Timothy Mason, timothy.mason@colorado.edu

CSCI 5922 – Neural Networks and Deep Learning, Sprint 2020

University of Colorado Boulder

6-May-2020

CRNN for Predicting Music Tags

INTRODUCTION

The original goal of this project was to reproduce the music tagging results presented in the paper **Convolutional Recurrent Neural Networks for Music Classification** (Choi, Fazekas, Sandler, & Cho, 2016). As work progressed, it became clear that reproducing their approach was beyond the scope of this project due to the complexity and processing time needed for gathering data. Therefore, this project was modified to only use data that is contained in the subset of the **Million Song Dataset** (Thierry, Daniel, Brian, & Paul, 2011) at http://millionsongdataset.com/pages/getting-dataset - subset plus the companion last.fm tags dataset at http://millionsongdataset.com/lastfm/.

Using this subset data, I was able to achieve very good results (0.9996 AUC) on a 95% / 5% training split. However, I am concerned there may be overfitting occurring which causes the fit to look better than it is. The Choi, et. al. paper showed AUC of about 0.86 as their best result. Unfortunately, due to the massive size (280GB) of the full Million Song Dataset, and the complexity of accessing it from an AWS public snapshot, I was not able to validate the results on a larger dataset.

LITERATURE REVIEW

Inspiration for this project came from the paper **Convolutional Recurrent Neural Networks for Music Classification** (Choi, Fazekas, Sandler, & Cho, 2016). As described in their abstract:

> *We introduce convolutional recurrent neural network (CRNN) for music tagging. CRNNs take advantage of convolutional neural networks (CNNs) for local feature extraction and recurrent neural networks for temporal summarization for the extracted features.*

They go on to summarize:

> *Overall, we found that CRNN's show a strong performance with respect to the number of parameter and training time, indicating the effectiveness of its hybrid structure in music feature extraction and feature summarization.*

DATA

The dataset I proposed to use for this project was the **Million Song Dataset** (MSD) with companion *last.fm* tags (Thierry, Daniel, Brian, & Paul, 2011). "The **Million Song Dataset**" is a freely available collection of audio features and metadata for a million contemporary popular music tracks." … "The core of the dataset is the feature analysis and metadata for one million songs" … "The dataset does not include any audio, only the derived features". On investigation, I found that the full dataset is prohibitively large at 280GB, and it is only available as an Amazon Web Services (AWS) snapshot, which would have posed significant challenges for import and usage. Luckily, the author also provides a much smaller "MSD subset" containing data for 10,000 tracks. This subset is easily downloadable from the MSD website and provided as a tree of Hierarchical Data **Format** version 5 (**HDF5**) files, one file per data record.

The *last.fm* dataset is described as "the official song tag and song similarity dataset of the Million Song Dataset". This companion dataset is provided in multiple formats, including "a subset, corresponding to tracks from the 10k songs in the MSD subset."

For this project, I chose to use the MSD subset plus the tags data from the *last.fm* subset companion JSON data. Both datasets are included in the github repo that was submitted with this project. For reasons of scope and practicality, I chose to use only the data from these two sources. This contrasts with the approach taken by Choi, et. al., wherein they used an unspecified service to download the raw audio files corresponding to each entry in the MSD dataset, and perform heavy post-processing on each file.

This turned out to be an acceptable compromise since the MSD contains a data field "segments_pitches" for each track which has a 2D vector of average pitch data versus time quantums for each song:

> *pitch content is given by a "chroma" vector, corresponding to the 12 pitch classes C, C#, D to B, with values ranging from 0 to 1 that describe the relative dominance of every pitch in the chromatic scale. For example a C Major chord would likely be represented by large values of C, E and G (i.e. classes 0, 4, and 7). Vectors are normalized to 1 by their strongest dimension, therefore noisy sounds are likely represented by values that are all close to 1, while pure tones are described by one value at 1 (the pitch) and others near 0.*

In the MSD, each song contains the 2D segments_pitches record, which turned out to be very suitable for training my CRNN music classification network.

For the truth values of music tags, the *last.fm* dataset turned out to be overkill. According to the documentation at http://millionsongdataset.com/lastfm/, there are 522,366 unique tags in the database. I chose to reduce this to a manageable number in the same way as Choi, et. al, by limiting the classifier to the top 50 most frequently used tags:

> *We train the networks to predict the top-50 tag, which includes genres (e.g., rock, pop), moods (e.g., sad, happy), instruments (e.g., female vocalist, guitar), and eras (60s – 00s).*

Here is the full list of target tags, in order by frequency of occurrence:

> *rock, pop alternative, indie, electronic, female vocalists, favorites, Love, dance, 00s, alternative rock, jazz, beautiful, singer-songwriter, metal, chillout, male vocalists, Awesome, classic rock, soul, indie rock, Mellow, electronica, 80s folk, british, 90s chill, american, instrumental, punk, oldies, seen live, blues, hard rock, cool, Favorite, ambient, acoustic, experimental, Favourites, female vocalist, guitar, Hip-Hop, 70s party, country, easy listening, sexy, catchy*

CHALLENGES – MODEL VS DATA DIMENSIONS

The CRNN network implemented by Choi, et. al., (Figure 1) contains a total of four CNN's with max-pooling of (2x2) - (3x3) - (4x4) - (4x4). This works for their implementation because the input data for each song is a 96x1366 time quantum vs frequency vector (zero padded to 96x1440). With that input dimensionality, the output from the 4th CNN is a 1x15 vector which is then fed into the first of two RNN's.

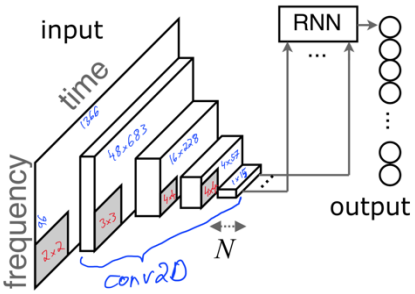


Figure 1: Original CRNN Architecture from Choi, at. al. Pooling in grey

The segments pitch data from the MSD, however, has dimension 12 x <variable>. The reduction in the pitch dimension is because the MSD representation considers all "C" notes to be equivalent, regardless of the octave whereas the Choi, et. al. approach encoded unique values for each octave. Given the way humans perceive music, however, this compression of the pitch axis shouldn't matter. While multiple octaves of pitch content add depth to our perception of a song, said song will still be recognizable even when compressed to a single octave.

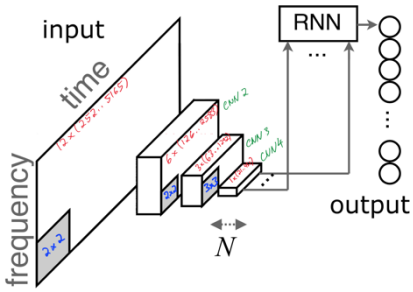
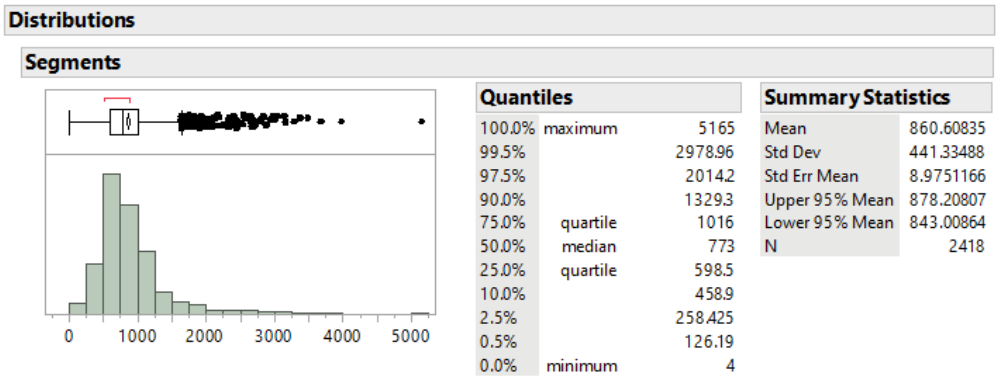


Figure 2: Modified CRNN Architecture

However, this presents a challenge in that the original CRNN model requires a depth of 96. To address this, I reduced the CNN count to three instead of the original four, and changed the pooling sizes to be (2x2) - (2x2) - (3x3). This accomplishes a divide-by-12 which succeeds in reducing the pitch axis to 1 at the input to the RNNs (Figure 2), while still harnessing the learning power of multi-level CNN's.

RAGGED TENSORS?

Another challenge was encountered with the time axis of the MSD segments_pitches data. As described earlier, Choi, et. al. took the approach of pre-processing (presumably) thousands of audio files for their input data. As part of this preprocessing, they chose to only include 1366 time samples, extracted from a midpoint within each song. The MSD pitch data, however, is of variable length on the time axis. The data for each song contains a different number of samples, ranging from some pieces with only 4 segments, to one that has over 5,000.



Quantiles		
100.0%	maximum	5165
99.5%		2978.96
97.5%		2014.2
90.0%		1329.3
75.0%	quartile	1016
50.0%	median	773
25.0%	quartile	598.5
10.0%		458.9
2.5%		258.425
0.5%		126.19
0.0%	minimum	4

Summary Statistics	
Mean	860.60835
Std Dev	441.33488
Std Err Mean	8.9751166
Upper 95% Mean	878.20807
Lower 95% Mean	843.00864
N	2418

Figure 3: Distribution of pitch data segment counts in MSD subset

The distribution of pitch data durations is show in (Figure 3).

Initially, I attempted to make use of the Tensorflow RaggedTensor object which seems specifically made for this sort of data. The plan was to filter the dataset to discard songs with too few segments, then train the network on the remaining (longer) samples. The threshold was originally chosen to be 252 (chosen because (12, 252) ÷ 12 would still leave a size (1, 21) input to the first RNN layer). In the end, I had to abandon this approach after several days of trying. I just couldn't get the CNN's to accept Ragged Tensors as the input, and I didn't want to go down the path of zero-padding to a fixed length.

Instead, I chose to modify the data preprocessing to only include the first *x* segments from each song. Any song with less than the *x* threshold segment count is discarded. The threshold *x* is implemented as the hyperparameter `segments`. By doing this truncation and filtration, I can use fixed size tensors as input. In hindsight, there was no reason to consider using the entire song from start to finish and messing with ragged data. Intuitively, it makes sense that one does not need to listen to an entire track to assign appropriate.

JOINING AND FILTERING

Mathcing each song in the MSD subset to its' corresponding *last.fm* top-50 tags was an interesting challenge. Both MSD and *last.fm* datasets contain a track_id category which is intended as a unique index key. However, the *last.fm* dataset web page warns that this index is not completely reliable. So, I chose to join the two datasets using a three-way tag key of track_id + title + artist_name. Also, not all songs have tag data, and many songs have tags that are not included in the top-50 list. The final merged dataset was filtered to only include those tracks which have at least `segments` pitch segments and at least one top50 tag. This

segments	Remaining Songs
1,024	587
512	2,049
252	2,243

Table 1: Songs Remaining After Filter

severely reduced the number of tracks in the dataset. The remaining track count depends on the value of `segments`. See (Table 1) for dataset sizes at various thresholds.

EXECUTION SPEEDS

A final challenge was encountered with execution speed. Two varieties of the modified CRNN were implement. The first was a "smaller" model based on the "100,000 parameter CRNN" hyperparameters from Choi, et. al. The second was a "large" model based on the "3,000,000 parameter CRNN" hyperparameters. See (Table 2) for a summary of the critical hyperparameter differences between the two models.

Layer	Smaller Model (N)	Large Model (N)
conv2d	60	339
conv2d	60	339
conv2d	60	339
rnn	30	169
rnn	30	169

Table 2: Model Hyperparameters

The execution time challenge came when attempting to train these networks. On my 2017 MacBook Pro with 8-core processor and 16GB of RAM, training the smaller network took about 1.5 hours. The Large model, on the other hand, took 7.5 hours. To reduce this time, I attempted to move to the Google Colab platform. Unfortunately, this caused an issue with data loading speeds.

As mentioned earlier, both the MSD subset and the *last.fm* tags data sets are provided as literally thousands of small files. On the MacBook Pro with a fast internal SSD, loading all of that data into Pandas dataframes takes around 6 minutes. On Google Colab, loading the same data files from gDrive storage takes several hours. gDrive clearly is not architected for efficient access to small files.

The solution was to take a hybrid approach. I took advantage of the MacBooks data loading speed to create Python 'pickle' files of the fully loaded, joined, and filtered Pandas dataframes. The pickle file has the value of the `segments` hyperparameter included in the filename, thereby allowing multiple unique snapshots to be created and stored. I then uploaded those pickle file snapshots to my gDrive for use by the Colab notebook. Because the pickle file is a single large file, loading it on Colab is extremely fast (less than one second). I was then able to train the networks on Colab with an attached GPU. By doing this, the large model training time was reduced from the previously mentioned 7.5 hours down to about 6 minutes!

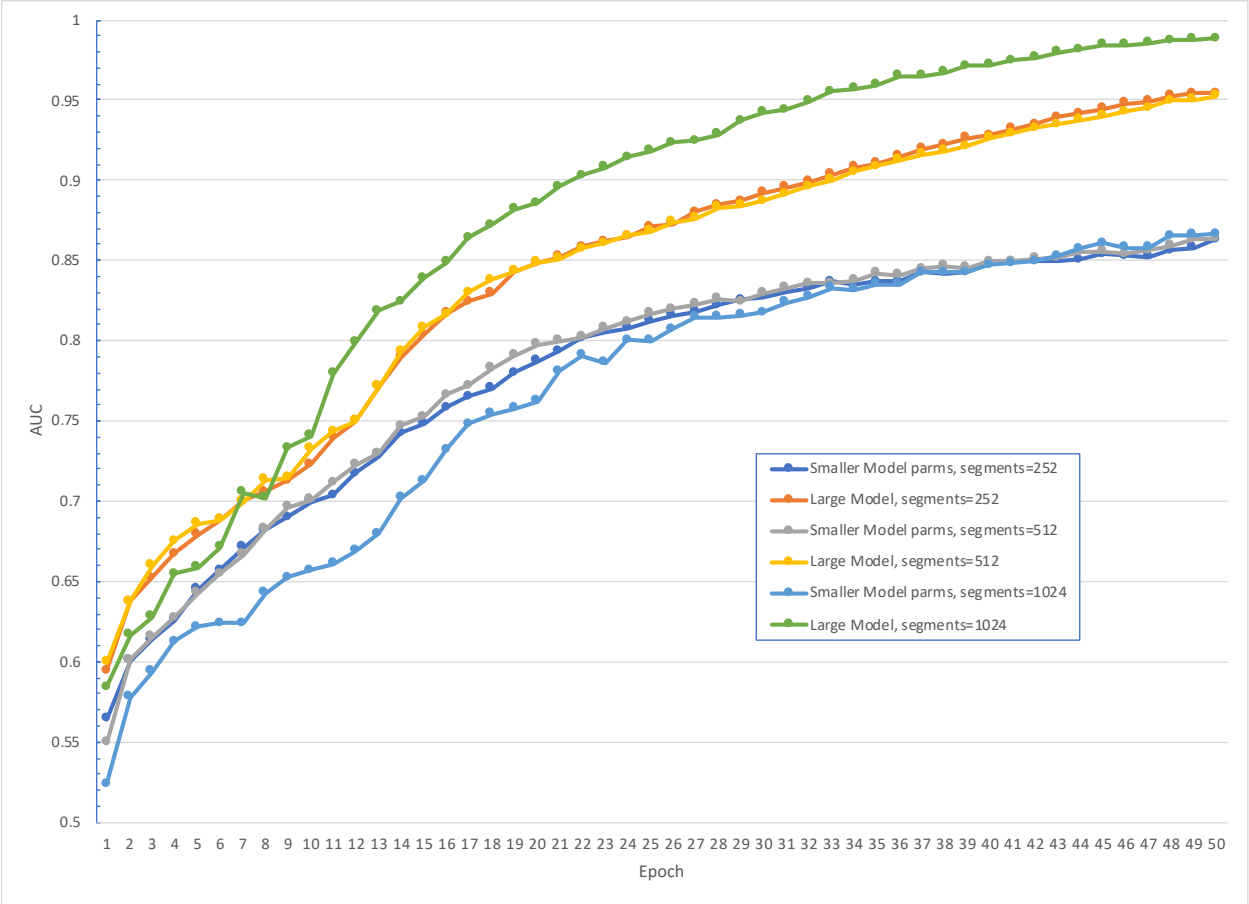


Figure 4: Initial Training Results

RESULTS – INITIAL TRAINING

As was done in Choi, et. al., my networks trained on a metric of AUC (Area Under Curve). Using this metric, Choi reported a best result of about 0.86 for their largest (3,000,000 parameter) CRNN model. That was contrasted against a state of the art (at that time) result of 0.851.

See (Figure 4) for the results of the initial model evaluations with various hyperparameter settings. Surprisingly, some of the models showed AUC results far exceeding what was reported in Choi, et. al. Unfortunately, I suspect that this is a false result resultant from overfitting. Because the dataset was so heavily filtered (especially at the larger `segments` settings), there is probably not enough data present to get a reliable result. The solution, of course, would be to get a larger data set. As mentioned previously, retrieving the full 1,000,000 song dataset from AWS was beyond the scope of this project, but that option remains open for future work.

We can, however, reliably look at the relationships of the curves for various hyperparameters. Unsurprisingly, the large model (with 2,514,154 trainable parameters and 2,058 untrainable) performs much better than the small model (81,314 trainable, 384 untrainable). Since the large model was able to be trained in a reasonable amount of time on Google Colab with attached GPU, my further explorations were limited to only using the large model.

As for the `segments` threshold, having a threshold of 1024 shows dramatically better results, but again I suspect that is because of overfitting. With the threshold at 1024, there are only 587 songs in the dataset. Despite that, I was curious to explore this threshold further.

I was surprised to see that setting the `segments` threshold to 512 versus 252 made no difference in the AUC -vs- epochs curve. I suspect a repeatability study would show that the difference between these two settings is not statistically significant, but that exercise is left to future work. This result does, however, lend credence to the hypothesis that the AUC performance is dominated by overfitting. Both 252 and 512 have approximately 2,000 songs in the dataset, and they perform the same for a given model. Based on this, I discontinued exploration with 512 to focus solely on 1024 and 252.

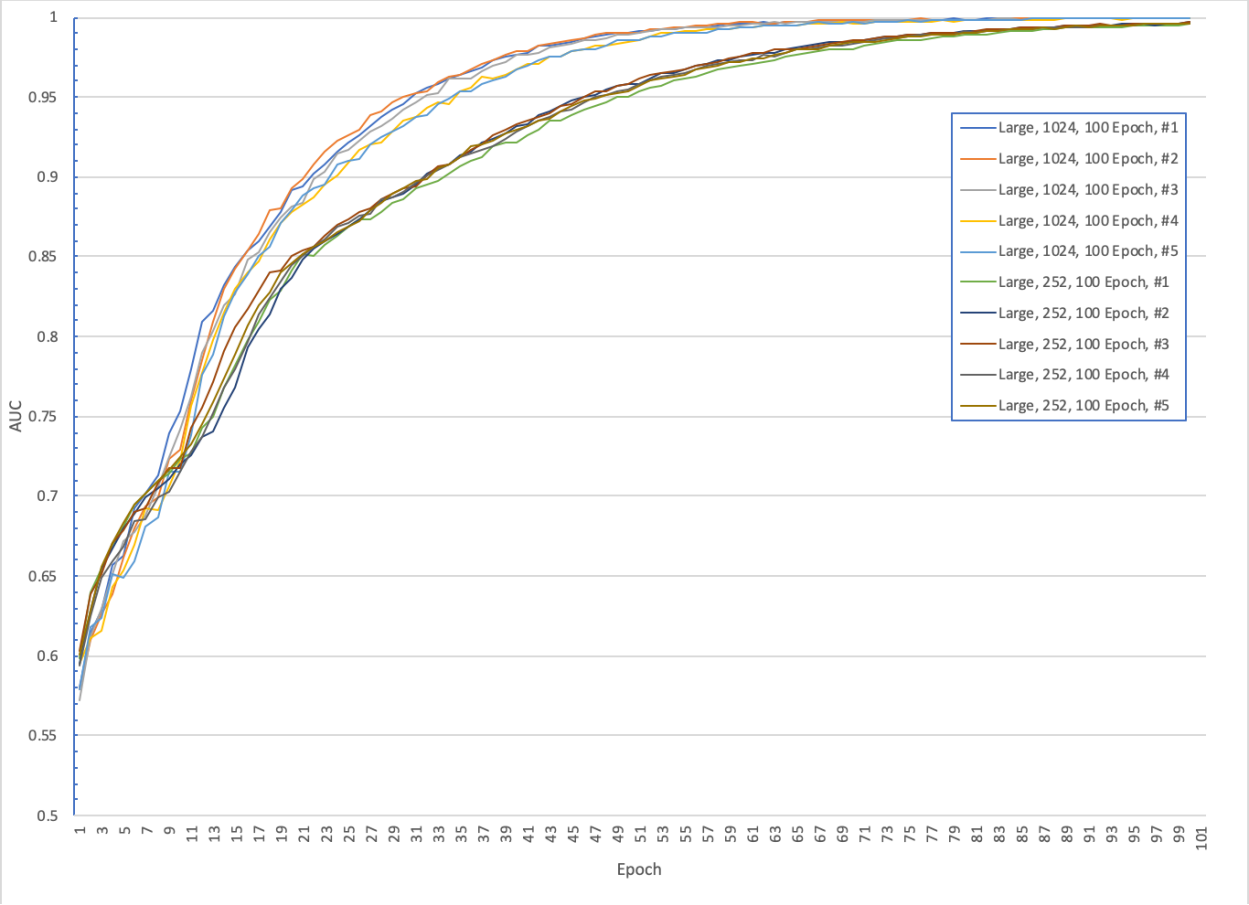


Figure 5: Repeatability and More Epochs

REPEATABILITY

See (Figure 5) for the results of a small 5-sample repeatability study. The model was fully retrained after a kernel reset. Five curves were gathered at `segments`=1024, and another five at `segments` =252. Also, I observed from (Figure 4) that the AUC curves had not fully 'flattened' at 50 epochs so I extended the epoch count to 100 to try and gauge how good the models could get.

Both models demonstrated an ability to reach an AUC very close to 1.0. Again, I believe this is a result of overfitting. The 252-segment model just takes longer to get to full memorization because it has more songs to consider.

CONCLUSIONS

The CRNN appears to have a powerful capability to automatically generate music tags based on the sequence of harmonic sound frequencies (pitches) over time. With more resources and more data, this system could be extended and made general purpose. For example, any streaming service such as Spotify which provides a third-party API could be used for automatic tagging of songs using a CRNN which was trained from MSD data. Some changes would have to be made to provide data from the full MSD dataset as well as increased countermeasures against overfitting (perhaps increase the dropout rate within the network).

The data presented in this paper does support the hypothesis that accurate tag determinations can be made based on a small subset of the song data. For future exploration, it would be interesting to explore how small is "too small".

Another avenue that would be interesting to explore is to increase the categorization power beyond the top-50. According to data at the MSD *last.fm* website (http://millionsongdataset.com/lastfm/), there are many tags which occur fewer than 10 times across the entire 1,000,000 songs. That is probably too few to train a model, but it should certainly be possible to extend beyond just top-50.

REFERENCES

Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2016). Convolutional Recurrent Neural Networks for Music Classification. *arXiv*, 1609.04243v3.

Thierry, B.-M., Daniel, P. W. E., Brian, W., & Paul, L. (2011). *The Million Song Dataset Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.