Music Generation Using Deep Learning

Topics of Deep Learning

Team

Kokila K N	PES2201800625
Rajesh Manchikanti	PES2201800570
Arjun Shekhar	PES2201800072

Outline

1. Problem Statement
2. Literature Review
3. Flow Diagram
4. Requirements
5. Frameworks
6. Deliverables

Problem Statement

MUSIC GENERATION USING DEEP LEARNING

Everyone like to listen interesting music and if there is some way to generate music automatically, particularly decent quality music then it’s a big leap in the world of music industry.

Music is nothing but a **sequence of musical notes**,so collection of such sequences of different musical instruments can be helpful to generate music with the help of AI as a human do.

Literature Survey

Paper 1: **Music Generation Using Bidirectional Recurrent Network**

- This paper talks about bidirectional recurrent network,an efficient approach which generates note suitable to global perspective.To generate harmonic ,both positive direction data(previous data) and negative direction data should be considered,but traditional methods use only previous data to generate current note.
- According to author of paper, In BRNN activation of current note is generated based on context,previous activations and next activations,an efficient lose function is used to overcome lot of meaningless results.Quality of music generated is far better than any traditional probabilistic approach and markov models.

- BRNN uses LSTM (special case of RNN) because of its capability of storing meaningful information for long short term . Two LSTM are stacked to increase the network capacity

- This paper uses 2 bit piano-roll as a input to BRNN

Literature Survey

Paper 2: **Generating Music Algorithm with Deep Convolutional Generative Adversarial Networks**

This paper proposes an advanced arithmetic for generating music using Generative Adversarial Networks (GAN). The music is divided into tracks and the note segment of tracks is expressed as a piano-roll, through trained a gan model which generator and discriminator continuous zero-sum game to generate a wonderful music integrallty. In addition to the final output layer, we use the sigmoid activation function to distinguish between
real data and generated data.
Also, the discriminator of the GAN model has been optimized according to the nature of the music data.
Because of the algorithm design of the paper, we are required to convert the prepared midi file into a piano roller blind and then transform it into the input data format that we need to train.After establishing the G-network and D-network models, we then trained the network GAN.
We used two sets of programs to evaluate our experimental results, the first set is based on statistically objective evaluations,while the second set is to conduct a sample survey. Experimental data and user evaluations show that the proposed model can generate music like sequence generation.The model is theoretically mature and has ideal properties.

Literature Survey

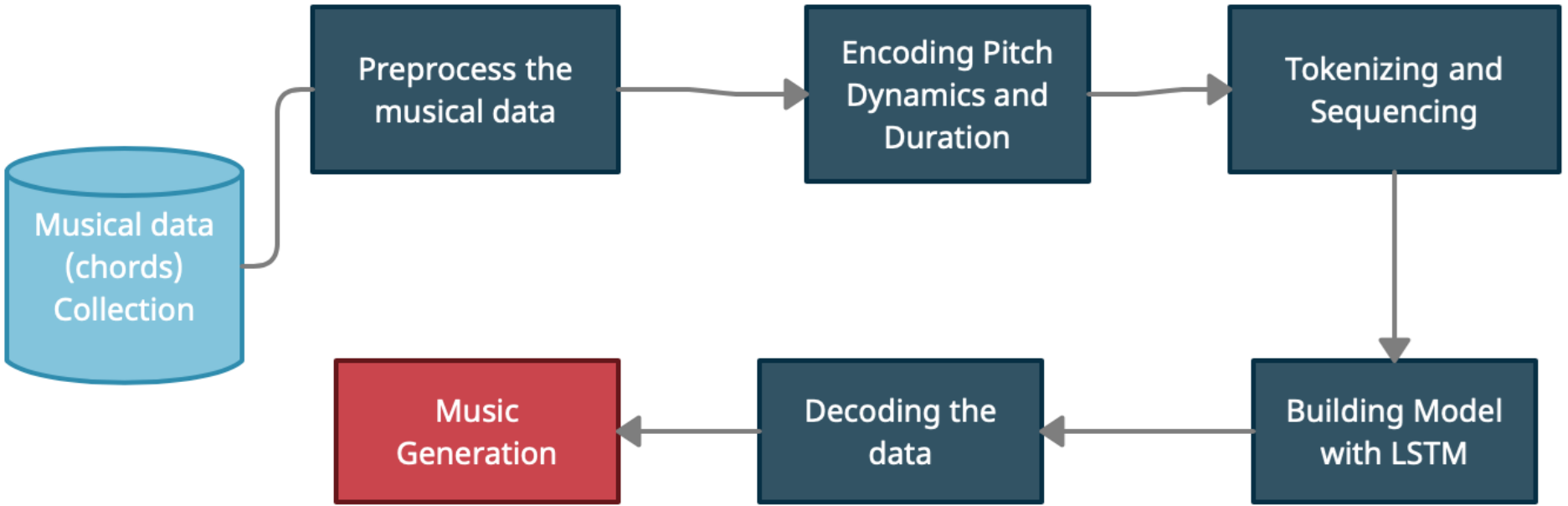
Paper 3 : **STYLE-CONDITIONED MUSIC GENERATION**

This paper talks about generating music automatically based on content and style inputs from user rather than training a single musical data and generating music.

This paper is based on the formulation of VAE (Variational Autoencoder) to generate music , where they have collected dataset from MIDI files (Bach Chorales (JSB) dataset and Nottingham Music Database (NMD)), this music data is converted to non overlapping segments and the data has been encoded with bidirectional hyper LSTM layer an decoded with unidirectional HYperLSTMand the model is built around 500 music sample of each data sample is generated, the number of chords and the unique pictures in the training data is compared against the generated samples and along with music generation style transfer is also done.

This paper presents about the vanilla sequence-to-sequence variational autoencoder that allows user to condition the style of generated output music. This model involves a continuous style embedding for each style in the dataset.

Flow Diagram



Software and Hardware Requirements

Software requirements

1. Operating System
2. Python
3. Jupyter Notebook

Hardware requirements

1. Processor - 1.9 - 3.3 gigahertz (GHz) or faster 64-bit dual core processor with SSE2 instruction set
2. Memory - 4GB RAM or more

Framework and Tools

For model

- Keras
- Tensorflow
- Music21
- Magenta by tensorflow
- Pandas
- Numpy
- Matplotlib
- Scipy

For Website

- HTML
- CSS
- JS
- Flask (for API)

For Hosting

- AWS/Heroku

Final Deliverables

1. SRS Document.
2. Collection of data for model building.
3. Deep Learning model for music generation.
4. Functional website which generates a completely new music every time a user access.
5. IEEE format report.

Music Generation

Data sources:

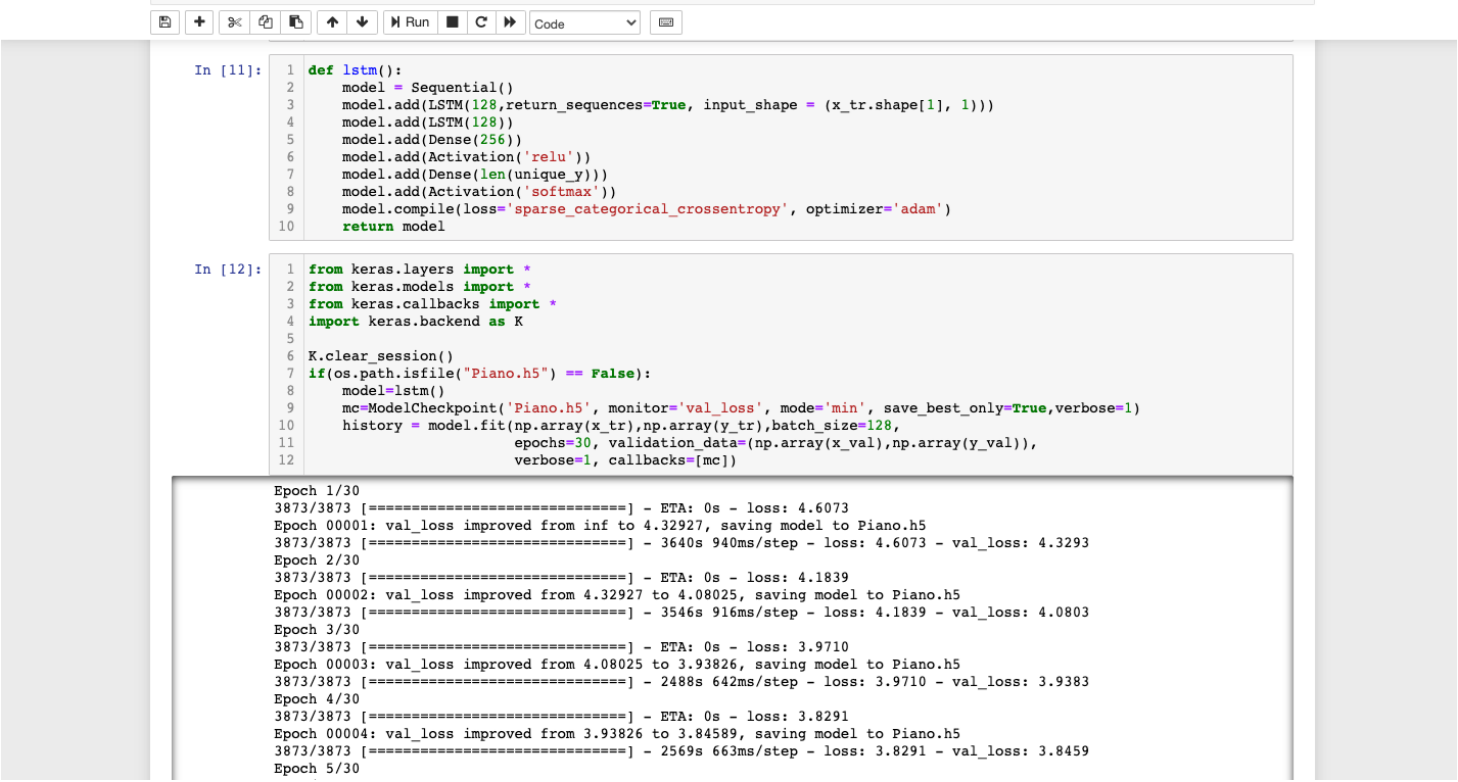
- https://drive.google.com/file/d/0B4wY8oEgAUnjX3NzSUJCNVZHbmc/view

Model Used?

- Long short Term Memory is used for generation model to generate music of various instruments
- One lstm model is used to for each instruments music present

- User can play music of each instrument separately or mix of random 3 instruments or mix of all instruments
- Recorded songs are also available for the user

Screenshots



In [11]:

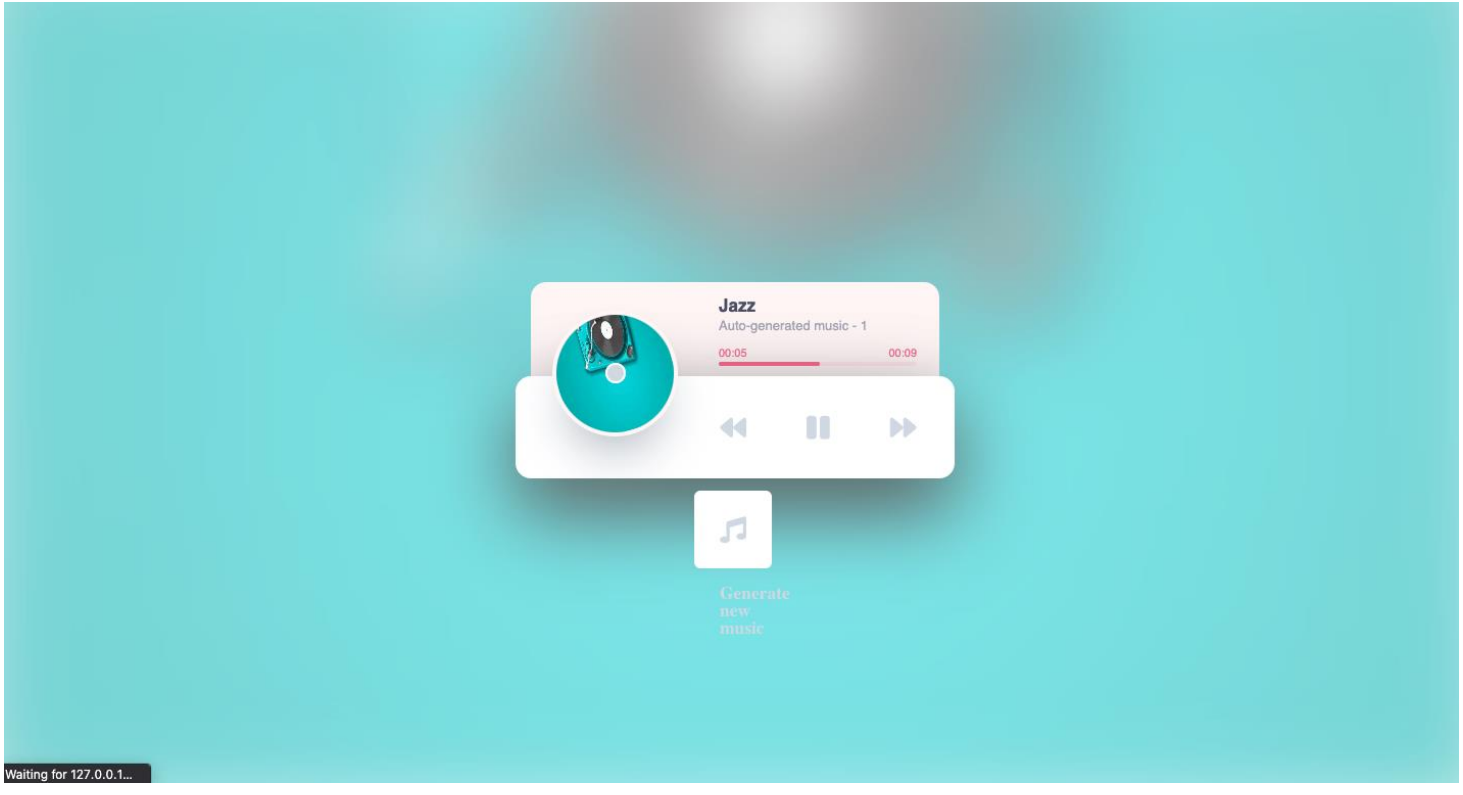
```
def lstm():
    model = Sequential()
    model.add(LSTM(128,return_sequences=True, input_shape = (x_tr.shape[1], 1)))
    model.add(LSTM(128))
    model.add(Dense(256))
    model.add(Activation('relu'))
    model.add(Dense(len(unique_y)))
    model.add(Activation('softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')
    return model
```

In [12]:

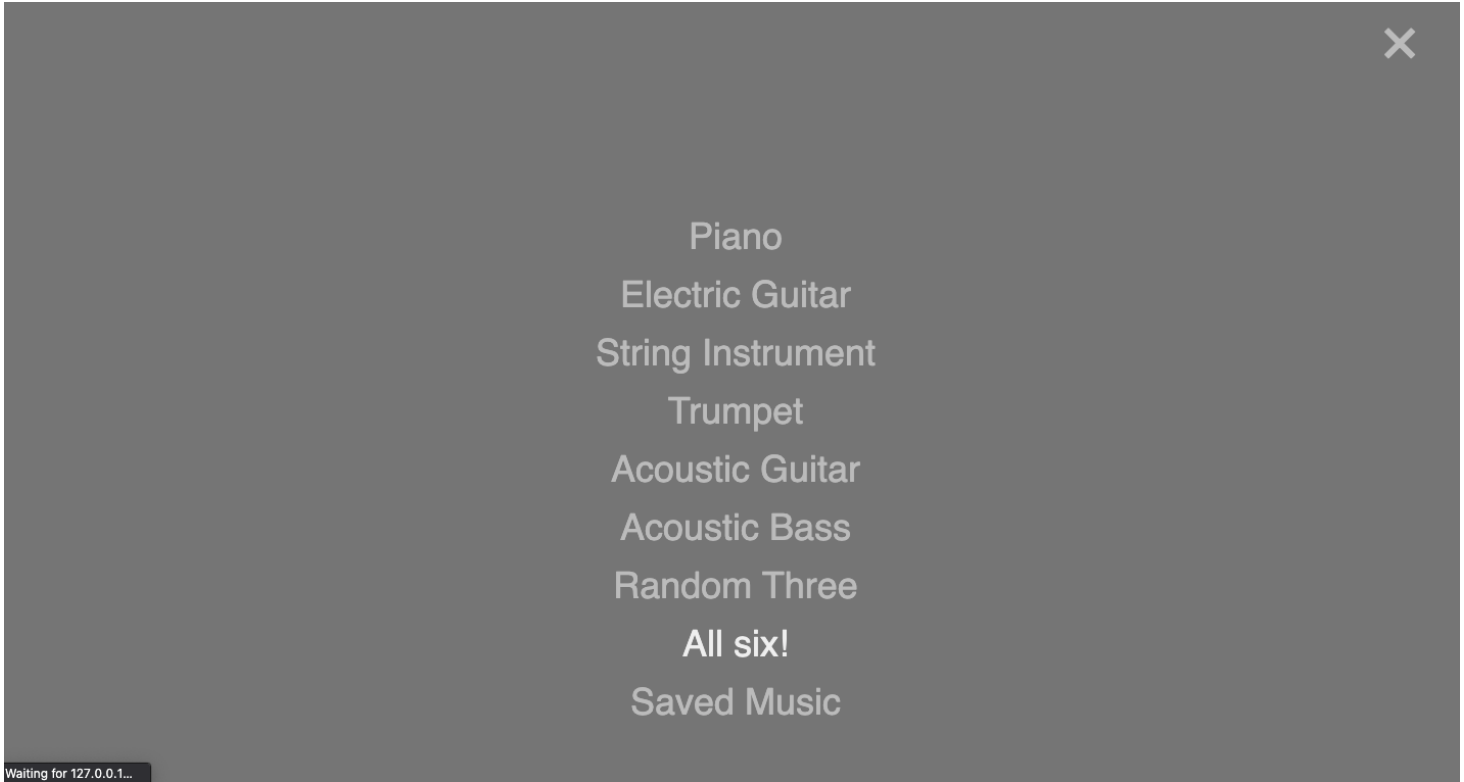
```
from keras.layers import *
from keras.models import *
from keras.callbacks import *
import keras.backend as K

K.clear_session()
if(os.path.isfile("Piano.h5") == False):
    model=lstm()
    mc=ModelCheckpoint('Piano.h5', monitor='val_loss', mode='min', save_best_only=True,verbose=1)
    history = model.fit(np.array(x_tr),np.array(y_tr),batch_size=128,
                        epochs=30, validation_data=(np.array(x_val),np.array(y_val)),
                        verbose=1, callbacks=[mc])
```

```
Epoch 1/30
3873/3873 [==============================] - ETA: 0s - loss: 4.6073
Epoch 00001: val_loss improved from inf to 4.32927, saving model to Piano.h5
3873/3873 [==============================] - 3640s 940ms/step - loss: 4.6073 - val_loss: 4.3293
Epoch 2/30
3873/3873 [==============================] - ETA: 0s - loss: 4.1839
Epoch 00002: val_loss improved from 4.32927 to 4.08025, saving model to Piano.h5
3873/3873 [==============================] - 3546s 916ms/step - loss: 4.1839 - val_loss: 4.0803
Epoch 3/30
3873/3873 [==============================] - ETA: 0s - loss: 3.9710
Epoch 00003: val_loss improved from 4.08025 to 3.93826, saving model to Piano.h5
3873/3873 [==============================] - 2488s 642ms/step - loss: 3.9710 - val_loss: 3.9383
Epoch 4/30
3873/3873 [==============================] - ETA: 0s - loss: 3.8291
Epoch 00004: val_loss improved from 3.93826 to 3.84589, saving model to Piano.h5
3873/3873 [==============================] - 2569s 663ms/step - loss: 3.8291 - val_loss: 3.8459
Epoch 5/30
```

Screenshots



Screenshots



Thank You