Music Generation Using Deep Learning

Ashika Meryl Pinto PES1201701346, Rachana Sudhindra Dani PES1201700950

***Abstract*—Composing music can be really tough if you do not have the hang of instruments. Everyone likes to listen to good music and if there is some way to generate music automatically, particularly some decent quality tunes, then it could be of great help to the music industry. This task can be achieved by using neural networks.**

I. INTRODUCTION

Our task is to take a set of already existing music ,then train our model using this data. By doing this, the model learns the patterns in music that humans enjoy. Once it does this , it should be able to generate music which is not present in the training data and is of decent quality.

Our training data has around two hundred songs in the midi format(a format to store music that directly encodes musical notes). The Midi format of music was chosen because of the availability of the extensive python libraries like python-midi to handle such files.

The project includes four main phases: processing the music files and bringing them to a form that can be understood by our model, training our model on this processed training data, using the trained model to generate new music in the encoded format, converting the encoded music back to midi format so that we can actually listen to it.

A restricted Boltzmann machine was used to train on the encoded music files. Unlike most of the neural networks, the restricted Boltzmann machine is a generative model that directly models the probability distribution of the data.To sample from a restricted Boltzmann machine , Gibb's sampling is used.

II. RESTRICTED BOLTZMANN MACHINE

.

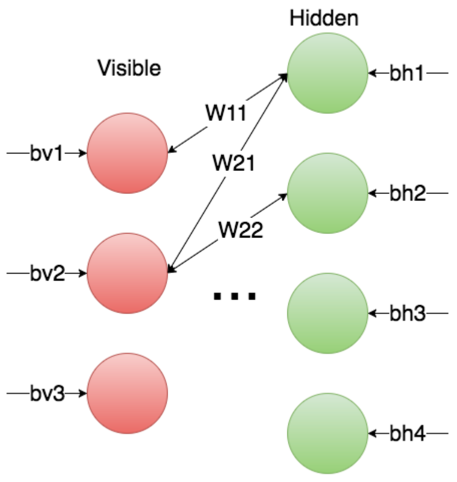
A. Introduction

Before understanding what a restricted Boltzmann machine does, we need to know what generative models are. Generative models specify a probability distribution over a dataset of iput vectors.

The restricted Boltzmann machine is one such generative model. It is a neural network that has two layers namely the visible layer and the hidden layer. These two layers are densely connected, that is, each neuron in the visible layer is connected to each neuron in the hidden layer(or vice versa), but no two neurons of the same layer are connected.

As the network has two layers, the primary parameters would be:

1. W, the weight matrix, which has size Nvisible * Nhidden
2. Bh is the bias for the hidden layer, it is a vector with Nhidden number of elements.
3. Bv is the bias for the visible layer, its is a vector with Nvisible number of elements.



B. RBM Sampling

Restricted Boltzmann machines are mainly used to learn the probability distribution of data. That's primarily why we have chosen it for music generation. If the model learns the distribution of the training data files, we can keep sampling from the learnt distribution to come up with new music files.To sample from an RBM, we use something known as Gibb's sampling which is done in the following way:

1. Initialize the first layer(visible layer) of the network with an input from the training data set.
2. Repeat these steps K number of times-
   a.Send the values in the visible nodes forward to the hidden layer, then sample the new values of the hidden nodes, that is, set the values of each hi to be 1 with probability σ(WTv+bh)i
   b. Send the values in the hidden nodes backward to the visible layer, then sample the new values of the visible nodes, that is, set the values of each vi to be 1 with probability σ(Wh+bv)i

At the end of the algorithm the visible nodes will store the value of the sample

C. *Training the RBM*

Training the RBM is nothing but finding the values of W, bv and bh such that the likelihood of data being drawn from the probability distribution that represents the training data.The procedure for traing would hence be:

1.First initialize the first layer(visible layer) of the network with an input from the training data set. Let's call this input as X

2. Next, sample X' using Gibb's sampling from the probability distribution.

3. Find difference between X and X' and update the weight matrix and the biases in a direction that minimizes this difference. i.e find the sample of the hidden nodes , starting from the visible state of x and the sample of the hidden nodes, starting from the visible sate of X'. Next we update the values of W, Bh and Bv based on the difference between the samples we drew and the original values.

The equations for updation would be:

$W = W + lr\ (X^T h(x) - X'^{\,T} h(X'))$

$Bh = Bh + lr\ (h(X) - h(X'))$

$Bv = Bv + lr\ (X - X')$

Where X is the original vector and X' is the gibb's sample drawn . lr is the learning rate. H is the function that takes in the values of the first layer(visible layer) and returns a sample of the hidden nodes. i.e

h = sample(sigmoid(X*W + Bh)). This way of using Gibb's sampling to train the RBM is called Contrastive Divergence.

III. PROCESSING MIDI DATA

Our training data has around two hundred midi files. Midi files encode musical notes. Midi files encode events that a synthesizer would need to know about, such as Note-on, Note-off, Tempo changes, etc.These midi files need to be converted into a format that the neural network can understand. Out of all the events that the midi file encodes, we will only be focusing on which note is pressed at what time instance and which note is released at what time instance. So every song in the traing data is converted into a binary matrix. Each row in this matrix represents a particular time instance. The number of columns in this matrix is equal to two times the note range (i.e the total number of distinct notes in the training data/of the instrument.

So, in the encoded matrix, M, if M[i,j]=1 it means that at time instance i , note j was pressed and if M[i, n+j]=1 where n is the note range, it means that at time i , note j was released.

So each song in the training data is converted to a matrix.

The following python libraries are seen useful in processing midi files:

- python-midi contains fundamental tools for reading and writing midi files in python
- music21 is a more advanced (and complicated) midi library

IV. IMPLEMENTATION

The project includes the following main phases:

1.Processing the music files and bringing them to a form that can be understood by our model: The 200 music files are each converted into binary matrices each of size timestamp_in_song * (2* note range). Then we reshape each of the songs so that each training example is of a fixed size.

2. Training our model on this processed training data: Then , for each training example, we split the matrix into batches of batch_size and train the RBM with one batch at a time. Updations to all parameters are done according to formulas specified in section II

3.Using the trained model to generate new music in the encoded format: Once the model is trained , and W,Bh and Bv are learnt according to the distribution, we will initialize the visible nodes to zero and then run a gibb's chain to piuck samples. The samples drawn are the encoding for the new music files. These samples will be reshaped again to the same fixed size we converted each song into. Basically, Generation of the music from the trained model includes the following steps:

- Randomly generate vectors of size 1X1560.
- Generate X' (sample) corresponding to that vector.
- Convert that vector to midi format.

4.Converting the encoded music of fixed size back to midi format so that we can actually listen to it.

5.Merging the midi files into one output midi file.

V. RESULTS

We are successfully able to generate music of reasonable quality. The aim of our project is to generate new music on the press of a key. As long as the music is melodious we can say that the output is good.

VI. FURTHER WORK

Since each additional timestep increases the dimensionality of the vectors we are inputting to the model, we are limited to sequences of only a few seconds.Hence, we aim to improve the model to produce longer sequences of music.
Also,as of now the data available is only for piano. We would like to extend our work for music files consisting of notes corresponding to multiple musical instruments.

REFERENCES

[1] https://github.com/hexahedria/biaxial-rnn-music-composition for midi panipulation.
[2] https://www.youtube.com/watch?v=NSQ7NwSe7Zc for Contrastive Divergence (math)
[3] https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976
[4] https://adventuresinmachinelearning.com/python-tensorflow-tutorial/