Team Dilophosaurus

Milestone 4 Report

Current State of the Project

The work of this milestone can be found in “genreClassifer_M4.ipynb” (increasing number of samples in dataset, model training), “ensemble_genreClassifier_M4.ipynb” (attempting to improve the genre classifier’s accuracy through a voting ensemble), “song_recommender.py” (GUI), and “recommender.py” (back-end of the GUI; i.e. our work until end of Milestone 3).

We have completed a music recommender app by creating a GUI to interact with all the different components of our project. These components include our genre classifier (developed in M2/3, “genreClassifier M4.ipynb”) and a wrapper for the model. The model wrapper generates spectrograms from an input wav file, feeds them into our model, interprets the model’s outputs to identify the genre of the input file, and recommends another song in the same genre (wrapper developed in M3, “Recommender System M3.ipynb”).

Regarding input, the GUI allows a user to select a wav file through a file browser built into it, and it uses a text box to display the app’s music recommendation.

Overall function of the app:

Input	A wav file of a song, with duration of at least 90 seconds.
Processing	Transform the input into a series of spectrograms and classify their genres (using the genre classifier) before finally taking the average of the outputs for a final verdict on the input’s genre.
Output	Song title and artist name of another song in the same genre.

In addition to the GUI, we also made attempts to improve our genre classifier. The options we came up with include implementing an ensemble of classifiers, increasing the number of samples per genre, and incorporating an RNN at the tail end of our model. Out of the three options, increasing the number of samples per genre was prioritized. We made this our priority, as having a low number of samples (as low as 47) for some genres was the genre classifier’s main bottleneck in learning; the effectiveness of any other method to improve the model was expected to be hampered by this. Thus, we added over 2000 samples across the least represented genres (work found in “genreClassifier_M4.ipynb”). We then tried training ensembles of classifiers with the increased number of samples.

The results (for training with 8 genres):

Model Improvement Method	# Samples per Genre	Test Loss	Max. Test Accuracy
None	47	0.0484	0.3684
Increasing # of samples	547	0.0242	0.6153
Ensemble of 3 classifiers, more samples	547	-	0.62215

Ensemble of 7 classifiers, more samples	547	-	0.63128

Since the ensembles of classifiers didn't perform dramatically better than just a single VGG-19 model with an increased number of samples, we opted not to use an ensemble. The 2% increase in accuracy is too little to justify the extra time to train them and the extra space needed to store them.

In Milestone 3, we had decided to go forward with a model that classified between four genres with 70% accuracy. For our final product, we switched to a classifier that classifies eight genres, but with 61% accuracy. This tradeoff in favor of the number of genres with a moderate downgrade to model accuracy seems reasonable, because many songs fall into multiple genres though we might try to classify them into single genres.

Adjustments to Proposal

- While we originally planned to get all of our data from the Million Song Dataset, we had trouble hosting the full dataset on the cloud. Instead, we used the much smaller Million Song Subset with 10,000 songs and scraped additional data from Youtube.
- The proposal does not specify the audio file type or minimum length for the input. Our final product requires the input to be a .wav file with a minimum length of 90 seconds.
- The proposal mentions training with approximately 30 samples from each genre. In practice, more samples were needed to achieve an accuracy we were satisfied with. Our final model was trained with 547 samples from each genre.

Current Challenges / Bottlenecks

- The full Million Song Dataset is available at https://aws.amazon.com/datasets/million-song-dataset/
    - This is meant to be hosted on Amazon Drive and getting it onto Google Cloud is a bottleneck we are currently dealing with in order to access the entire Million Song Dataset.
    - **Implemented Solution:** Instead of relying on the Million Song Dataset, we used data from the Million Song Subset in and created additional data from Youtube videos.
        - We used the Youtube DL Python library to download music from different genres and then extracted spectrograms from the downloaded music to add to our data.
            - Before: 47 samples/genre (Million Song Subset)
            - After: **547 samples/genre** (Million Song Subset + Youtube).
- Youtube threw the following error: `WARNING: Unable to download webpage: HTTP Error 429: Too Many Requests` in response to our scraping for data.
    - **Solution**: Instead of looking for playlists of hundreds of songs, we used longer videos with compilations of multiple songs. We downloaded and extracted spectrograms from approximately 30 hours of audio split between approximately 15 videos instead of hundreds.

Team Member Contributions

- Carroll, Quinn
  - Collected some more samples for Blues to increase the number of samples available for training by converting a long video compiling blues music into spectrograms
  - Collaborated with team members over visual studio live share and paired programming in Zoom to create the music recommender GUI (found in “song_recommender.py” and “recommender.py”).
- Jung, Cassiel
  - Collected more samples for country, folk, blues and latin to increase the number of samples we used for the classifier as we were using a minimum number of samples for all genres to avoid bias.
  - Attended collaborated zoom call and participated with visual studio live share(work found in “song_recommender.py” and “recommender.py”).
    - Made some feedback regarding error messages to make it clear to the users
- Poon, Matthew
  - Collected more samples to increase the number of samples available for training for the following genres:
    - Rap, Latin, Jazz, Electronic
  - Created the music recommender app and GUI in paired programming with the rest of the team using Zoom and visual studio code’s live share extension (work found in “song_recommender.py” and “recommender.py”).
  - Debugged the GUI and implemented fixes for bugs such as buttons not disabling properly with Aswin
- Sai Subramanian, Aswin
  - Collected some more samples for Folk music to increase the number of samples available for training.
  - Merged all the samples collected by the team into our dataset.
  - Created the music recommender app and GUI in paired programming with the rest of the team using Zoom and visual studio code’s live share extension (work found in “song_recommender.py” and “recommender.py”).
  - Tracked improvements to our genre classifier as the number of samples increased. Trained and saved the model used in the final product (work found in “genreClassifier_M4.ipynb”).
  - Tested out how much an ensemble of genre classifiers would improve genre classification accuracy. (work found in “ensemble_genreClassifier_M4.ipynb”).