User Guide' in part D, the client must be able to use the application to solve the proposed problem (as described in parts A, B, and D). For example, the client can input variables, and the application outputs a prediction.*
4. ***Three visualizations.*** *The visualizations can be included separately when including them in the application is not ideal or possible; e.g., the visualizations describe proprietary data, but the application is customer-facing.*
5. ***Submitted files and links are static and accessible.*** *All data, source code, and links must be accessible to evaluators on a Windows 10 machine. If parts of the project can be modified after submission, matching source files must be submitted. For example, if the application is a website or hosted notebook, the `.html` or `.ipynb` files must be submitted directly to assessments.*

Ideally, submitted applications should be reviewable using either Windows or Mac OS, e.g., Jupyter notebooks, webpages, Python projects, etc. If the source files exceed the 200 MB limit, consider providing screenshots or a Panopto video of the functioning application and contact your course instructor.

Part D: Post-implementation Report

Create a post-implementation as outlined below. Provide sufficient detail so that a reader knowledgeable in computer science but unfamiliar with your project can understand what you have accomplished. Using examples and visualizations (including screenshots) beyond the three required is recommended (but not required). ***Write everything in the past tense.***

Solution Summary

- *Summarize the problem and solution.*
- *Describe how the application solves the problem from parts A and B.*

This project successfully solved the challenge of reliably separating metal from non-metal tracks for a niche streaming service. Manual tagging and third-party metadata were found to be inconsistent, risking off-brand content and user dissatisfaction.

We built an automated classification tool that converts audio files into short clips, extracts MFCCs, and uses a convolutional neural network to distinguish between metal and non-metal music. The application consistently exceeded the target of 90% accuracy, achieving over 94% on the test set.

The final system met the project's original business requirements and provides a direct solution to the identified problem. By using this lightweight, script-based workflow, Explosive Media can now automatically vet large batches of songs. This not only ensures genre integrity and reduces the cost of manual review but also strengthens the brand's promise of delivering "only the heaviest" music.

Data Summary

- *Provide the source of the raw data, how the data was collected, or how it was simulated.*
- *Describe how data was processed and managed throughout the application development life cycle: design, development, maintenance, etc.*

The dataset combined the GTZAN benchmark collection with commercially purchased music files to ensure both academic rigor and real-world variety. In total, more than 1,500 songs were collected in MP3 format, spanning both metal and non-metal categories. From these, we extracted 15-second clips at multiple points within each track, yielding 4,792 usable samples.

All files were standardized to 22.05 kHz mono WAV format before feature extraction. MFCCs were computed using consistent parameters (16 channels, 512 FFT bins, and hop length of 128), then padded to uniform length for batch training. These features were cached in .pt files alongside their labels and metadata, ensuring reproducibility and fast iteration during development.

This approach created a balanced, high-quality dataset suitable for training and validating the model. Data anomalies, such as corrupted or incomplete files, were automatically excluded during preprocessing, keeping the dataset consistent. Because only MFCCs were stored and not raw audio, there were no ethical or legal concerns during development.

Machine Learning

For each machine learning model (at least one is required), provide the following:

- *Identify the method and what it does (the "what"). It's advisable to include an example of the model's output.*
- *Describe how the method was developed (the "how").*
- *Justify the selection and development of the method (the "why").*

This project used a convolutional neural network and supervised learning to classify audio clips as either metal or non-metal. The CNN analyzed 15-second clips converted into MFCC tensors and processed through three convolutional layers with batch normalization, pooling, and dropout. A classifier head then produced a single score between 0 and 1, indicating the probability that the clip was metal.

The model was developed in stages. An initial baseline model, with around 1,500 parameters, achieved a solid 88% accuracy on the test set. To address overfitting and improve performance, we optimized the architecture by increasing the number of parameters, adding a linear layer to the classifier, and increasing dropout in deeper layers. These changes improved generalization and stability. The final, optimized network contained approximately 25,000 parameters and achieved over 94% accuracy with balanced precision and recall.

A CNN was chosen because it's highly effective at processing time-frequency data like MFCCs, where local patterns in frequency bands are key to genre characteristics. While other approaches like recurrent networks were considered, they were deemed unnecessary for this binary classification task. The CNN provided a balance of high accuracy and efficient inference, making it the best choice for production.

Validation

For each machine learning algorithm described in the section above, do the following:

- *Identify the model's machine learning category, e.g., supervised, unsupervised, or reinforced. For blended approaches, identify the category most relevant to the model's application.*
- *An appropriate validation method for the model's performance.*

For supervised learning and reinforced learning

- *Describe an appropriate metric(s) for testing the model's performance.*
- *Provide results of testing using the described metric.*

The classifier was built as a supervised learning model, trained with labeled examples of metal and non-metal audio. We monitored performance during training using a validation set, and the final evaluation was conducted on a separate test set to ensure unbiased results.

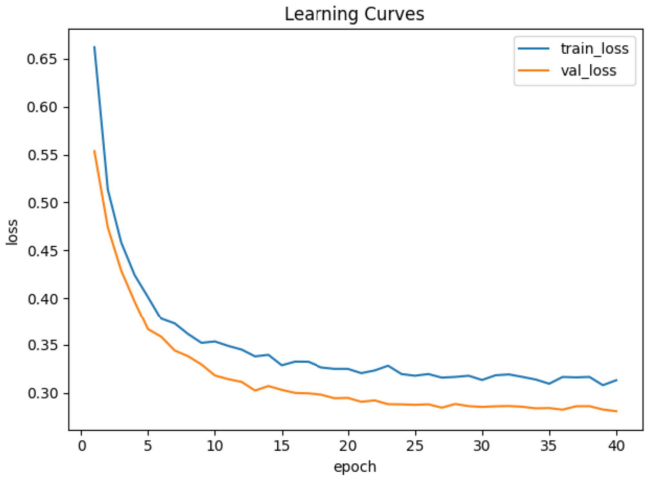
We used a variety of metrics to evaluate performance, including accuracy, precision, recall, F1 score, and ROC-AUC. These measures provided insight into both overall correctness and error types. For example, precision showed how often our predicted "metal" clips were correct, while recall measured how many true metal clips we successfully identified. The confusion matrix gave a clear view of false positives versus false negatives, helping us assess the business impact of accidentally approving non-metal songs.

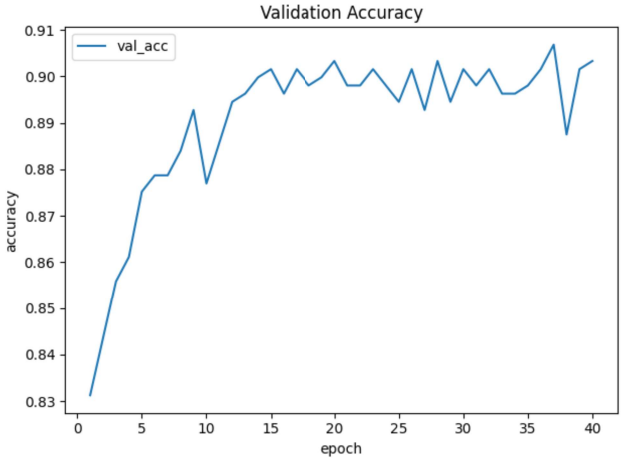
The baseline model achieved a test accuracy of 87.9%, with 84.8% precision and 91.4% recall. After optimization, the test accuracy increased to 94.6%, with precision at 95.7% and recall at 92.5%. An ROC-AUC score of 0.987 confirmed strong separation between the classes. These results exceeded the project's success criterion of ≥90% accuracy, with validation and test metrics within a 5% margin, demonstrating excellent generalization. Through this process, we confirmed that the final CNN reliably distinguished metal from non-metal tracks. The improved precision also reduced false approvals, directly supporting the business requirement of protecting the brand's genre authenticity.

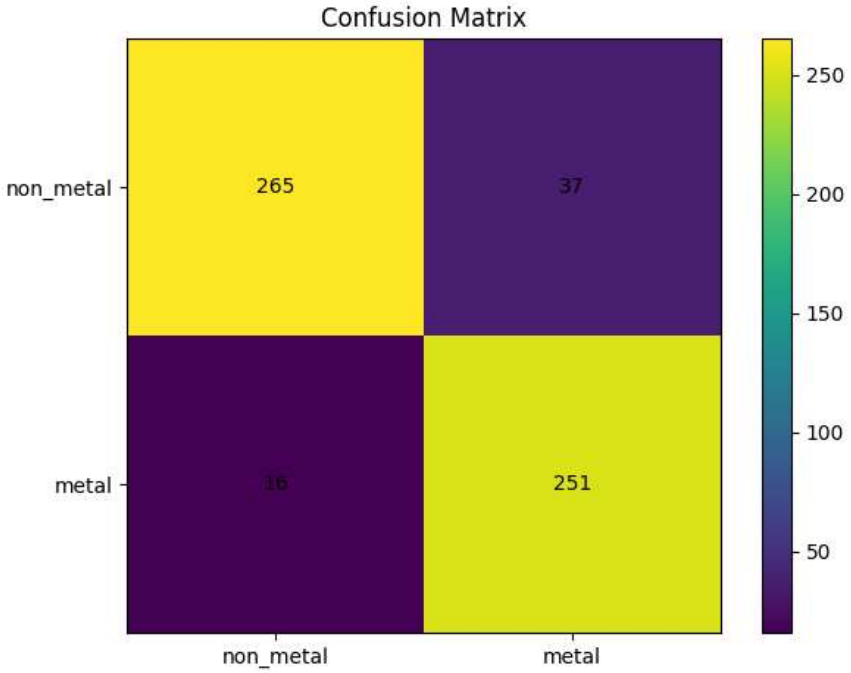
Visualizations

Identify the location of at least three unique visualizations. They can additionally be included here.

Visualizations of the different training runs can be found in the */runs/baseline* and */runs/optimized* folders. These are the learning curves, accuracy, and confusion matrix for the optimized model.


Learning Curves
train_loss
val_loss
loss
epoch
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
5
10
15
20
25
30
35
40


Validation Accuracy
val_acc
accuracy
epoch
0.91
0.90
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0
5
10
15
20
25
30
35
40


Confusion Matrix
non_metal
metal
265
37
16
251
non_metal
metal
250
200
150
100
50

User Guide

Include an enumerated (steps 1, 2, 3, etc.) guide to execute and use your application.

- *Include instructions for downloading and installing any necessary software or libraries.*
- *Give an example of how the client should use the application.*

Quick Start Guide

System Requirements

Windows 10 or 11

Requires FFMPEG installation

Python 3.13.3

Python packages for classification only:

torch, torchaudio, os, shututil, subprocess, tempfile

additional Python packages required for training:

argparse, json, csv, math, numpy, sklearn

Installation

Unzip (or clone if using Git) files to working directory

Place MP3 files in /input_mp3

Run “python process_directory.py”

Files will be automatically sorted into the /sorted_mp3 directory

Note: some sample files have been placed in the directory for testing purposes

Training

1. Collect Music Files

Music files are collected into three directories in the project directory each with a metal and non_metal subdirectory:

/dataset/metal	These files are used to train the model
/dataset/non_metal	
/valset/metal	These are the files against which dataset is validated
/valset/non_metal	
/testset/metal	These are files used to evaluate the final model
/testset/non_metal	

The suggested ratio is an even split between metal and non_metal in each subdirectory with an overall balance of approximately 80% dataset, 10% validation set, and 10% test set.

Current model used 82%-11%-6% (3933/569/290 = 4792).

2. Extract Features

With directories populated in the project directory, run “python save_mfcc_cache.py”. This will process all mp3s by converting them to 22.05 kHz mono WAV files and extracting three 15 second clips. These clips are then converted to MFCCs and saved in the appropriate /mfcc_cache subdirectory.

3. Modify Model

Current optimized model is located in *metal_classifier_optimized.py*. Make desired changes to it, e.g. alter number of parameters or change activation function. Note that this model must stay the same for all following steps as further steps will require the same structure and number of parameters to work consistently.

4. Train Model

Once all features have been extracted, run "python train_with_metrics.py" with the following parameters:

parameter	default	description
--run_name <name>	"baseline"	specify the directory it is saved in
--epochs <number>	40	the number of epochs to train
--step_size <number>	5	# epochs until learning rate is reduced
--gamma <float>	0.6	amount learning rate reduced by
--lr <float>	1e-4	learning rate

Consult train_with_metrics.py for other parameters. Model and metrics are saved in /runs/<run_name> folder.

5. Evaluate Metrics

Run "eval_test.py --ckpt /run/something/best_model.pt" (substitute location of preferred model).

This will give Accuracy, F1, Precision, Recall, and Confusion Matrix information for model evaluation.

Repeat steps 3 through 5, adding/removing parameters, changing training functionality, etc until desired metrics are achieved.