SMAI PROJECT REPORT
MUSIC GENERATOR

Team and work distribution
Abhay Patil (2020101022) - WaveNet approach
Vaibhav Singh Tomar (2020101058) - WaveNet approach
Rishab Jain (2020111003) - LSTM approach
Mayank Shukla (2020101029) - LSTM approach

Brief Introduction of the topic
In the process of "automatic music generation," a computer programme creates brief musical compositions based on variables such as pitch interval, notes, chords, tempo, etc. The following terminology will be used with the piano instrument in this project:

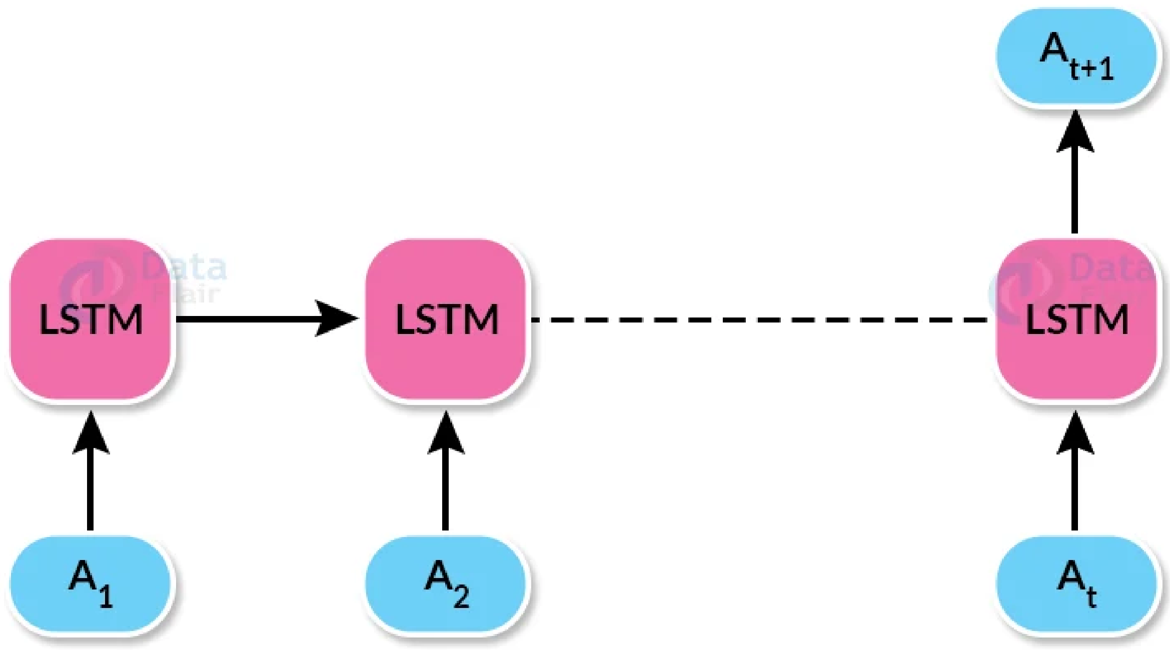
Note that only one key was used to create this sound.
Chords: A chord is a grouping of two or more notes.
Octave: In a piano, the space between two notes is expressed as an octave. Particularly, it is the space between the two notes that begin with the same letter.

Different approaches to automatic music generation
In this project, we explore two deep-learning architectures in detail for automatic music generation - LSTM and WaveNet. But first, why did we choose only deep learning architectures? Thinking about the problem, we can see that it is very similar to NLP (Natural Language Processing), and there are already well established deep-learning solutions to many core NLP problems. Deep learning architectures are capable of automatically extracting subtle features from the dataset, and are capable of learning complex nonlinear functions.

Why use MIDI files?
MIDI (Musical Instrument Digital Interface) are compact in size, and are organized into a collection of notes and chords, with a specific start time, offset etc. and other salient features that allow for easy manipulation of data.

LSTM
In this project, we'll use LSTM to build a model for automatic music generation. All music files will be mined for notes, which will be fed into the model for forecasting. Finally, using these anticipated notes, we will produce a MIDI file.

A form of RNN (Recurrent Neural Network), Long Short Term Memory (LSTM), resolves some situations where RNN failed. RNN networks keep data from past output in a memory for a very little amount of time, while LSTM networks solve the long-term reliance problem of RNNs.



The Vanishing Gradient and the Exploding Gradient problem is also resolved by LSTM, where the model attempts to minimize loss after each time step by computing loss with regard to specified weights. This weight gradually decreases until it eventually disappears or approaches zero at which point the model stops exercising.

For this project we used the dataset of pop songs MIDI files.
Link to the database:
https://drive.google.com/drive/folders/1cI-17cfBBW3689je-7VpeLMo5k9Hf9uY?usp=sharing
Link to the project Files:
https://github.com/rishabhj2912/Generate_music_ML

We only worked on files containing successive streams of Piano data for this research. We used only Piano and split all files according to their instruments. To train the model, we only used nodes and chords, transforming them into an array.

We found out the unique notes with their frequency and set the threshold to 20.
We made two dictionaries, one with notes index as the key and notes as the value, and the other with notes as the key and its corresponding index as the value. Later, we're going to go observe how it's used.Input and output sequences for our model were then built. A timestep of 50 was employed. So, if we went through our input sequence's first 50 notes, the output note for that sequence would be the 51st note.
For our model, we rearranged our array and divided the data into an 80:20 ratio. 80 percent for the training set and 20 percent for the test set.
We used 2 stacked LSTM layers with a dropout rate of 0.2. Dropout basically prevents overfitting while training the model, while it does not affect the inference model. Finally we used a fully connected Dense layer for output.
Output dimension of the Dense Layer will be equal to the length of our unique notes along with the 'softmax' activation function which is used for multi-class classification problems.
After building the model, we will now train it on the input and output data. For this will be using 'Adam' optimizer on batch size of 72 and for total 50 epochs.
Firstly generate a random integer(index) for our testing input array which will be our testing input pattern. We will reshape our array and predict the output. Using the 'np.argmax()' function, we will get the data of the maximum probability value. Convert this predicted index to notes using 'ind2note'(index to note) dictionary. Our next music pattern will be one step ahead of the previous pattern. Repeat this process till we generate 400 notes. Again you can change this parameter as per your requirements.
Finally, we are ready with the predicted output notes. Now we will save them into a MIDI file.

WaveNet

WaveNet is a deep-learning based generative model developed by Google DeepMind. The main goal of the WaveNet is to generate new samples from the original distribution of data, hence the name generative model. We will use this model as inspiration, generating a CNN based model of our own.

Input and output

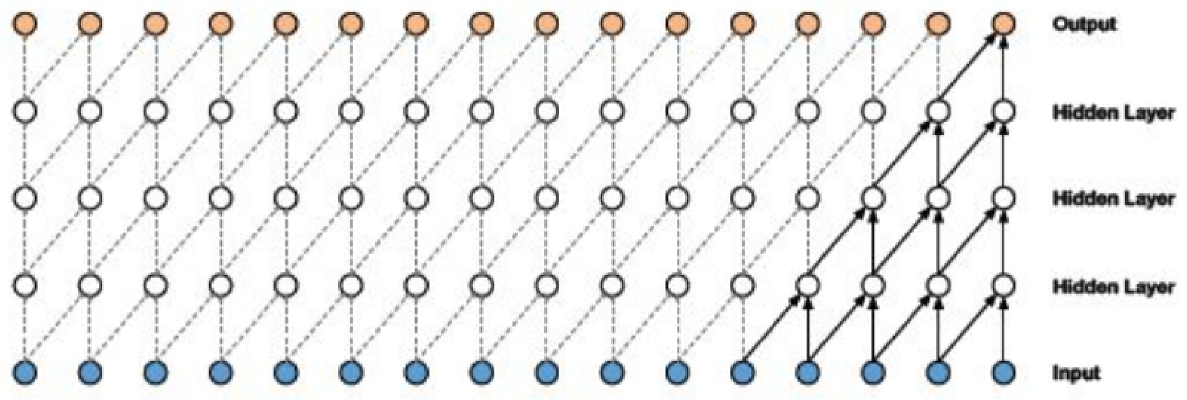
WaveNet takes a chunk of raw audio as input (representation of a wave in the time-series domain). The audio wave takes the form of amplitude values recorded over different intervals of time. From a given sequence of amplitude values, WaveNet tries to predict the successive amplitude value.

Working

WaveNet is composed of building blocks that are dilated 1 dimensional convolution layers, which captures the sequential information present in the input sequence. The training phase is much faster when compared to GRU and LSTM owing to the absence of recurrent connections. There are two types of convolutions:

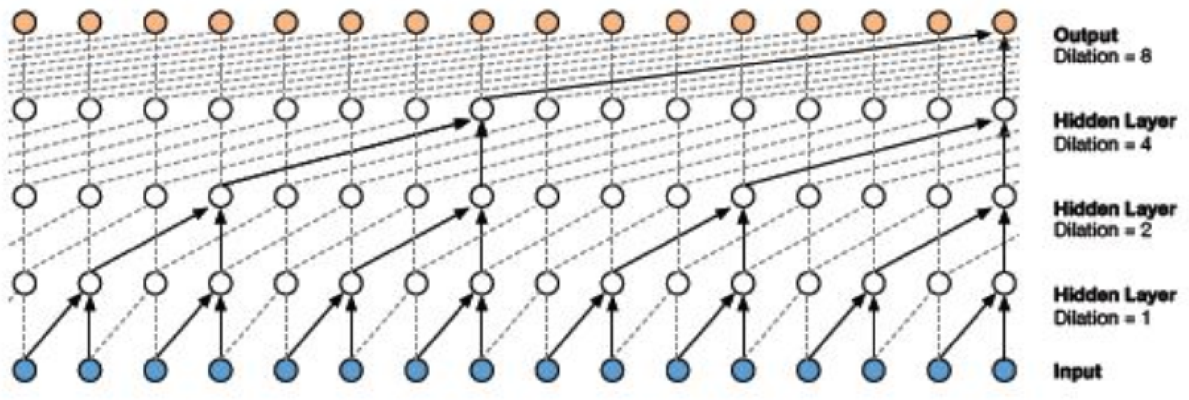
Casual convolution:

The receptive field (the number of inputs that the output is influenced by) is very low here. As shown in the diagram below, the output is only dependent on 5 input values.



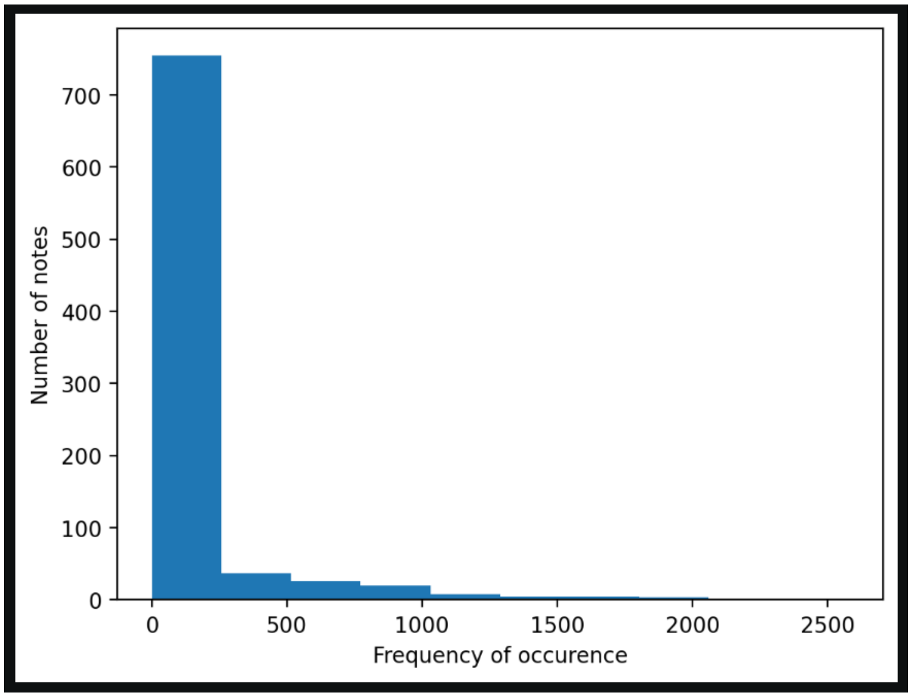
Dilated Causal convolution:

The receptive field is increased by exponentially increasing the dilation rate at every hidden layer.



Steps

1. We extract the notes and chords from the collection of all midi files into a single list / array. It is recommended to use a threshold frequency to filter out notes that appear less frequently. This simplifies the process while retaining all the important data that is carried by the more frequent notes.



The above figure shows that over 700 notes had a frequency between 0 and 250, while less than 100 notes had a frequency between 250 and 500 (frequency here implies frequency of occurrence in the set of all notes, and not in the musical sense). A threshold of 100 would mean that notes that we only choose the notes that appeared more than 250 times.

2. Preparing the data - From the list of notes at our disposal, we must create pairs of sequences of input notes and the subsequent output note. Our model will use these sequences of input notes to predict the output note, and compare it with the actual output note.

3. Convert to numbers - We must convert the notes (which can consist of both numbers and alphabets) to numbers that can be interpreted by the model. This is essentially a one-to-one mapping.

4. Now, we build the model, which uses relu and softmax activation functions, adam optimizer etc to build a deep learning network that can generate music.

5. Finally, we allow the model to work on the data that we prepared in steps 2 and 3, convert the numbers back to notes to generate a new midi file, which basically corresponds to a new song!

Dataset

For the dataset, we personally downloaded more than 300 midi files from the following website

Observation

The generated music is saved in the folder ./WaveNet/saved_music/epochs50.
The default batch size is 256 and the default number of epochs is 50. We have experimented with different parameters and saved the music in their respective folders such as './batch16', './batch4096', 'epochs01' and 'epochs10'.
A notable reduction in performance was observed when the batch size was reduced to 16, as well as increased to 4096. The deterioration of quality was seen in the form of repetition of the same note for an unnaturally long period of time.

Future work

We will continue to experiment with the parameters (batch size and number of epochs) until we get a sizable set of music for each category. Further, we will conduct a survey to see how people respond to the automatically generated music, and inquire about things such as the Turing test (whether the generated music can pass as human-made) and quality of the composition etc.