Project on Application of Machine Learning Approaches

Music Generation using LSTM

Divyam Kaushik

MCA-III

Roll No. 19

Introduction

Music is not just an art, music is an expression of the human condition. When an artist is making a song you can often hear the emotions, experiences, and energy they have at that moment. **Music connects people all over the world and is shared across cultures.** So there is no way a computer could possibly compete with this right? That's the question I asked myself when I chose my semester project for our Machine Learning class. My goal was to create something that would make the listener believe that what they Ire listening to was created by a human. I think I succeeded personally, but I will let you be the judge.

Music composition is a vital task that artists have used to express themselves since the birth of civilization. In the field of artificial intelligence, designing algorithms that produce human-level art and music is a complex but fascinating and satisfying challenge. Recent breakthroughs in neural networks have aided us in moving away from defining music composition rules and toward developing probabilistic models that learn empirically-driven rules from a large collection of previous music. Furthermore, deep learning systems do not provide direct methods for controlling generation (e.g., imposing some tonality or other arbitrary constraints). Again, **deep learning architectures are autistic automata that make music autonomously without human user input**, which is far from the goal of enabling musicians to develop and refine music interactively.

Challenges: Our goal is to be able to build a generative model from a deep neural network architecture to try to create music that has both harmony and melody and is

passable as music composed by humans. Also knowing/learning the libraries Keras, TensorFlow, and music21 is another challenging part.

Issues: Previous work in music generation has mainly been focused on creating a single melody. More recent work on polyphonic music modeling, centered around time series probability density estimation, has met some partial success. In particular, there has been a lot of work based on Recurrent Neural Networks combined with Restricted Boltzmann Machines (RNNRBM) and other similar recurrent energy-based models. Our approach, holver, is to perform end-to-end learning and generation with deep neural nets alone.

Related Work From Paper

In the paper, Briot and Pachet focused on issues such as control, structure, creativity, and interactivity. In this work, the authors selected several drawbacks of a direct application of deep learning to music generation and examined why the difficulties are not met and how various techniques can overcome them.

Challenges:

1. **Control**

   Musicians regularly seek to adapt notions and patterns borrold from other contexts to their own goals, such as transposition to another key or reducing the number of notes. Control is the ability to direct the generation of a deep learning architecture. Some strategies to control the network generation are shown below:

   i. Dimensions of control strategies are Input, Output, and Encapsulation/ reformation

ii. Sampling a model to generate content may be an entry point for control if I introduce constraints on the output generation.

iii. The strategy of conditioning (sometimes also named conditional architecture) is to condition the architecture on some extra conditioning information,

iv. Input manipulation is the idea that states that the initial input content, or a brand new (randomly generated) input content, is incrementally manipulated in order to match a target property.

v. The strategy of reinforcement is to reformulate the generation of musical content as a reinforcement learning problem, while using the output of a trained recurrent network as an objective and adding user-defined constraints, e.g., some tonality rules according to music theory, as additional objective.

2. Structure

Another challenge is that most of the existing systems have a tendency to generate music with “no sense of direction.” In other words, although the style of the generated music corresponds to the corpus learned, the music lacks some structure and appears to wander without some higher organization, as opposed to human-composed music which usually exhibits some global organization (usually named a form) and identified components

3. Creativity

The issue of the creativity of the music generated is not only an artistic issue but also an economic one, because it raises a copyright issue. There are two known approaches to resolve this is priori and posteriori.

4. **Interactivity**

   In networks, the generation is automated, with little or no interactivity. As a result, local modification and regeneration of musical content are usually not supported, the only available option being a whole regeneration (and the loss of the previous attempt). This is in contrast to the way a musician works.

TODO: Explain the methodology opted by each research paper with advantage(s) and disadvantage(s)

My Implementation : Problem Statement and Background

This project aims at building a Deep Neural Network (deep learning) to generate melodies using TensorFlow and Keras and handle melodies as time-series data with the help of a Research Paper which will be summarized in this project report throughout.

In order to create music, I needed some way to learn the patterns and behaviors of existing songs so that I could reproduce something that sounded like actual music. Before I go into the details of the implementation there is some terminology that I should clarify to understand the paper.

Recurrent Neural Networks (RNN)

A recurrent neural network is a class of artificial neural networks that make use of sequential information. They are called recurrent because they perform the same function for every single element of a sequence, with the result being dependent on previous computations. Whereas outputs are independent of previous computations in traditional neural networks. In the Implementation part of it, I will be using a Long Short-Term Memory (LSTM) network. They are a type of Recurrent Neural Network that can efficiently learn via gradient descent. Using a gating mechanism, LSTMs are able to recognize and encode long-term patterns. LSTMs are extremely useful to solve problems where the network has to remember information for a long period of time as is the case in music and text generation.

Music21

Music21 is a Python toolkit used for computer-aided musicology. It allows us to teach the fundamentals of music theory, generate music examples and study music. The toolkit provides a simple interface to acquire the musical notation of MIDI files. Additionally, it allows us to create Note and Chord objects so that I can make our own MIDI files easily. I will use Music21 to extract the contents of our dataset and to take the output of the neural network and translate it to musical notation.

Keras

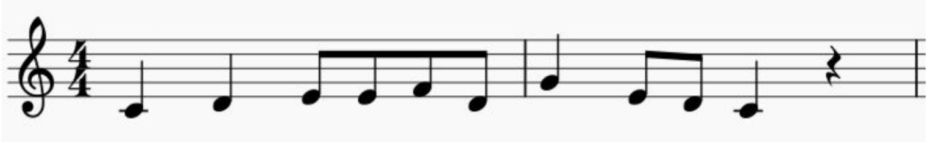
It is a high-level neural networks API that simplifies interactions with Tensorflow. It was developed with a focus on enabling fast experimentation. I will use the Keras library to create and train the LSTM model. Once the model is trained I will use it to generate the musical notation for our music.

Melody and Time Series

A melody is a collection of musical tones that are grouped together as a single entity. Most compositions consist of multiple melodies working in conjunction with one another. In a rock band, the vocalist, guitarist, keyboardist, and bassist are all playing melodies on their respective instruments. Even the drummer is playing one.

The melody in a piece of music consists of two primary components:

1. **Pitch**. This refers to the actual audio vibration produced by an instrument. These pitches are arranged as a series of notes with names like C4 or D#5.
2. **Duration**. The definition of melody also includes the duration of time that each pitch will sound. These durations are divided into lengths such as whole notes, half notes, quarter-note triplets, and more.



A time series is a sequence of data points that occur in successive order over some period of time. This can be contrasted with cross-sectional data, which captures a point in time.

Simply put, Melody is a sequence of notes and rests and I will treat melody as a time series and will predict the next time series. Time series is a data structure where I have samples that are taken at equally spaced positions in time.

Midi Data and using it in my codebase

Midi data Midi files are structured as a series of concurrent tracks, each containing a list of meta messages and messages. I extract the messages pertaining to the notes and their duration and encode the entire message as a unique token.

Dataset Used for our Model Training

Dataset (https://kern.humdrum.org/cgi-bin/browse?l=essen%2Feuropa%2Fdeutschl), It consist of various songs in the form of German Folk melodies in krn format. Out of which I have only used erk/ folder for model training.

Preprocessing of Midi Data(preprocess.py)

For every song and filtering out songs that have non-acceptable durations, I are transposing, i.e. get the key from the song and according to the mode, converting it to C-major(mode = major) or A-minor(mode = minor), every node in a song and encoding it to time-series representation using preprocess(dataset_path) function in our code.

Code will generate the numerous files of encoded songs into the dataset/ folder present at the root of our project present at Github.

Then I used this generated folder to create a single file of all the encoded songs using create_single_file_dataset(dataset_path, file_dataset_path, sequence_length), and further converted them into sequences for training the model using generate_training_sequences(sequence_length).

Output of running preprocessing Code:

1. Dataset Folder at the root of project
2. file_dataset file at the root of project.

Training our Model (train.py)

Used **single layered LSTM** and dropout layer to prevent overfitting using tenserflow and keras and then compile the model with **sparse_categorical_crossentropy** and then trained the model over the sequence that I created in last step, and then I saved the model as model.h5` file.

Use of Generated Model (meldoyGenerator.py)

After generating the model in last step, I generate a melody using saved model (model.h5) and returns a melody and then I saved that melody by converting it into a midi file using save_memody().

Methodology

I used a 1-layered Long Short Term Memory (LSTM) recurrent neural network (RNN) architecture to produce a character level model to predict the next note in a sequence. I create an embedding matrix which maps each token into a learned vector representation. A sequence of tokens in a piece is then concatenated into a list of embedding vectors that forms the time sequence input that is fed into the LSTM. The output of the LSTM is fed into softmax layer over all the tokens. The loss corresponds to the cross entropy error of our predictions at each time step compared to the actual note played at each time step. Our architecture allows the user to set various hyperparameters such as number of layers, hidden unit size, sequence length, batch

size, and learning rate. I clip our gradients to prevent our gradients from exploding. I also anneal our learning rate when I see that the rate that our training error is decreasing is slowing. I generate music by feeding a short seed sequence into our trained model. I generate new tokens from the output distribution from our softmax and feed the new tokens back into our model. I used a combination of two different sampling schemes: one which chooses the token with maximum predicted probability and one which chooses a token from the entire softmax distribution. Our deep learning implementation was done in TensorFlow.

Results

Our Model got an accuracy of 91.42% and generated the music that has both harmony and melody and is passable as music composed by humans. I trained this midel for 50 Epochs with a learning rate of 0.001, and it took a time of 2.5hr (approx) on my architecture.

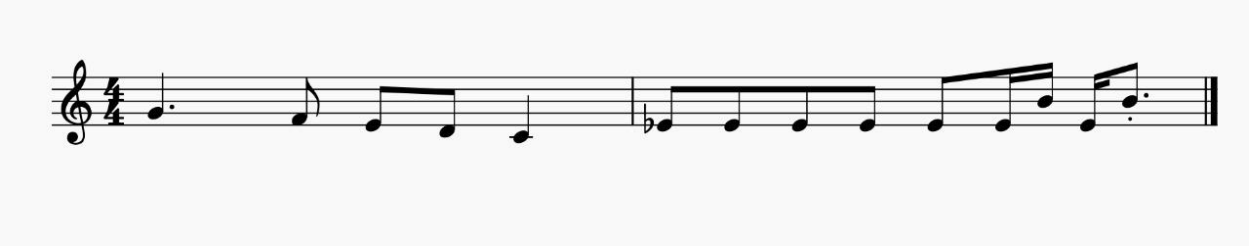
```
Epoch 50/50
5660/5660 [==============================] - 39s 7ms/step - loss: 0.2500 - accuracy: 0.9142
```

Summary

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None, 38)]        0

 lstm (LSTM)                 (None, 256)               302080

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 38)                9766

=================================================================
Total params: 311,846
Trainable params: 311,846
Non-trainable params: 0
```

Experimental Results and Discussion

I had our untrained model generate sequences. As you can see in the image below, it tried to mimic the previous node again and again.



Some possible ideas/experiments include:

- Take note duration as a separate input to the network rather than treating each pitch/duration separately.

- Expand to multiple instruments.
- Move away from midi files and produce/learn from actual MP3s.
- Learn the time step, sequence length, and time signature.
- Introduce randomness to emulate “human error/experimentation”.
- Allow for multiple keys.

Limitations

While being reasonably satisfying, the above examples of generated tracks also highlight the main limitations of our models:

- Not-that-good generalization: the outputs strongly depend on the chosen seeds and sometimes the models produce very few notes, or even complete silence, or noisy outputs
- Difficulty to generate long musical tracks: our models have trouble when I ask them to generate tracks exceeding the length of the training sequences (here 10s). They have a tendency to “dry out” and generate felr and felr notes as time goes.

Potential improvements

- Normalizing the MIDI dataset to control the variability of our seed sequences in terms of tempo, pitch ranges, etc.

- Improve training data management by using generators in order to increase the size of our training datasets (matrix-based approach is rather greedy in terms of memory needs).
- Use another paradigm for data encoding, such as NLP-inspired approaches instead of one-hot encoded piano roll matrices.

Future Work

I was able show that a single-layer LSTM, character-level language model applied to midi-data representations is capable of generating music that is at least comparable to sophisticated time series probability density techniques prevalent in the literature. I shold that our models Ire able to learn meaningful musical structure.

Given the recent enthusiasm in machine learning inspired art, I hope to continue our work by introducing more complex models and data representations that effectively capture the underlying melodic structure. Furthermore, I feel that more work could be done in developing a better evaluation metric of the quality of a piece – only then will I be able to train models that are truly able to compose original music!

I have achieved remarkable results and beautiful melodies by using a simple LSTM network. HoIver, there are areas that can be improved.

First, the implementation I have at the moment does not support varying duration of notes and different offsets betIen notes. To achieve that I could use more than two notes besides C-major and A-minor for each different key.

To achieve satisfying results with more complex melody, I would have to increase the depth of the LSTM network, which would require a significantly more poIrful computer.

Second, add beginnings and endings to pieces. As the network is now there is no distinction betIen pieces, that is to say the network does not know where one piece ends and another one begins.

Third, add a method to handle unknown notes. As it is now the network would enter a fail state if it encounters a note that it does not know. A possible method to solve that issue would be to find the note or chord that is most similar to the unknown note.

Finally, adding more instruments to the dataset. As it is now, the network only supports pieces that only have a single instrument. It would be interesting to see if it could be expanded to support a whole orchestra.

Conclusion

I have shown how to create a LSTM neural network to generate music. While the results may not be perfect, they are pretty impressive nonetheless and shows us that neural networks can create music and could potentially be used to help create more complex musical pieces.

Music generation by machines is indeed possible. Is it better or could it be better than music generated by humans? Only time will tell. From these results though I would say that it’s definitely possible.

References

The code for this project can be found here:
https://github.com/Mace-don/MusicGenerationLSTM