Classification of Electronic Music Subgenres Using CNNs

COMP4107 Alex Trostanovsky
Prof. White
TAs: Rui Li, Vasileios Lioutas Due Date: 16/12/18

Introduction

The popularity of electronic dance music (EDM) has increased substantially in recent years [1]. EDM artists adapt and experiment with different musical production elements associated with genres such as Techno, House, and Trance. In many cases, these adaptations evolve to produce novel electronic music subgenres [2][3].
Outlined below is an investigation of the capacity of Convolutional Neural Networks to classify the subgenres of a 'niche' electronic music genre: Psychadelic Trance.
We introduce relevant architectures which have been shown to correctly classify well-established musical genres such as: hip-hop, rock, and pop.
We experiment with various training hyperparameters and analyze their impact on the classification accuracy of our model. Finally, we discuss the results of our experiments, outline future research possibilities that may benefit from the results outlined in this report, and consider the application of our model to music recommendation systems.[1]

Background/related work

Due to the recent shift in the music industry towards digital distribution, automatic music recommendation systems have become a well researched and discussed problem amongst online music providers such as Youtube Music, Apple Music,and Spotify.
Recent implementations of Convolutional Neural Networks surpassed the genre classification accuracy metrics achieved by the use of traditional linear regression models.[4]
Using a CNN model that was trained on songs from the Million Song Dataset [5], an AUC (area under the ROC, Reciever Operating Characteristic, curve; a plot of the True Positive vs. False Positive classification rate) of 0.77192 (with an upper bound of 0.96070) was achieved. [4]
The implementation of a CNN based genre classifier has since been adapted by Music Information Retrieval (MIR) researchers by incorporating Recurrent Components.
A CRNN (Convolutional Recurrent Neural Network) can be described as a CNN in which the last convolutional layers are replaced with a RNN (Recurrent Neural Network.)
CRNNs exhibited strong performance with respect to the number of parameters and training time, achieving a consistently higher AUC-ROC (≥ 0.85), when compared to traditional CNNs.[6]

[1]All code available at https://github.com/alextrosta/4107_Final_Project

Possible Bias in Training Data

Upon review of the relevant work, we noted that most of the research that has demonstrated the effectiveness of CNNs/CRNNs in the domain of automatic music genre classification focused on the same datasets (Million Song Dataset, Free Music Archive (FMA))[7]. In addition, the models that were trained on these datasets mainly considered the 'top 50' tags ('rock', 'hip-hop', 'pop').
These genres possess discriminating musical features that could aid in their classification. For example, the dominant use of electric guitars in 'rock' music, or the use of 'Drum Machines' (electronical musical instruments that imitate percussive sounds) in 'hip hop'. Such features are usually indicative of songs belonging to these genres.

Problem Statement

Can CNNs achieve the same strong classification metrics when trained on 'niche' Electronic Music subgenres that posses a lower degree of genre discernability when compared to popular music genres such as 'rock' and 'hip hop'?

Data Preparation

To answer our problem statement, we required a (relatively) large dataset of songs that were established exemplars of Psychadelic Trance subgenres.
Psychadelic Trance is, itself, a subgenre of trance music that is characterized by rhythm arrangements overlaid with high tempo melodies and basslines.
A collection of 920 Psy-trance tracks were obtained from Ektoplazm, a source of 'free and legal psytrance, techno and downtempo music' that has served more than 21 million Creative Commons-licensed releases.
The 920 tracks that make up the dataset consist of:

- 184 Dark-Psytrance tracks - BPM (Beats Per Minute) range: ≥ 148
- 184 Downtempo tracks - Ambient, with Avg. BPM range: ≤ 120
- 184 Full-on tracks - Avg. BPM range: ≈ 145
- 184 Goa Tracks - Melodic, with BPM range: ≈ 145
- 184 Techno tracks - BPM range: ≤ 130

All genre labels were obtained from tags present in the repository.
Since Trance music is heavily reliant on a repetitive 4/4 time signature, some of the chosen subgenres (Full-on, Goa, Dark) may sound very similar due to their common use of instrumentation. For example, different training examples may utilize the same synthesizers to produce basslines and kick drums that take up equivalent frequency ranges.

In addition, the datset also includes compositionally distinct subgenres such as Techno and Downtempo.

Preprocessing

Each track was processed using the core IO functionalities included in the music and audio analysis python package `libROSA`[8].
All tracks were sampled at 44100 Hz (Mono) and a Log-amplitude mel-spectrogram was computed for each track.
The mel scale is a quasi-logarithmic function of acoustic frequency. A mel scale spectrogram groups together perceptually similar pitch intervals (such as octaves) into 'bins' so that they appear equal in width over the full hearing range [9].
CNN models that were trained on Mel-Frequency spectrograms have been shown to achieve a higher degree of classification accuracy when compared to other forms of audio representation [6].
Each track was further segmented into slices of 128 pixels $\times$ 128 mel-frequency bins (essentially a visual representation of the frequencies present in a 2.56s window in a track).
To construct the training and testing dataset, the midpoint of each track was calculated and 30 spectrograms slices were extracted from around that midpoint.

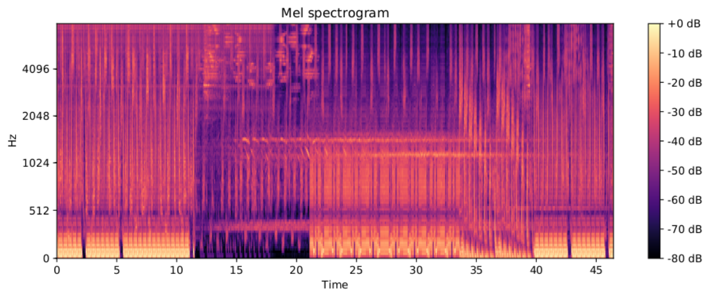


Figure 1: A Log-amplitude Mel-Spectrogram

Model

The architecture was based on a CNN model which has been shown to achieve a high classification accuracy when trained on mel-spectrograms of audio tracks[4].
Implementational details were adapted from a finalist solution to the CrowdAI WWW Musical Genre Recongition Challenge.

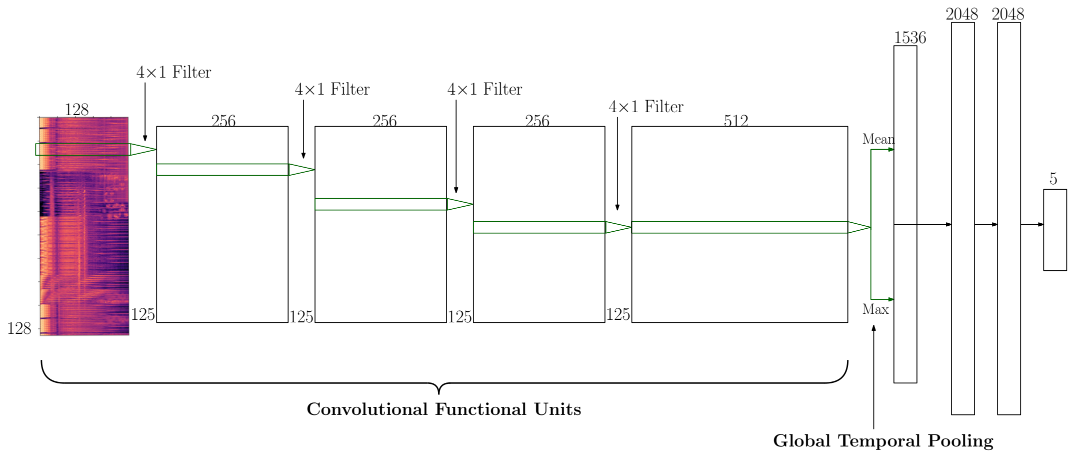


Figure 2: A sample CNN architecture. The time axis (which is convolved over) is vertical.

As opposed to traditional square convolutional filters, the filter that was chosen across layers was rectangular. The convolutions are one dimensional, and happen only across the time dimension, and not in the mel-frequency dimension.
Using square filters is appropriate in cases where the two axes of an image represent the same feature (pixels × pixels), whereas the axes of mel-spectrograms represent time and frequency.
After the last convolutional layer, a global temporal pooling layer was introduced. This layer pools across the entire time axis, and computes aggregate statistics (Mean, Max) of the learned features across time [4].

Training

30 spectrogram slices were extracted from each track to create a dataset of $920 \times 30 = 27600$ samples.
Training was performed in 5-folds. For each fold, the model was trained on 22080 samples, and tested on 5520 samples.
We ensured that for each data batch consumed during training and testing, an equal distribution of all 5 subgenre classes was maintained. This was done by creating protocol buffered, `TFRecord` files that contained a permuted array of 150 spectrogram slices (30 per subgenre).

Experiments

Three experiments were conducted to investigate the performance of the model using different configurations of hyperparameters. All experiments were performed in an Ubuntu 18.04.1 LTS Linux Environment, on an Intel(R) Core(TM) i5-4690K CPU.
Model compilation, training, testing, and predictions were evaluated in a `Keras`, GPU-enabled, `Docker` image, using a GeForce GTX 1070Ti GPU.

Experiment 1: Dropout Probabilities

Models trained using Dropout have been shown to develop more robust feature representations [13]. When training, a certain amount of neurons in the network (proportional to the dropout `keep_prob` parameter) is randomly removed. This causes the activated neurons to 'rely' less on their neighbours, and generate non-zero gradients.
Using Dropout can be viewed as sampling from an exponential weight configuration search space. When a certain weight configuration performs optimally, it will reinforce the neurons and weights which made up that configuration.
While the base model (from which our model was adapted) utilized a dropout probability of 0.2, we were interested to investigate the performance of higher dropout probabilities.

Experiment 2: Optimizers

The following optimizers were asssesed [11] :

- `Adadelta:`
  Adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients.
- `Adam:` Calculates a moving average of the gradient and the squared gradient, and uses hyperparameters to maintain decay rates of these moving averages.
- `Adamax:`
  A generalization to the L^2 norm based update rule of `Adam` using the L^∞ norm. [12]
- `SGD:` (Stochastic Gradient Descent)
  Random samples from the dataset allow timely estimation of gradients in a large search space.

Experiment 3: Global Pooling Layer

The addition of a global temporal pooling layer following the convolutional layers is due to the nature of the task of genre classification. [10]
In general, CNNs use the absolute location of features to classify images. For example, a circular shape could either mean a wheel or a frisbee, depending on the location of the shape in the image.
However, a correct classification of the genre of a spectrogram slice does not depend on an optimal detection of the *locations* of latent features. Instead, the overall absence or presence of features can inform correct classifications of genres. For example, the presence of a repetitive kick drum pattern may be indicative of a Full-on track, while a smooth spectrogram across slices will likely represent a downtempo/ambient track.
Therefore, we investigated classification accuracy with the original global temporal pooling layer (a concatenated `MAX + MEAN pool`), or with either a `MAX pool` or `MEAN pool` global temporal layer.

Results

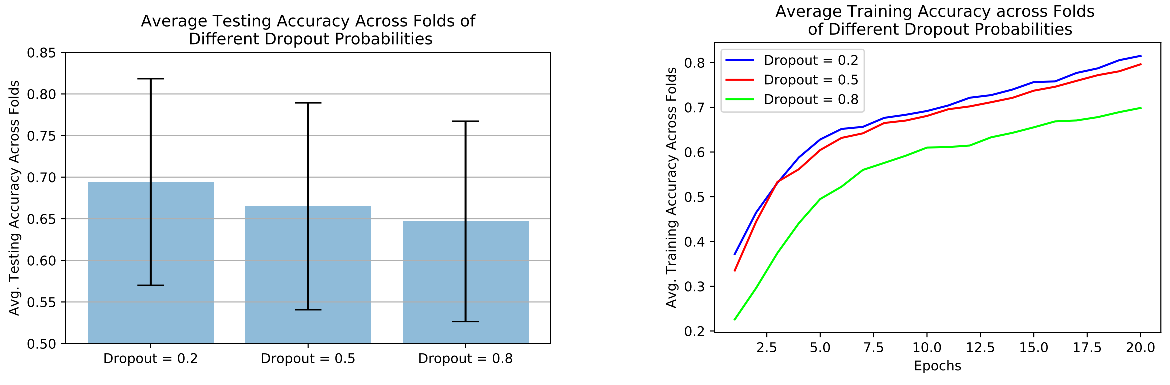


Figure 3: Experiment 1: Dropout. All models trained with `Adam`, `MAX+MEAN` pooling layer.

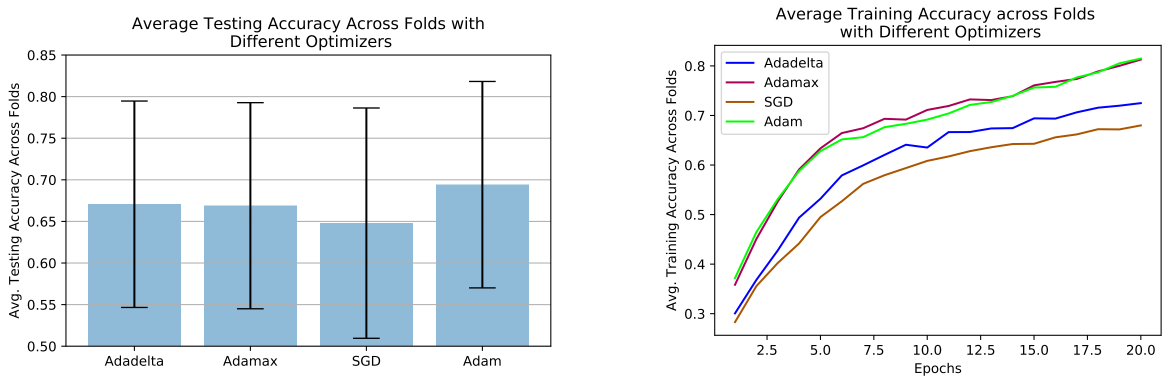


Figure 4: Experiment 2: Optimizers. All optimizers trained with `keep_prob` $= 0.2$

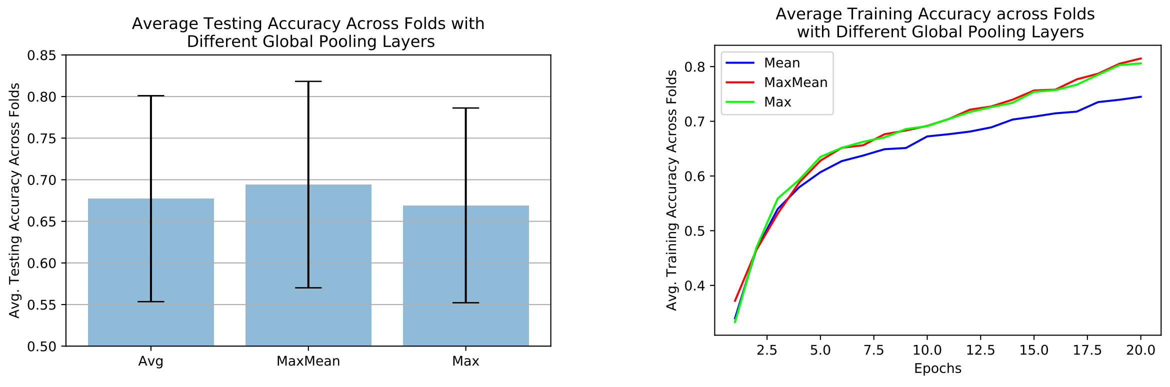


Figure 5: Experiment 3: Global Pooling. All models trained with `keep_prob` $= 0.2$ and `Adam`

Analysis

We conclude that the hyperparameter configuration (Dropout probability = 0.2, `Adam` Optimizer, with a `Max+Mean` global temporal pooling layer) chosen by the original authors of the architecture was optimal. Note that in all three experiments, accuracy on the training data is closely matched by other possible configurations (Dropout = 0.5 in experiment 1, `Adamax` in experiment 2, and just a `Max` pooling layer in experiment 3). However, if performance on the unseen testing set is also considered, then the original hyperparameter configuration performed optimally in both training and testing.

Classification on the testing set did not achieve an accuracy $\geq 70\%$. We believe this was due to the manner in which data was presented for testing. When testing the model, batches of 150 randomly ordered spectrogram slices (30 of each genre) were presented. To calculate the accuracy of the model, the number of correct classifications were divided by the total training samples in the batch.

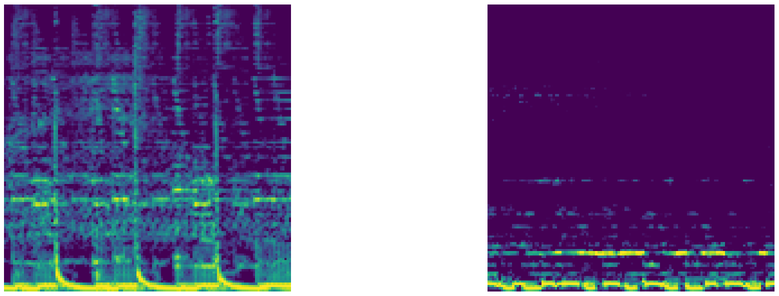
We decided to re-evaluate the prediction performance of our model in the following manner: instead of shuffling 150 samples of 5 different tracks, we presented the model with 30 spectrograms from the midpoints of a selection of novel tracks (unseen in both training and testing) belonging to the 5 classes. The model computed the predictions for each of the 30 spectrograms. To classify a track, the dominant classification out of the 30 would be chosen. For example, if presented with a Full-on track, we would expect some of the spectrograms to be classified as either Goa, or Dark, since they occupy a similar BPM and frequency range. However, if the majority of the spectrograms were classified as 'Full-on', the prediction would be considered accurate.

After training a model using the optimal hyperparameter configuration, the model achieved an 84% correct prediction rate when presented with 51 novel tracks (with approximately 10 tracks per class).

Discussion

As was anticipated, most misclassifications occurred between tracks belonging to genres with a similar BPM or Instrumentation Frequency Range (Goa, Dark, Full-on).

Surprisingly, a substantial amount of the misclassifications occurred when the model incorrectly predicted a track was Downtempo. We hypothesize that these tracks were misclassified because the 30 spectrograms (that were extracted from the midpoint of the track) did not contain representative features which were associated with the track's genre. This can happen if the spectrograms for a track were contained within a segment of the track known as a Breakdown or a Drop (sudden changes in rhythm or bass line).


(a) Correctly classified as 'Goa'

(b) Misclassified as 'Downtempo'

Figure 6: Two slices of two different Goa tracks. (a) was correctly classified as it exhibits the common 4/4 kick pattern, while (b) likely represents a break/drop, and so was misclassified.

To address this, an alternative spectrogram segmentation strategy may be used. Instead of arbitrarily choosing the midpoint of a track as the representative locus around which spectrograms are extracted, one could use onset detectors [14], or beat and tempo trackers [15], to preprocess an audio track. Using these, the choices of loci around which one segments an audio track may be more informed, and provide relevant data with which to train CNN models.
Finally, as was mentioned earlier, future research may also investigate the accuracy performace of CRNNs in genre classification of Electronic Music. Since RNNs have been shown to be flexible in the summarization of local features [6], issues of misclassifications due to unrepresentative segmentations (as were shown by our preprocessed data) may be resolved by: a) making informed choices about sampling locations within an audio track and b) using a Recurrent Architecture that takes into account the global structure of an audio track.

Conclusions

We have shown that CNNs trained on Log-amplitude mel-spectrograms of Psychadelic Trance audio tracks can achieve an 84% correct subgenre classification metric when presented with novel samples. Previous research has shown that CNN/CRNNs perform strongly in the domain of genre classification of popular music. The results obtained in this report indicate that CNNs can be trained to accurately classify a diverse range of musical genres.

The increasing popularity of electronic music, along with the growing demand for online music recommendation services, should be considered when developing intelligent recommendation systems that can correctly identify new trends in Electronic Music. It appears that CNNs can learn the latent features that set trending subgenres apart, but to do so, models should be trained on modern datasets that contain examples of the current Electronic Music landscape.

References

[1] It's a $6.2 billion industry. But how did Electronic Dance Music get so popular?. (2015, October 30). CNN. Retrieved from https://www.cnn.com/2014/12/18/world/how-did-edm-get-so-popular/index.html

[2] Timeline of electronic music genres. In Wikipedia. Retrieved December 15, 2018 https://en.wikipedia.org/wiki/Timeline_of_electronic_music_genres

[3] McLeod, K. (2001). Genres, subgenres, sub-subgenres and more: Musical and social differentiation within electronic/dance music communities. *Journal of Popular Music Studies*, 13(1), 59-75.

[4] Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems* (pp. 2643-2651).

[5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th *International Society for Music Information Retrieval Conference* (ISMIR 2011), 2011.

[6] Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017, March). Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2392-2396). IEEE.

[7] Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, Xavier. FMA: A Dataset for Music Analysis. *18th International Society for Music Information Retrieval Conference.* 2017 https://github.com/mdeff/fma

[8] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Vincent Lostanlen, Colin Raffel, ... Adrian Holovaty. (2018, August 9). librosa/librosa: 0.6.2 (Version 0.6.2). Zenodo. http://doi.org/10.5281/zenodo.1342708

[9] https://librosa.github.io/librosa/generated/librosa.core.mel_frequencies.html

[10] Dieleman, S. (2014). *Recommending music on Spotify with deep learning.* Sander Dieleman, 5. Retrieved from http://benanne.github.io/2014/08/05/spotify-cnns.html

[11] https://keras.io/optimizers/

[12] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[13] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.

[14] https://librosa.github.io/librosa/onset.html

[15] https://librosa.github.io/librosa/beat.html