An embedding for melodies generated by a Recurrent Neural Network

Eric Wolf
Department of Computer Science
Tsinghua University/ETH Zurich
erwolf@student.ethz.ch

Abstract

I extend existing recurrent neural network models for composing sheet music to also learn the initial states of LSTM cells to learn an embedding for the training set. In this paper, I evaluate this embedding and reuse it for melody generation.

1 Introduction

Machine learning in music generation was an almost undeveloped domain until Google took on the field this year with Project Magenta mag [a] and filled the gap of effectively using neural networks for composition. They implemented a number of interesting models and encodings that were so far blatantly lacking research. However, their generative model only learns note-to-note predictions and therefore needs to be supplied with a starting sequence in order to generate a melody. I remove this dependency by learning the initial states used in the LSTM cells and thereby train an embedding of the training set.

2 Related works

First attempts to train an RNN on music go back as far as 1994, when Mozer [1994] achieved modest success training a small recurrent neural network without LSTM gates on classical music. In order to generate harmonically sounding composition, he tried to model similarly sounding notes as similar representations in the feature space.

More recently, Eck and Schmidhuber [2002] used LSTM gates to compose music using a relatively simple RNN, encoding notes in a binary vector which allows for multiple notes to be played simultaneously. However, their representation is still very limited. It does not, for example, differentiate between multiple short notes and one long note of the same pitch.

Others have tried to reuse a character-based RNN to generate music lis. This model is simple to set up, but has severe limitations as it does not even necessarily output a valid rhythm.

A completely different direction is taken by Google's DeepMind team. Their primary objective for WaveNet van den Oord et al. [2016] was the development of a text-to-speech system by modeling sound as raw audio waveform. Their results are remarkable, but require very expensive computations. As a byproduct, WaveNet was also trained on music and produced reasonable-sounding results of short duration. However, as music generation was not DeepMind's main focus, it is hard to judge WaveNet's performance on music generation in comparison to other publications.

A few months ago, Google's Brain team has released their TensorFlow Magenta framework mag [a], which is intended for artistic uses of machine learning, especially in the field of music generation. They combined most of the ideas found in previous works and augmented them with state of the art Machine Learning techniques used on other datasets. For example, they have tried to use an attention mechanism to solve the problem of modeling long-term dependencies, as first introduced

by Bahdanau et al. [2014] that lets the RNN learn to use a weighted vector of the previous RNN outputs as an additional input. The Google Brain team also attempted to enforce harmonic constraints dictated by music theory using reinforcement learning.

Only two papers have been released by the Magenta team so far: one on reinforcement learning Jaques et al. [2016] and one on Gibbs sampling Huang et al. [2016], which is, however, not part of the main functionality of the openly available code. No official evaluation of the results has been published yet, perhaps due to the hard-to-evaluate nature of such a generative model. However, samples published on their blog sound reasonably well and Project Magenta has won the Best Demo award at NIPS 2016. Some informal explanations and demonstrations are available on the Project Magenta blog mag [b].

Magenta is an open-source project that intends to invite everyone who is interested to discuss or contribute. The goal is an interdisciplinary synthesis between computer scientists and artists to develop tools that support artists.

3 Dataset

The Lakh MIDI dataset Raffel [2016] comprises about 170.000 MIDI files. This means it is larger than the datasets used in most previous publications and should be sufficiently strong to train a neural network with millions of model parameters.

A subset of 45.000 files from the Lakh MIDI dataset have been matched to entries in the Million Song Dataset Bertin-Mahieux et al. [2011a]. While the Million Song Dataset does not contain genre annotations, the features included in the dataset allowed others to build genre classifiers on top of it. Therefore, it is not too hard to find genre-annotated subsets of the Million Song dataset. One such dataset is provided by the ground truth used in Schreiber, which links entries in the Million Song Datasets to date from the Last.fm dataset Bertin-Mahieux et al. [2011b], the Top-MAGD dataset and the beaTunes Genre Dataset (BGD).

4 Overview of Project Magenta

Since the authors of Project Magenta have not released any papers detailing their model and data preprocessing pipeline, and because modification of these modules was an integral part of this project, I would like to elaborate on their implementation in the following paragraphs. Most of these information was extracted laboriously from the source code, as the documentation offered usually was limited to usage instructions and did not allow for a scientific understanding of the methods used. Since Project Magenta is an open source project, its codebase and features are constantly undergoing changes, as can be observed on their GitHub repository mag [c]. The following documentation is to my best knowledge at the time of my writing.

4.1 Models

The Google Brain team has experimented with various types of models for music generation. Notably, each model is relatively tightly coupled with its preprocessing pipeline, which is explained below. The following models are of interest:

- **Drums RNN** This model is used exclusively for the generation of drum tracks. As drum tracks are non-melodic, their modelling has been decoupled from that of melodies. Interestingly, polyphony has been modelled by a one-hot encoding of the powerset of all 9 pieces available in the modelled drum kit, leading to a size 512 binary vector, which demonstrates the difficulty in accurately modelling polyphonic data.
- **Melody RNN** This is the main model for melody generation. It encompasses several different configurations:
  - **Basic RNN** A regular LSTM network using a one-hot encoding.
  - **Lookback RNN** An extension of the basic RNN that also includes notes that have been played exactly n bars ago in the input vector.
  - **Attention RNN** A further extension of the above that also uses previous outputs as an input for an attention mechanism.

- **Polyphony RNN** In the very recently released polyphonic model, polyphony is modeled by using a variable number of RNN timesteps for one actual timestep, using each step to start or continue a certain note. Physical timesteps are separated by a special marker.

4.2 Pipelines

As mentioned before, each model requires different input feature. Therefore, detailed information about model parameters is often required already at the preprocessing stage. The process of converting a MIDI file into input sequences is far from lossless - MIDI is a very broad format as will be explained in the section below. Furthermore, the mapping of midi files to input records is not one-to-one, in fact, usually multiple melodies are extracted from each input file.

4.2.1 The MIDI format

MIDI files are commonly used not only in manual composition but also to transfer music data in real-time between instruments and music-processing devices. While MIDI files distinguish notes and similar events, the timing is therefore continuous and has to be quantized to be embedded in a sheet music rhythm scheme (see below). Besides note information, MIDI files can also contain pitch bend information, very generic control information and tempo information. MIDI files used in composition may contain key and time signatures. The MIDI format therefore is very flexible, allowing users to specify human-readable annotations if needed, but not necessarily.

4.2.2 Quantization

After an initial conversion of MIDI or MusicXML files into a TensorFlow data format, Project Magenta proceeds by quantizing the notes. At this stage, files with time signature or tempo changes are rejected. Then, all notes are aligned at the nearest quantization boundary, by default 1/16 of a bar. Control information and pitch bends are dropped.

4.2.3 Melody extraction

This stage extracts monophonic melodies from polyphonic and multi-instrumental sequences. In order to discord polyphony within a single instrument, only the note of highest pitch is kept at any given quantization time step. Periods of silence of one bar or longer are used to separate melodies within one instrument. Eventually, melodies that are shorter than 7 bars or that use less than 5 unique pitches are discarded. The resulting melodies are stored as a sequence of time steps, each representing the start of a note of a certain pitch, the end of a note, or no event.

4.2.4 Encoding

The most basic encoding is a simple one-hot encoding of the events explained above. When using lookback, this is augmented by a one-hot encoding of preceding events and a by a binary counter of the position in the current bar as well as binary values indicating if the events at the fixed lookback positions have been repeated. Lastly, the Attention RNN uses a key melody encoding, which also includes estimates of the current key.

4.3 Evaluation

From the samples on the Project Magenta website, it is not clear which model performs best. Since I wanted to closely work with the individual melodies in the training set, I extracted those melodies and found that the key melody encoding transformed the melodies in a way that made them audibly less pleasing, even though most of the information may have been retained. In order to have a better starting point for generation, I thus proceeded to use the Basic RNN model.

5 Learning initial states

The model as developed by the Google Brain team always initialized the LSTM cells' initial state with zeroes and does not train these values. I extended their model by not only learning the initial states, but learning them for every training example individually. Since the learned values are directly

tied to the training set, they cannot be used for melody generation in general. They do, however, represent an embedding of the melodies in the training set. In the following, I analyze their predictive power and what happens if trained initial states are used for generation.

As Magenta was designed for generation of melodies with a priming melody, I had to change the generation process as well as the training process. I included a special event, MELODY_START, that is supplied to the RNN input on the first step.

In order to inspect the data, I wrote scripts to extract the embedding from the model and to extract melodies from the training set and convert them back into MIDI.

6 Performance

Training values for every training example posed memory issues during training. Even when using only the matched subset of the Lakh dataset and only two layers of 64 LSTM cells each, a matrix of 111180x2880 floating point values needed to be trained. The Adam algorithm keeps two copies of every trained variable in memory, leading to a total memory consumption of about 960MB only for the initial states. Since millions of other model parameters also had to be trained, allocating such large contiguous chunks of memory caused TensorFlow to fail. Therefore, I developed a mechanism to swap out model parameters and the associated Adam variables during training from GPU to main memory. Additionally, the matrices are stored as memory-mapped numpy matrices in main memory, which means that the main memory they occupy should not become problematic, as the access pattern is relatively sparse but random.

Performance could be further optimized by queueing the transfers to the GPU such that TensorFlow can schedule them in advance, as is already done for the constant training data.

7 Classification

I hypothesized that the embedding generated could be used for genre classification. This classification task, however, is not so straightforward as a variable number of melodies belong to each song. I attempted both predicting the genre for each melody and aggregating these predictions into a single prediction for the song.

For the latter, the class probabilities for each melody prediction were added and normalized. However, this turned out to perform much worse in practice. I suspect that this problem arises because the aggregation was only reflected in testing, but not in training - as the contribution to the loss of a wrongly predicted melody might be smaller when the loss is measured over a sum of melodies, hard-to-predict melodies such as background melodies might artificially render the classification problem more difficult while not contributing to the actual classification.

During preprocessing, the Lakh dataset, the Million Song Dataset and the genre dataset were combined. On one test dataset of 45.129 songs and 100.009 melodies, 9.037 songs and 37.763 melodies could be assigned a genre from the dataset. There were 2880 features learned features for each melody and the melodies were divided into 15 different genres.

Table 1: Classification results

Attention RNN				
Classifier	Accuracy	Precision	Recall	F1
RandomForest	0.436	0.484	0.436	0.332
AdaBoost	0.387	0.250	0.387	0.230
RandomForest + PCA	0.411	0.417	0.411	0.305

For classification, grid search was run over Random Forests with different numbers of estimators and gini as well as entropy criteria, AdaBoost with various numbers of estimators, logistic regression with different values for the C parameter and k-Nearest-Neighbors with different k. The results were only slightly detoriarated when reducing the number of features from 2880 to 128 using PCA, however, for comparability, all methods were also trained on the full set of features.

Although the results are not competitive with algorithm specifically aimed at genre classification, the scores clearly show that some genre information can be deduced from the embedding. Considering that due to the splitting of songs into different melodies, some of which, e.g. background tunes, may not even be identifiable as belonging to a certain genre even by a human listener, this is remarkable and demonstrates that the learned embedding does indeed represent the learned melodies.

8 Generation

To use the embedding for generation, the learned initial states for a specific melody from the training set were loaded into the model, followed by a MELODY_START tag as first input. No starting melody was primed. Alternatively, two melodies were combined by feeding the average initial state of those two melodies into the model.

The evaluation of generated tunes is, of course, subjective. First of all, melodies generated from the initial states of one or two trained melodies were of similar quality as those generated by the Project Magenta team. Especially the fact that combining different points in the embedding space yields usable results indicates that the model did not overfit on individual points in the training set, but can also deal with previously unseen data.

The generated melodies were not as clearly discernible as being generated from the same or different initial states as I initially hoped. An inspection of the training revealed that, due to the loss during preprocessing, many of the training melodies did sound like full songs, but, due to merely being part of a song, sounded more like improvisation. This could explain why the model is not perfectly trained on capturing a song's characteristics.

Another contributing factor is probabilistic softmax sampling, which encourages high diversity in the generated results. Decreasing the softmax temperature leads to more similar results, but the RNN often gets 'stuck' in a loop, causing the result to be less melodical - this is a known problem when decreasing the softmax temperature and suggests that the RNN model itself could be further improved with a better loss function.

9 Training

The neural networks were trained on Amazon WS g2.xlarge instances with Nvidia GRID K520 GPUs with 4GB VRAM, 16GB RAM and 8 cores. Classification tasks were run on ETH's Euler cluster whose nodes have 24 cores and 64GB RAM each.

Training on Attention RNN with batch size of 256 and layer sizes of 2x64:

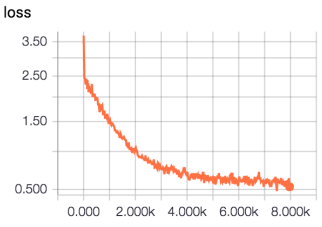

Figure 1: Training loss over the course of 13 hours

Figure 2: Training accuracy

Figure 3: Training speed

Acknowledgments

I want to thank the Google Brain team working on Project Magenta for replying to my questions on their mailing list.

10 Conclusion

I have shown that obtaining an embedding space by learning the initial states of an LSTM for each item in the training set is feasible and leads to features that can be used in predicting genre information.

This is in spite of the fact that these parameters can only be trained once per epoch, as only one item in the training set contributes to one row in the embedding matrix. I believe that the embedding could be further improved by adapting the preprocessing pipeline to extract a melody that is perceived by the listener as the main melody of the piece, and thus more clearly represents listeners' perceptions of music or by further improve the RNN model to also work at lower softmax sampling temperatures.

References

Magenta: Make music and art using machine learning. `https://magenta.tensorflow.org`, a.

Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE, 2002.

"lisl's stis": Recurrent neural networks for folk music generation. `https://highnoongmt.wordpress.com/2015/05/22/lisls-stis-recurrent-neural-networks-for-folk-music-generation/`.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. *CoRR*, abs/1611.02796, 2016. URL `http://arxiv.org/abs/1611.02796`.

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *Under review as a conference paper at ICLR 2017*, 2016. URL `https://openreview.net/pdf?id=r1Usiwcex`.

Magenta: Blog. `https://magenta.tensorflow.org/blog/`, b.

Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, COLUMBIA UNIVERSITY, 2016.

Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, volume 2, page 10, 2011a.

Hendrik Schreiber. Improving genre annotations for the million song dataset.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011b.

Magenta: Github. `https://github.com/tensorflow/magenta`, c.