Shazam 2.0: Music Genre Classifier

Jasmine Lo, Sean Fuhrman, So Hirota

Introduction

Shazam is a service that can identify a song based on a short segment of a song. Our project aims to create a similar product which can determine the genre of a song, Shazam style. We utilized the Million Song dataset and various Python packages to create a CNN model that classifies the song genre of raw audio data. This poster will explain the methods we used to create the product, which will ultimately be used in our Shazam 2.0 product.

Dataset

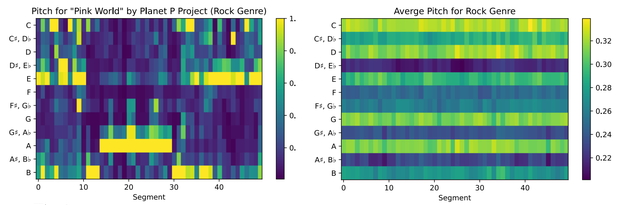
The Million Song Dataset (MSD) is a collection of audio features and metadata for 1 million songs published before 2012. Some audio features were extracted using the Echo Nest API, an API that was provided by Echo Nest, now part of Spotify. We augmented this dataset with the Tagtrum MSD Genre Labels Dataset, since MSD does not come with genre labels. The combined dataset has around 200k rows. We lose around 4/5th of our dataset since many songs do not have a corresponding label in Tagtrum. MSD has many interesting metrics including danceability, loudness, and song hotness, but for our project, we are interested in **segments**, **pitch**, and **timbre**.

Segments

- The MSD does not contain raw audio data. Instead, it contains numeric information for each segment. Each segment roughly represent one note, and comes with two metrics, duration and confidence. Each segment has a corresponding timbre and pitch.
- A segment duration is roughly 0.3 seconds

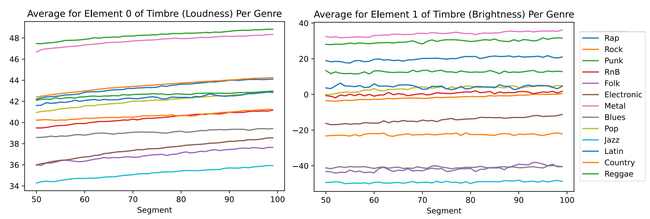
Pitch

- Pitch vector has 12 elements per segment. Pitch cuts the spectrogram into 12 bands called chroma features, where each band corresponds to musical notes from C to B. The first element corresponds to C, specifically it is the sum of the energy of all frequency bands corresponding to C.



Timbre

- Timbre is a high level abstraction of audio data into 12 elements. It is calculated for each segment of the song, and represents the “quality” of musical notes, and can be used to distinguish different types of instruments and voices.
  - Roughly speaking, the first elements in timbre correspond to loudness of the note, the second to the brightness of the note, and the third to the flatness of the note.



Process

Data Ingestion and Preprocessing

- MSD was hosted on AWS as a public dataset snapshot. We used an EC2 instance to attach this snapshot to a volume, and copied all 300GB of data to the UCSD hosted DataHub servers.
- Each song was stored as a hdf5 file. We extracted the pertinent attributes of each song and saved them as a csv.

Preliminary models

- Linear Model: 36% accuracy
  - 50 segments of Timbre and Pitch, 40 epochs
- Single CNN Model: 41% accuracy
  - 50 segments of Timbre and Pitch into one CNN layer, 20 epochs
- Double CNN Model: 44% accuracy
  - 50 segments each of Timbre and Pitch into two separate CNN layer, 20 epochs

Hyperparameter Tuning

- From the results we obtained, we decided on the "Double CNN" model. We improved this model by tuning the hyperparameters.
- Input Segment Length

Segments	Layer 1 Size	Layer 2 Size	Epochs	Accuracy
10.0	256.0	512.0	80.0	0.491
20.0	256.0	512.0	80.0	0.769
30.0	256.0	512.0	80.0	0.879
40.0	256.0	512.0	80.0	0.978
50.0	256.0	512.0	80.0	0.99

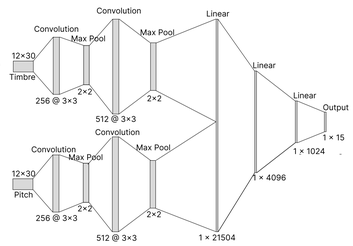
  - Nearly 100% accuracy at 40 segments
- Input Data

Input	Segments	Layer 1 Size	Layer 2 Size	Epochs	Accuracy
Timbre	50	256	512	80	0.47
Pitch	50	256	512	80	0.41
Both	50	256	512	80	0.99

  - The model needs both pitch and timbre

Final Model

- **Achieved 99.6% accuracy on over 200,00 songs**
- Pitch and timbre go through separate CNN networks, then are combined through fully connected layers
- This allows the model to detect different patterns in pitch and timbre.
- For our final model, we used 30 segments to minimize the length of audio needed (about 9 seconds)

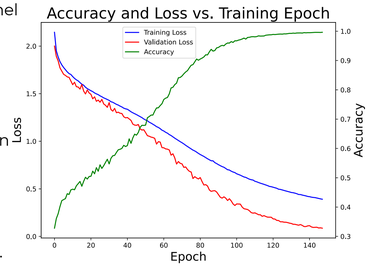


Model Flow:

1. **CNN Layer:** 1x256 channels, 3x3 kernel
2. **Max-Pool Layer:** 2x2 kernel
3. **CNN Layer:** 256x512 channels, 3x3 kernel
4. **Max-Pool Layer:** 2x2 kernel
5. **Linear Layer:** 36864 x 4096
6. **Linear Layer:** 4096 x 1024
7. **Linear Layer:** 1024 x 15

Implementation:

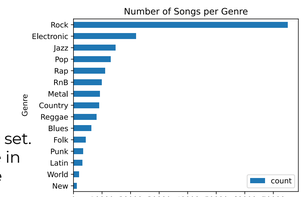
- We implemented our model in Python using PyTorch library.
- We split our data into training, validation, and test datasets that we used to gauge overfitting
- We used a learning rate = 0.001 and CrossEntropyLoss as the loss function.
- Training for 140 epochs was sufficient for the model to reach convergence.



Challenges

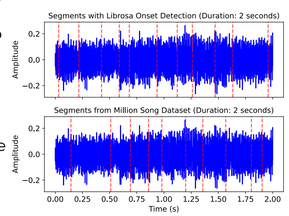
Class Imbalance

- Dataset had a dramatic class imbalance.
  - Rock music represented 40% of our dataset
  - New music represented 0.6% of our dataset
- **Solution:** The Tagtrum Genre labels contained annotations for each song, specifying its inclusion in the test or train set. There were exactly 2000 songs per genre in the training set, which helped ensure the model would not be biased in training.

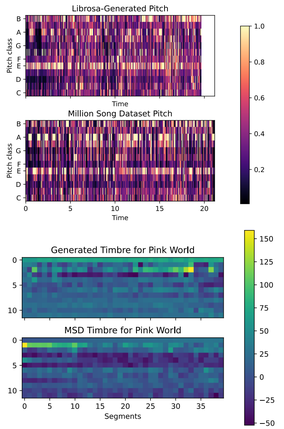


Generating Pitch and Timbre from Raw Audio Data

- The Echo Nest API is no longer available, so we had to recreate segments, pitch, and timbre from raw audio.
- Segment Generation
  - **Solution:** Utilized onset detection from the Librosa Python library
  - **Note**: We don't have the exact audio the MSD used, so it's hard to verify if our segmentation is similar.



- Pitch Generation
  - **Solution:** Utilized Librosa Chroma STFT to generate pitch
  - **Note:** Because our segments don't line up, our pitch is also slightly different.
- Timbre Generation
  - **Solution:** Trained a neural network to generate timbre on segmented waveform data.
  - **Notes**:
    - We wanted to generate timbre using the same mathematical techniques Echo Nest used such as PCA, but none of the methods we tried worked.
    - Our model was trained on waveforms reconstructed from timbre and pitch. The reconstruction methods were provided by Dr. Ellis.



Results

Model

- Final accuracy of 99.6% accuracy
- Our final model was quite large, with 256 and 512 CNN layers
- We trained it on 30 segments for 140 epochs

Hyper-Parameter Tuning

- The model required both pitch and timbre to perform well
- More than 80 epochs is necessary
  - Around 140 epochs, both losses are close to converging
  - Accuracy is sufficiently close to 100% at around 100 epochs

Segments, Pitch, Timbre

- We were able to replicate segments and pitch with Librosa onset detection and chroma STFT
- We were not able to calculate timbre directly, but we were able to approximate them

Conclusion

- The MSD was difficult to work with because the Echo Nest API no longer works
  - We had to reverse engineer audio features to obtain MSD-like features to use as input for our model
- Timbre and pitch combined are very good indicators of song genre
- It would be interesting to do the same project, but without the constraint of using the MSD
  - Potentially scrape our own audio data from YouTube, Spotify, etc.
  - Could create a model that better generalizes to modern songs
  - Could possibly have a more streamlined process, instead of trying to approximate data and reverse engineer the API
- If we had more time, we could have done more hyperparameter tuning
  - Kernel size
    - We used the default kernel size of 3 x 3. While this worked, we want to test out different values that make more sense for our dataset, like 12 x n
  - Trying more epochs
    - It could have been interesting to see how many epochs were necessary for loss to completely bottom out

Acknowledgements

- Special thanks to Dr. DAn P. W. Ellis, a Google researcher who worked on the MSD and Librosa, for pointing us to resources for timbre extraction.
- Thanks to DS3 for providing us with guidance and financial resources to use AWS.
- Thanks to UCSD IT Services for allowing us to utilize computing resources for data processing and model training.

Connect with us: