MUSIC GENERATION WITH AI

Tarun Sharma[1], Vineet Pandey[2], Manoj Singhal[3]

[1]B. Tech Student, Greater Noida Institute of Technology, Greater Noida, India
[2]B. Tech Student, Greater Noida Institute of Technology, Greater Noida, India
[3]Professor, Dept. of Information Technology, Greater Noida Institute of Technology, Greater Noida, India

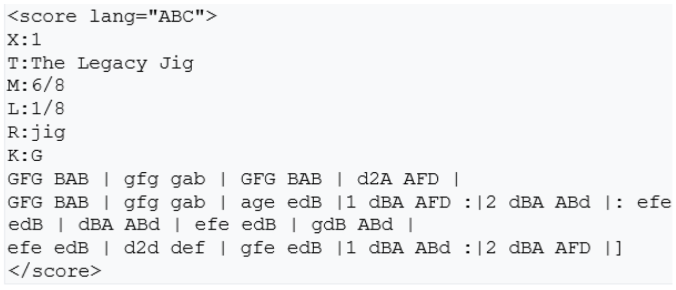
---***---

Abstract - *One of the exciting applications of recent advances in the field of AI is Artificial Music generation. Is it possible to reproduce artists' creativity through AI? Can a Deep Learning model be an inspiration or a productivity tool for musicians? To find the answers we tried to do a project on Music generation through AI. The overall idea of the project was to take some existing music data and then train a neural network model using this data. So, the model has to learn the patterns in the music that we human enjoy and once it learns this, the model should be able to generate new music for us. One important condition was that it cannot simply copy paste from the training data.*

Key Words: LSTM, RNN, abc notations,

1.INTRODUCTION

Like every other AI project, we also started with data. We had to think what sort of data should we use or simply which music representation of data that we can use. There were variety of options for us to choose from like MIDI notation, ABC notation, Sheet music etc., we decided to go with ABC notation. Just to visualize, ABC notation looks like (source Wikipedia).

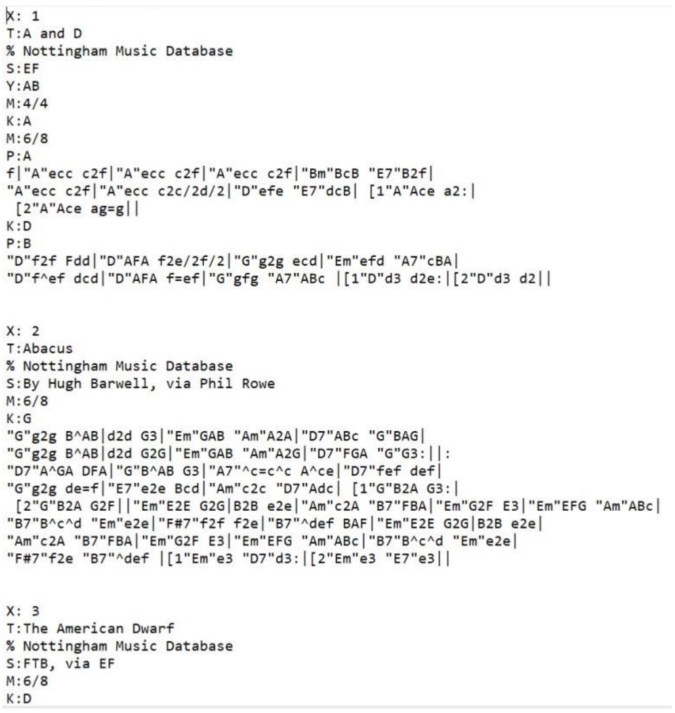


```
<score lang="ABC">
X:1
T:The Legacy Jig
M:6/8
L:1/8
R:jig
K:G
GFG BAB | gfg gab | GFG BAB | d2A AFD |
GFG BAB | gfg gab | age edB |1 dBA ABd :|2 dBA ABd |: efe
edB | dBA ABd | efe edB | gdB ABd |
efe edB | d2d def | gfe edB |1 dBA ABd :|2 dBA AFD |]
</score>
```

So, we collected musical data of around 400 melodies all represented in ABC notation. For our model we used LSTMs. LSTMs are the special case of RNN.

1.1 Data

We pasted all the melodies into a single .txt file separated by a line. Our data looked like (original screenshot).



```
X: 1
T:A and D
% Nottingham Music Database
S:EF
Y:AB
M:4/4
K:A
M:6/8
P:A
f|"A"ecc c2f|"A"ecc c2f|"A"ecc c2f|"Bm"BcB "E7"B2f|
"A"ecc c2f|"A"ecc c2c/2d/2|"D"efe "E7"dcB| [1"A"Ace a2:|
 [2"A"Ace ag=g||
K:D
P:B
"D"f2f Fdd|"D"AFA f2e/2f/2|"G"g2g ecd|"Em"efd "A7"cBA|
"D"f^ef dcd|"D"AFA f=ef|"G"gfg "A7"ABc |[1"D"d3 d2e:|[2"D"d3 d2||


X: 2
T:Abacus
% Nottingham Music Database
S:By Hugh Barwell, via Phil Rowe
M:6/8
K:G
"G"g2g B^AB|d2d G3|"Em"GAB "Am"A2A|"D7"ABc "G"BAG|
"G"g2g B^AB|d2d G2G|"Em"GAB "Am"A2G|"D7"FGA "G"G3:||:
"D7"A^GA DFA|"G"B^AB G3|"A7"^c=c^c A^ce|"D7"fef def|
"G"g2g de=f|"E7"e2e Bcd|"Am"c2c "D7"Adc| [1"G"B2A G3:|
 [2"G"B2A G2F||"Em"E2E G2G|B2B e2e|"Am"c2A "B7"FBA|"Em"G2F E3|"Em"EFG "Am"ABc|
"B7"B^c^d "Em"e2e|"F#7"f2f f2e|"B7"^def BAF|"Em"E2E G2G|B2B e2e|
"Am"c2A "B7"FBA|"Em"G2F E3|"Em"EFG "Am"ABc|"B7"B^c^d "Em"e2e|
"F#7"f2e "B7"^def |[1"Em"e3 "D7"d3:|[2"Em"e3 "E7"e3||


X: 3
T:The American Dwarf
% Nottingham Music Database
S:FTB, via EF
M:6/8
K:D
```

Then we created a vocabulary of all the different characters that we had in the data. Instead of directly feeding all our data at once in our model, we used batch processing system. We created batches of batch size 16 and sequence length of 64.

1.2 Model

We use Keras framework to apply deep neural networks. We started with sequential model. The first layer that we added was LSTM layer (with return sequence = True and stateful = True). Return sequence = True, allows us to apply many to many RNN. Stateful = True, helps in batch processing. So, what is does is that it saves the output generated at the last time stamp after the current batch so as to pass it to the LSTMs at first time stamp of the next batch. We used Dropout regularization with probability 0.2. Then we added time distributed dense layer. Time Distributed dense layer is basically having a dense layer at every time stamp. So, the output weights from the LSTMs are passed to a dense layer at every time stamp. Time distributed dense layer not normal dense layer is required here as we are trying to apply many to many RNN.
At the end we used softmax classifier.

1.3 RNN

In traditional machine learning models, we cannot store a model's previous stages. However, we can store previous stages with Recurrent Neural Networks (commonly called RNN). An RNN has a repeating module that takes input from the previous stage and gives its output as input to the next stage. However, RNNs can only retain information from the most recent stage, so our network needs more memory to learn long-term dependencies. This is where Long Short Term Memory Networks (LSTMs) come to the rescue.

LSTMs are a special case of RNNs, with the same chain-like structure as RNNs, but a different repeating module structure.

2. Music Generation

So how our model works is that it takes a character index and outputs a sequence of probabilities for all the unique characters in our vocabulary. That's what softmax classifier gives. Then the model samples the output sequence of probabilities and choses the next character. Then the next character generated is fed to the model and again same process is repeated and next character is generated.
The input size used in the trained model is the batch size. And for the generation of music via machine learning, the input size is a single character. So, we create a new model which is similar to the trained model, but with the input size of a single character which is (1,1). To this new model, we load the weights from the trained model to replicate the characteristics of the trained model.

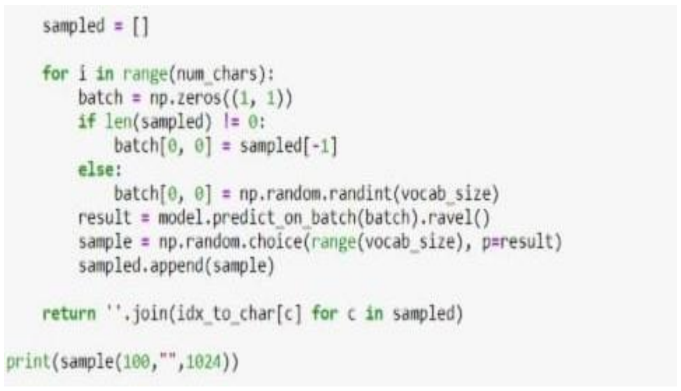
```
model2 = Sequential()
model2.add(Embedding(vocab_size, 512,
batch_input_shape=(1,1)))
for i in range(3):
   model2.add(LSTM(256,        return_sequences=True,
stateful=True))

   model2.add(Dropout(0.2))
model2.add(TimeDistributed(Dense(vocab_size)))

model2.add(Activation('softmax'))
```

In the process of music generation, the first character is chosen randomly from the unique set of characters, the next character is generated using the previously generated character and so on. With this structure, we generate music.

The code snippet that helped us achieve us this:



```
    sampled = []

    for i in range(num_chars):
        batch = np.zeros((1, 1))
        if len(sampled) != 0:
            batch[0, 0] = sampled[-1]
        else:
            batch[0, 0] = np.random.randint(vocab_size)
        result = model.predict_on_batch(batch).ravel()
        sample = np.random.choice(range(vocab_size), p=result)
        sampled.append(sample)

    return ''.join(idx_to_char[c] for c in sampled)

print(sample(100,"",1024))
```

3. CONCLUSIONS

Our model successfully generated decent quality of music. Some of the music samples generated by our model can be seen below (original music generated by our model).

X: 111
T:Garly Hill
% Nottingham Music Database
S:Trad, arr Phil Rowe
M:6/8
K:D
"D"A2f f2f|A3 -e2c|"G"B2d BAG|"D"F3 F2A|"D"d2c d2c|d3 -d2c|
"G"B2B Bcd|"C"e3 "Bm"d2d|"C"e2d "G"B3|"Am"A2c "Em"B2G| [1"A"A3 -A2||
[2"A"A3/2z/2e/2f/2|"A"e3 c3|"D"d3/2f/2g fga|"Em"g3 "A7"a3|"D"fde d2:|

X: 189
T:Magcon Ay
% Nottingham Music Database
S:Trad, arr Phil Rowe
M:6/8
K:D
"A"A3 |:"D"DFA "C"=CEC|"D"DED F2A|"D"def "G"gab|"D"agf "A7"ecA|
"D"d2A d2e|"D"fgf fed|"A7"ecA ABA|"D"d3 -d2||

X: 173
T:Leaping Jack
% Nottingham Music Database
S:EF
Y:AB
M:4/4
K:G
M
:
6
/
8

```
P
:
A

d/2c/2|"G"BGB dcB|"D"ABA "Em"dAB|"Am"cBA
"D7"AGF|"G"GAG G2:|
P:B
|:A|"G"d2d ded|"C"e2d efg|"D"f2A fed|"G"g2d "D7"edc|
"G"B2A GBd|"G"dBg "Em"deg|"Em"agf
"A7"edc|"D7"d2d "G7"de=f|
"C"ece "G"dBg|"C"e/2a3/2e "A7"gec|"D"d3 -d3:|
```

As we can see our model was able to generate the entire format of ABC notation. There are so many online editors available on the internet that can play ABC notations. When we played these music samples, definitely we can say our AI did fantastic job. And one important point that I want to highlight here is that we trained our model for 100 epochs. So, when we tried to generate music from diff models which we trained for 10 epochs, 20 epochs, 30 epochs and so on up to 100 epochs, we saw improvement in the quality of the music generated by our model.

While working in this model, we looked at how to process music for use with neural networks, the in-depth workings of deep learning models like RNN & LSTMs, and we also explored how tweaking a model can result in music generation. We can apply these concepts to any other system where we generate other formats of art, including generating landscape paintings or human portraits.

REFERENCES

[1] http://abc.sourceforge.net/NMD/
[2] http://trillian.mit.edu/~jc/music/book/oneills/1850/X/
[3] http://abc.sourceforge.net/NMD/
[4] https://en.wikipedia.org/wiki/ABC_notation
[5] https://abcjs.net/abcjs-editor.html