Classifying Song Genres Using Raw Lyric Data with Deep Learning

Connor Brennan, Sayan Paul, Hitesh Yalamanchili, Justin Yum CS194-129: Group 3
University of California, Berkeley

Abstract

With the advent of music streaming services in recent years and their respective recommendation algorithms, classifying song genres is a commercially useful application of deep learning. Our main focus was to use raw lyric data, rather than a bag of words or other types of summaries, so that we could capture positional information for each word. Specifically, we used lyric information to classify a song as one of 11 genres. The three main architectures we investigated were the LSTM baseline, hierarchical multilayer attention with GRUs (HAN-GRU), and Transformers with multi-head attention. We found that stacking layers in the HAN-GRU was both the fastest to train and the best performer of the three, reinforcing the power of hierarchical attention.

1. Introduction

Today, music is widely consumed through streaming services, which provide instant access to millions of tracks. With such an abundance of material, users are exploring new music with great ease. A problem that may arise is that there is too much music, with fairly sparse metadata, which may complicate music discovery.

Modern services like Spotify and last.fm can commonly featurize their users using their music listening history, and then apply a variety of techniques to make recommendations. However, both of the aforementioned services could potentially have a cold start problem, where they have no user history to reference. Without a user's history, one of the most important factors in predicting whether one may enjoy new music is whether it falls into a musical genre that one already likes (Gossi & Gunes, 2016). This way, the user can take matters into their own hands, and simply look for music in their favorite genres.

Musical genres can be distinguished from each other on the basis of both instrumentation and lyrical conventions. In the modern musical landscape, there are dozens of identifiable musical genres and subgenres. Our chosen problem is to utilize deep learning to classify a song as belonging to a particular genre, given just the lyrics to that song, with the list of genres being Indie, Country, Jazz, Metal, Pop, R&B, Other, Folk, Rock, Electronic, and Hip-Hop.

1.1. Dataset

We used the MetroLyrics Dataset, which can be found in Kaggle's dataset library[1]. Although it has a fairly limited number of features for each of the songs in the dataset, it does have the song name, year, artist, genre, and most importantly, the full lyrics of the songs, which are more than sufficient for our goal of classifying song genres only using the full lyric text. The songs in the dataset were categorized into the following genres: Indie, Country, Jazz, Metal, Pop, R&B, Other, Folk, Rock, Electronic, and Hip-Hop.

2. Background

Metrolyrics is a website which receives contributions from fans of the music, who manually transcribe and enter lyrics into the site. As there is no regulatory body of moderators, different contributors adhere to different conventions. For instance, a user may transcribe the hook from the hit Black Eyed Peas song "I Gotta Feeling" in either of the following ways:

1. *[Hook]*
   I got a feeling that tonight's gonna be a good night
   That tonight's gonna be a good night
   That tonight's gonna be a good, good night
   A feeling
   That tonight's gonna be a good night
   That tonight's gonna be a good night
   That tonight's gonna be a good, good night

[1] `https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics`

I got a feeling that tonight's gonna be a good night
That tonight's gonna be a good night
That tonight's gonna be a good, good night
A feeling
That tonight's gonna be a good night
That tonight's gonna be a good night
That tonight's gonna be a good, good night

2. *Repeat x2*
I got a feeling that tonight's gonna be a good night
That tonight's gonna be a good night
That tonight's gonna be a good, good night
A feeling
That tonight's gonna be a good night
That tonight's gonna be a good night
That tonight's gonna be a good, good night

In addition to the inconsistent formatting of the song lyrics as text, the genre representation of songs in this dataset was also unbalanced, as there were almost 85000 Indie songs after the dataset was cleaned. This is something to note as we develop the approaches used for training the model on this dataset and analyze the results, particularly when looking at the confusion matrices for the various approaches used.

The clearest measure of success is validation accuracy on 10% of the songs in the dataset that were held out from the training set. We were also interested in testing the possibility of high-accuracy binary classifiers, i.e. a classifier which reports whether a song is likely hip-hop or not. A successful genre classifier could be of use to a music-providing company, wishing simply to categorize its music catalog or eventually to jumpstart its recommendation system. A simple questionnaire requesting only genre preferences would enable said music provider to provide a service similar to Netflix's ability to start showing recommendations for movies based on stated genre preferences, even before users start rating specific titles.

There have already been many non-neural approaches to classifying a song's genre just from its lyrics. Canicatti used a combination of song lyric data collected from the public MetroLyrics API with a database of song metadata (including features like artist and BPM) to classify songs into one of five genre: rock, rap, country, jazz, and pop. These genres were specifically chosen because they are unlikely to have overlap in either lyrical content or song structure. After converting all of the songs into word-frequency vectors, he used PCA to determine which words were the most indicative of a song's genre. With this, he performed a number of experiments with classifiers like J48 decision trees, Random Forest, k-nearest-neighbor, and Naive Bayes to see which worked the best for his dataset. Because the rap lyrics he used had misspelled or misformatted words, that hurt the number of distinguishing features for the classifiers he used, so he reported his accuracy without the rap genre. Of his adjusted accuracy measurements, Random Forest was the best of the four with about 52% accuracy while the others were between 34% and 40% accuracy (Canicatti, 2016).

Tsaptsinos focuses on using only lyrics, as we are, to train a deep genre classifier. He uses a hierarchical attention network (HAN) built on top of a recurrent neural network, with an eye towards explicitly learning hierarchical features on top of the text inputs such as words, lines, and segments. He obtained their data through a signed research agreement with LyricFind, which consisted of unlabeled JSONs, and retrieved the genre classes through the iTunes API. Comparison models included the mode (the most common genre), logistic regression, long-short term memory units (as opposed to the gated recurrent units used in the HAN), and a hierarchical network (which lacked attention). The study found the HANs to be the most effective model in the complex 117-genre classification task, with a 46.42% accuracy, although it found LSTMs to suffice and perform better on the simpler 20-genre task (Tsaptsinos, 2017).

Briefly as context, Holmdahl attempts the different task of guessing the song title from the lyrics, reasoning that the former is often contained in the latter. He used a bidirectional gated recurrent unit network on a dataset combined from Lyricsfreak and Metrolyrics (as we are using). He used an n-grams extraction model as a baseline, and managed to achieve an 11.8% improvement using the neural techniques (Holmdahl, 2017).

3. Approach

3.1. Data Preparation

Our primary data source for this research was the MetroLyrics Dataset, as mentioned earlier. Because of the open-source creation of this dataset by many different contributors, there many formats for the lyrics transcriptions and the transcriptions themselves were messy. Initially, all songs that were purely instrumentals and did not contain lyrics were removed from the dataset. Almost all punctuation, non-ASCII characters and musical descriptors like "Chorus" and "Verse" were also removed from the lyrics. In addition, looking

through the dataset, there were many songs that had non-English lyrics or contained multiple language translations of the song lyrics. Using the `DETECTLANGUAGE` function provided by Google Apps Scripts, any songs containing these translated copies or were entirely non-English were removed using this API.

In addition to these overall modifications to the dataset, for some models like the HAN-GRU architecture models, lines and new line characters were preserved in the song lyrics to take advantage of applying attention in a multi-layered manner on various lines of the song and then the words of the lines. For the other models, the new line characters were also cleaned out as line-level attention was not used.

3.2. Baseline Models

3.2.1. Vanilla LSTM Recurrent Network

We considered several baseline models for this study. One of the models we decided to use as a baseline was a standard recurrent neural network consisting of a vanilla LSTM cell with 60 hidden units, shown in figure 1. A one-dimensional max-pool over the hidden states followed by a dense softmax converted these temporal outputs into final classification probability predictions. This model performed surprisingly well and would prove to be very close in validation accuracy to some of our best final models.

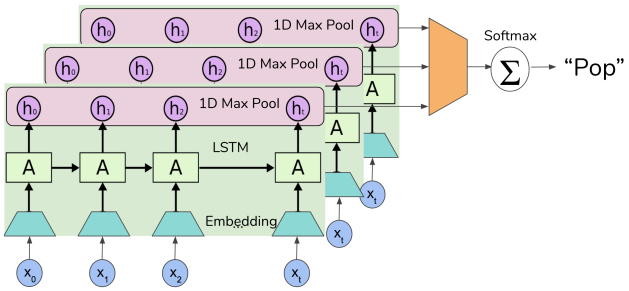

Figure 1. Basic Vanilla LSTM Architecture

As an addendum to this model, we also considered a bidirectional LSTM layer with the same parameters but a bidirectional encompassing of the LSTM cells. However, this performed slightly worse than the vanilla LSTM network, and we presume this is likely due to the addition of more parameters to the network which led to overfitting and a lower final validation accuracy after 3 epochs.

3.2.2. Bidirectional HAN-GRU Network

Another baseline model we looked at was a re-implementation of Tsaptsinos' HAN-GRU network, shown in figure 2 (Tsaptsinos, 2017). In this architecture, the idea is to pay particular attention to certain words and lines which are important in inferring the genre of a certain song. The same method of attention is applied twice, once on the word level and once on the line level. In particular, first words are embedded using a standard embedding matrix approach. For each line, each word in the line is fed as input into a bidirectional GRU. The hidden states for these words are attended to as proposed by Bahdanau et al. (Bahdanau et al., 2014). These weighted hidden states for each line are fed to another bidirectional GRU layer, this time generating hidden states for each line. The attention mechanism is applied to the output of this bidirectional GRU layer. We then apply a standard feedforward layer and softmax, and attempt to minimize the categorical cross entropy loss for genre classification. This model performs slightly worse than the vanilla LSTM baseline model, but would also prove to be the key model underlying our final architecture.

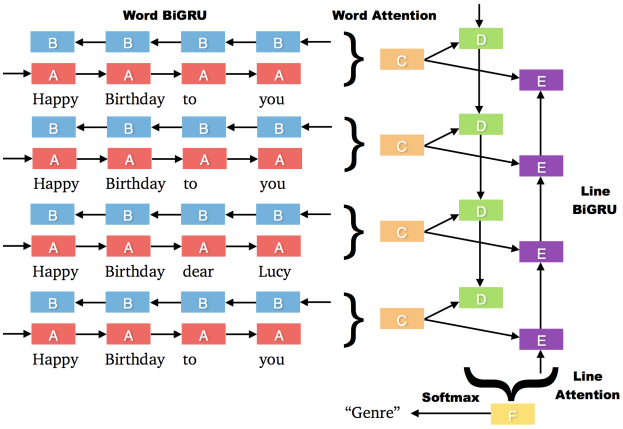

Figure 2. Bidirectional GRU Network Architecture with Hierarchical Attention, image courtesy of Tsaptsinos.

3.3. HAN-LSTM Network

Naturally, observing the dual successes of the vanilla-LSTM and the HAN-GRU network, we sought to combine the approaches. The HAN-LSTM network has exactly the same architecture as Tsaptsinos's HAN-GRU net, except that the recurrent cell is an long short-term memory cell rather than a gated recurrent unit.

3.4. Transformer Network with Multihead Attention

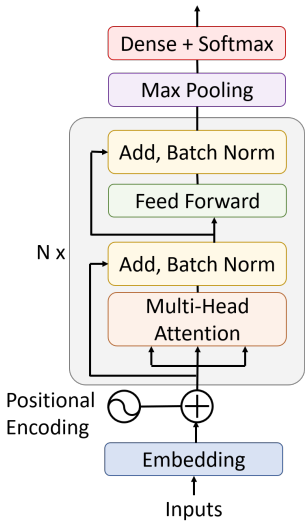

Figure 3. Modified Transformer network with added Max Pooling and Dense layers, with a Softmax activation to select the label.

The original transformer architecture was designed with the purpose of achieving machine translation by Vaswani et. al., so it had a distinct encoder-decoder structure (Vaswani et al., 2017). For our purposes, we applied multiheaded self-attention on the words for each song for classification, so we implemented the encoder self-attention part. For our attention mechanism, we used scaled dot-product attention and also used positional encoding like Vaswani et al. in order to take into account the positions of the words within the song lyrics. On top of that, we added a max pooling layer to reduce the dimension of the output of the transformer and a dense layer with a softmax activation to finally find the corresponding class, which is shown in figure 3.

3.4.1. Hyperparameter Tuning

One of the most interesting and difficult characteristics of the transformer architecture is the massive space of hyperparameters that required tuning. For example, some of the hyperparameters that needed to be tuned include the number of heads in the multi-headed attention, the number of transformer blocks, the dropout rate for the dropout layer of the transformer block, the maximum song lyric length to be used, the word embedding length, and the batch size. Because of the vast space, hyperparameter tuning was a big part of our struggle in training the transformer network architecture and involved many days of training in order to optimize the parameters to get the best test accuracy.

Our choice for batch size, and partially for the number of attention heads, was limited by the amount of memory able to be allocated on the GPU regardless of performance. In general, we deemed a small batch size to be acceptable because conventional wisdom holds that such a choice results in models that tend to generalize better (Hoffer et al., 2018).

We initially tried 6 layers for the transformer, achieving around 58.85% accuracy after 3 epochs of training. The original Transformer paper motivated this choice, however our group theorized that perhaps the data was not sufficient for the excessive number of parameters (such as in the case of the bidirectional LSTM and the LSTM with HAN) (Vaswani et al., 2017). Thus we proposed reducing the number of layers and saw improved test accuracy with a 3-layer transformer.

We also tried reducing the maximum number of words each song could be. We found that reducing the maximum length from 600 to 400 words substantially improved training time, from many hours for a single epoch to fewer than two hours for all three. Test accuracy, however, decreased to 59.78%. Although this was not the best result, and somewhat expected, the relatively small decrease in accuracy raised interesting questions about the value of the later words in a song for classifying genre, as well as tradeoffs between accuracy and training time. The reduced training time also allowed for more experimentation with other hyperparameters.

We found that two attention heads performed the best empirically, with smaller (1) and larger (4, 8, 10) numbers of heads resulting in decreased performance for the 3-layer model. In addition, the dropout rate was also significant in improving the test accuracy as it acted as a regularizer for the 3-layer model and ensured that the model did not overfit the training data. Thus, a dropout rate of 0.4 performed optimally for the 3-layer model. Putting all of the conclusions together, our best test accuracy was achieved with a 3-layer model with 2 heads, a maximum length of 400 words, and a dropout rate of 0.4 with an embedding vector size of 96 (determined through a simple linear search to be optimal) at 61.76%. This likely struck the right balance of expressibility and avoiding overfitting for the model.

3.5. Final Model

3.5.1. Bidirectional HAN-GRU Network with Layer on Inter-Line Level

The winning model was an iteration on the HAN-GRU network, in which another layer was added on the inter-line level 4. We believe that this allows the network to learn more refined understanding of the high-level song structure. However, adding another layer on the intra-line level decreased performance, perhaps because lines are already short enough that adding another layer to attend to a single line is over-parameterizing.

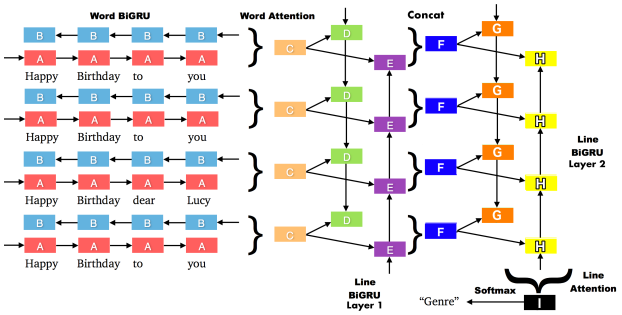

Figure 4. Extension of Tstaptsinos' architecture with added line-level attention

3.6. Other Considerations

3.6.1. Upsampling to Correct Dataset Class Imbalance

One of the things we noticed about the dataset is the large imbalance of genres, specifically with about 90000 of the 250000 data samples being of the genre Indie. When evaluating our models, we looked at potentially modifying our data preparation technique in order to up-sample the genres so that there was an equal number of each genre in the training set. Note that the upsampling was only applied to the training set and was performed only after the train-test split was done.

The upsampling was done by randomly sampling with replacement from the training set until each genre was equally represented. However, the upsampling approach proved to be unsuccessful, especially when the transformer architecture was trained on the upsampled dataset, most likely indicating that the prior that is represented by the training set is a significant part of this dataset and cannot be simply neglected or removed to achieve good test accuracy.

4. Results

The baseline LSTM model achieved surprisingly good results, with a test accuracy of 62.28%. Our first attempt to iterate on a baseline architecture design was to make the LSTM bidirectional, which actually yielded a small loss in performance to 62.13%.

The second baseline, Tsaptsinos' HAN-GRU architecture, was also a high performer and achieved 61.89% accuracy. This was further improved by adding another layer on the inter-line level to beat the baseline performance and yielded a 62.58% accuracy. This turned out to be the best performer over all the architectures. We also tried adding another layer to the HAN-GRU on the intra-line level yielded a slight loss in test accuracy.

We tested the idea of using an LSTM with hierarchical attention, but the HAN-LSTM model performed very poorly after 3 training epochs, with only 49.08% accuracy on the test set. We suspect that 3 epochs is too few, and our training data too small, in order to take advantage of a complex, multi-layered LSTM net. We noted that the training accuracy plateaued late in the very first epoch, which is a symptom of LSTM networks' well known problem with vanishing gradients.

Finally, we tested a modified version of the Transformer architecture on this problem. Given the success of attention in HAN-GRU, it seemed natural that a fully attention-based model would also perform well. We were able to achieve a 61.76% test accuracy after tuning the hyperparameters—certainly good results, but still beat by the HAN-GRU. This was surprising, especially because of how well self-attention worked in the original paper. However, the original paper applied the Transformer to translation between sentences, so the sentences in their dataset were much shorter than the songs in our dataset. Furthermore, they were able to apply 8-headed attention to the sentences, more than we could due to hardware limitations. The combination of having to use fewer heads for word-level attention, and the fact that our songs had many more words, may have limited the effectiveness of the Transformer. The best results for each class of architectures are shown in table 1.

Vaswani et al. cite one of the benefits of self-attention over attention in recurrent networks as being the lowering of the upper bound on path lengths representing long-range dependencies of parts of the input sequence to other parts (Vaswani et al., 2017). A conclusion that we can draw from the relative success of the LSTM and HAN-GRU models is that the lengths of songs is not suited for self-attention to order to reduce training

Table 1. Classification accuracies for various neural models on the MetroLyrics Dataset. Specific approaches and architectures for models detailed in the Approach section.

Model	Accuracy	Parameters
LSTM	62.28%	3,039,511
Bidirectional LSTM	62.13%	3,078,811
HAN-GRU	61.89%	1,051,775
Modified HAN-GRU	62.57%	1,097,075
HAN-LSTM	49.08%	1,075,175
Transformer	61.76%	3,198,107

time, nor to bound the length of long-range dependency paths. Of the lyrics in the dataset, the average number of words in a song was 231, and the average number of lines was 34. Here, we can hypothesize that perhaps one of the benefits of using hierarchical attention is that long-term dependencies can be easily accounted for within the length of a line, but less easily over the length of an entire song.

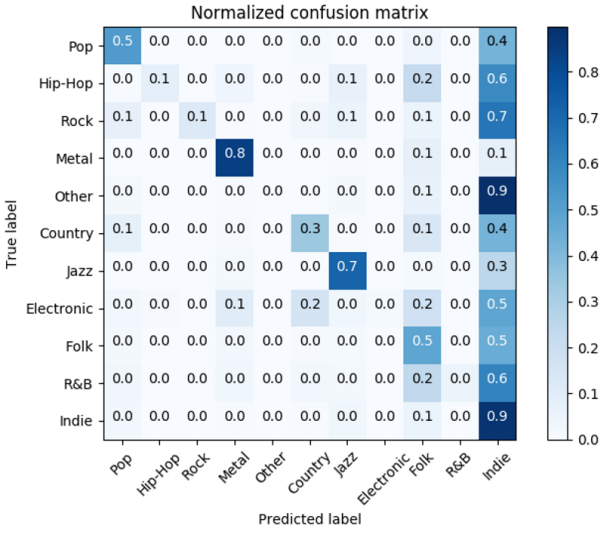

Figure 5. Modified HAN-GRU confusion matrix

The confusion matrix also indicates some insights into the performance of the model, what it learns, and properties of the underlying data (see figure 5).

The accuracy for "Indie" music is very high, and many errors in the model seem to lean towards classifying another genre as "Indie" music. Examining the data, this makes sense - it turns out that the "Indie" genre dominates the dataset, and thus has a strong prior. This is further complicated by the nature of the genre classification problem itself - "Indie" music is defined primarily by the fact that it is published under independent labels, and although it does have its own stylistic cues, is a much broader and flexible genre than the others. For example, it seems like a particularly forgivable error that "Other" music is often misclassified as "Indie". Perhaps the performance of the model reflects the subjectivity of the original genres rather than the inherent difficulty of the problem.

It is worth noting the model excels at classifying "Jazz" and "Metal". These genres have distinctive lyrical styles to them, at least from human musical experience, compared to others. The model also does reasonably well on classifying "Pop" and "Folk" music, although again it often misclassifies songs in these genres as "Indie".

The low performance on "Hip-Hop" is surprising, given the distinctiveness of the genre similar to "Jazz" and "Metal", although we suspect it is related to the use of slang and alternate spellings in hip-hop that is not captured in the word embedding model. We suspect a better embedding model would improve performance on "Hip-Hop".

Overall, the model seems to classify certain distinctive genres well, while also learning the priors and characteristics of the underlying data. Even some of its errors seem more like misinformed attempts than blind mistakes. In addition, our model performs well when compared to past literature; on 117 genres, Tsaptsinos' HAN-GRU architecture achieved a 46.42% accuracy and on 20, an LSTM was the best performer at 49.77% (Tsaptsinos, 2017). Our model also outperforms the non-neural techniques used by Canicatti, who was able to achieve a 52% accuracy on 4 genres (Canicatti, 2016).

5. Tools

The primary tools used for this research study were Python, Keras with a TensorFlow backend, iPython, and Scikit-Learn. In addition, for some of the data pre-processing we used the Google Sheets and Apps Scripts API in order to filter out song lyrics that were non-English, specifically the `DETECTLANGUAGE` function of the API.

In terms of hardware, we primarily used a team member's laptop GPU as well as the hive cluster GPUs, as well as just our CPUs for some of the faster models. The hive GPU was an NVIDIA GeForce GT 740 which has 993 CUDA cores, a base clock of 993 MHz, 2GB of memory, and a memory bandwidth of 28 GB/s. The laptop GPU was an NVIDIA GeForce GTX 950M, which has 640 CUDA cores, a base clock of 914 MHz, 4GB of memory, and a memory bandwidth of 32 GB/s.

6. Lessons Learned

We learned several lessons over the course of this project applicable to other projects. We gained an appreciation for the long training times of machine learning models, which complicates efforts such as debugging and hyperparameter tuning. We also learned to be wary of overparameterizing and to educate ourselves on the characteristics of the underlying data in designing and assessing the model. Conventional wisdom about the training efficiency of non-recurrent models and improved performance of newer architectures did not pan out, in the case of the self-attention transformer, for the particular problem of classifying song genres based on the lyric lengths in this project. In terms of results, we learned that hierarchical attention is a powerful technique for lyric-genre classification, and can be extended from the original design and application in Tsaptsinos (Tsaptsinos, 2017).

7. Team Contributions

A lot of the work was done in pairs and as a group, so these are only approximate divisions of the work. We all put in an equal amount of effort into both the actual research of this project as well as the development of project requirements like the presentation, poster, and report.

- Connor Brennan (25%)
  - HAN-GRU and modified HAN-GRU code implementation
  - HAN-LSTM code implementation
  - Work on final presentation, poster, report
- Sayan Paul (25%)
  - Dataset pre-processing
  - Transformer code implementation and hyperparameter tuning
  - Local training on GPU
  - Work on final presentation, poster, report
- Hitesh Yalamanchili (25%)
  - Dataset pre-processing
  - Transformer code implementation and hyperparameter tuning
  - Upsampling, baseline bidirectional LSTM implementations
  - Work on final presentation, poster, report
- Justin Yum (25%)
  - Data preparation for model training, training on GPU (Hive)
  - Initial LSTM code implementation
  - Transformer hyperparameter tuning
  - Work on final presentation, poster, report

8. Source Code

The code we used to train and evaluate our models is available at `https://github.com/hiteshyalamanchili/SongGenreClassification`.

References

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.

Canicatti, A. Song genre classification via lyric text mining. *International Conference on Data Mining*, pp. 44–49, 2016.

Gossi, D. and Gunes, M. H. Lyric-based music recommendation. In Cherifi, H., Gonçalves, B., Menezes, R., and Sinatra, R. (eds.), *Complex Networks VII*, pp. 301–310. Springer, 2016.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks, 2018.

Holmdahl, R. L. Song title prediction with bidirectional recurrent sequence tagging, 2017.

Tsaptsinos, A. Lyrics-based music genre classification using a hierarchical attention network. arXiv:1707.04678, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. arXiv:1706.03762, 2017.