In [2]:

```
Music creation by LSTM Network
In [59]:

#For understanding MIDI files MIDI
import music21
from music21 import converter, pitch, interval
import os
import glob

Dataset
In [60]:

#Dataset - Midi files - Music Instrument Digital interface(MIDI)
song_dir = "Schubert_dataset"
song_list = os.listdir(song_dir)
song_list

Out[60]:

['schubert_D850_1.mid',
 'schubert_D850_2.mid',
 'schubert_D850_3.mid',
 'schubert_D850_4.mid',
 'schubert_D935_1.mid',
 'schubert_D935_2.mid',
 'schubert_D935_3.mid',
 'schubert_D935_4.mid',
 'schub_d760_1.mid',
 'schub_d760_2.mid',
 'schub_d760_3.mid',
 'schub_d760_4.mid',
 'schub_d960_1.mid',
 'schub_d960_2.mid',
 'schub_d960_3.mid',
 'schub_d960_4.mid',
 'schuim-1.mid',
 'schuim-2.mid',
 'schuim-3.mid',
 'schuim-4.mid',
 'schumm-1.mid',
 'schumm-2.mid',
 'schumm-3.mid',
 'schumm-4.mid',
 'schumm-5.mid',
 'schumm-6.mid',
 'schu_143_1.mid',
 'schu_143_2.mid',
 'schu_143_3.mid']

Separation of Note and Chords
In [61]:

#getting in musical format
from music21 import note, chord, instrument
import numpy as np

Notes_Chords = []
for song in glob.glob(song_dir + '/*.mid'):
    global score
    score = converter.parse(song)
    notes_to_parse = None
    parts = instrument.partitionByInstrument(score)
    if parts:
        notes_to_parse = parts.parts[0].recurse()
    else:
        notes_to_parse = score.flat.notes
    for element in notes_to_parse:
        if isinstance(element,note.Note):
            Notes_Chords.append(str(element.pitch))
        if isinstance(element,chord.Chord):
            Notes_Chords.append(".".join(str(n) for n in element.normalOrder))
total_notes = len((Notes_Chords))
print("Total Length of the Notes List :",len(Notes_Chords))
print("Different Notes :",np.array(Notes_Chords))
```

```
Total Length of the Notes List : 83104
Different Notes : ['2.6.9' '2' '9.2' ... '9.0.4' '9.0.4' '9.0.4']
```

Instruments used in Dataset

In [62]:

```
#to get instrumets used in the midi file
from music21 import instrument
for song in glob.glob(song_dir + '/*.mid'):
    parts = instrument.partitionByInstrument(score) #or use (parts = score.parts.stream(
))
    for p in parts:
        global music_instrument
        music_instrument = p.partName
        print("Instruments: ",music_instrument)
```

```
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
Instruments:  Piano
```

In [63]:

```
#Finding the unique notes or chords,then sorted mapping with a number
Unique_Notes_Chords = sorted(set(item for item in Notes_Chords))
print(Unique_Notes_Chords)
total_unique_notes = len(Unique_Notes_Chords)
print("Total Unique Notes:",total_unique_notes)

#Mapping every unique note with an integer
Notes_Chords_to_int = dict((note , number) for number, note in enumerate(Unique_Notes_Ch
ords))

#Notes_Chords_to_int
np.array(Notes_Chords_to_int)
```

```
['0', '0.1', '0.1.5', '0.2', '0.2.4', '0.2.5', '0.2.6', '0.2.7', '0.3', '0.3.4', '0.3.5'
, '0.3.6', '0.3.6.8', '0.3.6.9', '0.3.7', '0.4', '0.4.5', '0.4.6', '0.4.7', '0.4.8', '0.
```

```
5', '0.6', '1', '1.2.6', '1.3', '1.3.5.8', '1.3.6', '1.3.6.9', '1.3.7', '1.3.7.8', '1.3.
8', '1.4', '1.4.5', '1.4.6', '1.4.7', '1.4.7.10', '1.4.7.9', '1.4.8', '1.5', '1.5.8', '1
.5.9', '1.6', '1.7', '10', '10.0', '10.0.2', '10.0.2.5', '10.0.3', '10.0.4', '10.0.4.5',
'10.0.5', '10.1', '10.1.3', '10.1.4', '10.1.4.5', '10.1.4.6', '10.1.5', '10.2', '10.2.4',
'10.2.5', '10.3', '10.3.4', '11', '11.0', '11.0.2', '11.0.4', '11.1', '11.1.4', '11.1.4.7
', '11.1.5', '11.1.6', '11.2', '11.2.4', '11.2.4.7', '11.2.5', '11.2.5.7', '11.2.6', '11.
3', '11.3.6', '11.4', '2', '2.3', '2.4', '2.4.6', '2.4.7', '2.4.8', '2.4.9', '2.5', '2.5
.7', '2.5.8', '2.5.8.10', '2.5.8.11', '2.5.8.9', '2.5.9', '2.6', '2.6.10', '2.6.8', '2.6
.9', '2.7', '2.8', '3', '3.4', '3.5', '3.5.10', '3.5.7', '3.5.8', '3.5.8.11', '3.5.9', '
3.6', '3.6.10', '3.6.8', '3.6.8.11', '3.6.9', '3.6.9.11', '3.7', '3.7.10', '3.7.11', '3.
7.8', '3.7.9', '3.8', '3.8.9', '3.9', '4', '4.10', '4.5', '4.5.10', '4.5.8', '4.5.9', '4
.6', '4.6.10', '4.6.10.0', '4.6.11', '4.6.9', '4.6.9.0', '4.7', '4.7.10', '4.7.10.0', '4.
7.11', '4.7.9', '4.8', '4.8.11', '4.9', '5', '5.10', '5.11', '5.6.10', '5.6.9', '5.6.9.0
', '5.7', '5.7.0', '5.7.10', '5.7.11', '5.7.11.1', '5.8', '5.8.0', '5.8.10', '5.8.10.11'
, '5.8.11', '5.8.11.1', '5.9', '5.9.0', '5.9.11', '6', '6.10', '6.10.0', '6.10.1', '6.11
', '6.7', '6.7.11', '6.7.9', '6.8', '6.8.0', '6.8.1', '6.8.11', '6.9', '6.9.0', '6.9.0.2
', '6.9.1', '6.9.11', '7', '7.0', '7.10', '7.10.0', '7.10.0.3', '7.10.1', '7.10.1.3', '7
.10.2', '7.11', '7.11.1', '7.11.2', '7.8', '7.8.0', '7.9', '7.9.0', '7.9.1', '7.9.10', '
7.9.11', '7.9.2', '8', '8.0', '8.0.1', '8.0.2', '8.0.3', '8.1', '8.10', '8.10.1', '8.10.
1.4', '8.10.2', '8.10.3', '8.11', '8.11.0', '8.11.1', '8.11.2', '8.11.2.4', '8.11.3', '8.
9', '9', '9.0', '9.0.2', '9.0.3', '9.0.3.5', '9.0.4', '9.1', '9.1.3', '9.1.4', '9.10', '
9.10.2', '9.11', '9.11.2', '9.11.2.5', '9.11.3', '9.11.4', '9.2', 'A1', 'A2', 'A3', 'A4'
, 'A5', 'A6', 'B-1', 'B-2', 'B-3', 'B-4', 'B-5', 'B-6', 'B1', 'B2', 'B3', 'B4', 'B5', 'B
6', 'C#2', 'C#3', 'C#4', 'C#5', 'C#6', 'C#7', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'D1',
'D2', 'D3', 'D4', 'D5', 'D6', 'D7', 'E-2', 'E-3', 'E-4', 'E-5', 'E-6', 'E-7', 'E1', 'E2'
, 'E3', 'E4', 'E5', 'E6', 'E7', 'F#1', 'F#2', 'F#3', 'F#4', 'F#5', 'F#6', 'F1', 'F2', 'F
3', 'F4', 'F5', 'F6', 'F7', 'G#1', 'G#2', 'G#3', 'G#4', 'G#5', 'G#6', 'G1', 'G2', 'G3',
'G4', 'G5', 'G6']
Total Unique Notes: 308

Out[63]:

array({'0': 0, '0.1': 1, '0.1.5': 2, '0.2': 3, '0.2.4': 4, '0.2.5': 5, '0.2.6': 6, '0.2.
7': 7, '0.3': 8, '0.3.4': 9, '0.3.5': 10, '0.3.6': 11, '0.3.6.8': 12, '0.3.6.9': 13, '0.
3.7': 14, '0.4': 15, '0.4.5': 16, '0.4.6': 17, '0.4.7': 18, '0.4.8': 19, '0.5': 20, '0.6
': 21, '1': 22, '1.2.6': 23, '1.3': 24, '1.3.5.8': 25, '1.3.6': 26, '1.3.6.9': 27, '1.3.
7': 28, '1.3.7.8': 29, '1.3.8': 30, '1.4': 31, '1.4.5': 32, '1.4.6': 33, '1.4.7': 34, '
1.4.7.10': 35, '1.4.7.9': 36, '1.4.8': 37, '1.5': 38, '1.5.8': 39, '1.5.9': 40, '1.6': 4
1, '1.7': 42, '10': 43, '10.0': 44, '10.0.2': 45, '10.0.2.5': 46, '10.0.3': 47, '10.0.4'
: 48, '10.0.4.5': 49, '10.0.5': 50, '10.1': 51, '10.1.3': 52, '10.1.4': 53, '10.1.4.5':
54, '10.1.4.6': 55, '10.1.5': 56, '10.2': 57, '10.2.4': 58, '10.2.5': 59, '10.3': 60, '1
0.3.4': 61, '11': 62, '11.0': 63, '11.0.2': 64, '11.0.4': 65, '11.1': 66, '11.1.4': 67,
'11.1.4.7': 68, '11.1.5': 69, '11.1.6': 70, '11.2': 71, '11.2.4': 72, '11.2.4.7': 73, '1
1.2.5': 74, '11.2.5.7': 75, '11.2.6': 76, '11.3': 77, '11.3.6': 78, '11.4': 79, '2': 80,
'2.3': 81, '2.4': 82, '2.4.6': 83, '2.4.7': 84, '2.4.8': 85, '2.4.9': 86, '2.5': 87, '2.
5.7': 88, '2.5.8': 89, '2.5.8.10': 90, '2.5.8.11': 91, '2.5.8.9': 92, '2.5.9': 93, '2.6'
: 94, '2.6.10': 95, '2.6.8': 96, '2.6.9': 97, '2.7': 98, '2.8': 99, '3': 100, '3.4': 101
, '3.5': 102, '3.5.10': 103, '3.5.7': 104, '3.5.8': 105, '3.5.8.11': 106, '3.5.9': 107,
'3.6': 108, '3.6.10': 109, '3.6.8': 110, '3.6.8.11': 111, '3.6.9': 112, '3.6.9.11': 113,
'3.7': 114, '3.7.10': 115, '3.7.11': 116, '3.7.8': 117, '3.7.9': 118, '3.8': 119, '3.8.9
': 120, '3.9': 121, '4': 122, '4.10': 123, '4.5': 124, '4.5.10': 125, '4.5.8': 126, '4.5
.9': 127, '4.6': 128, '4.6.10': 129, '4.6.10.0': 130, '4.6.11': 131, '4.6.9': 132, '4.6.9
.0': 133, '4.7': 134, '4.7.10': 135, '4.7.10.0': 136, '4.7.11': 137, '4.7.9': 138, '4.8':
139, '4.8.11': 140, '4.9': 141, '5': 142, '5.10': 143, '5.11': 144, '5.6.10': 145, '5.6.9
': 146, '5.6.9.0': 147, '5.7': 148, '5.7.0': 149, '5.7.10': 150, '5.7.11': 151, '5.7.11.1
': 152, '5.8': 153, '5.8.0': 154, '5.8.10': 155, '5.8.10.11': 156, '5.8.11': 157, '5.8.1
1.1': 158, '5.9': 159, '5.9.0': 160, '5.9.11': 161, '6': 162, '6.10': 163, '6.10.0': 164,
'6.10.1': 165, '6.11': 166, '6.7': 167, '6.7.11': 168, '6.7.9': 169, '6.8': 170, '6.8.0'
: 171, '6.8.1': 172, '6.8.11': 173, '6.9': 174, '6.9.0': 175, '6.9.0.2': 176, '6.9.1': 1
77, '6.9.11': 178, '7': 179, '7.0': 180, '7.10': 181, '7.10.0': 182, '7.10.0.3': 183, '7.
10.1': 184, '7.10.1.3': 185, '7.10.2': 186, '7.11': 187, '7.11.1': 188, '7.11.2': 189, '7
.8': 190, '7.8.0': 191, '7.9': 192, '7.9.0': 193, '7.9.1': 194, '7.9.10': 195, '7.9.11':
196, '7.9.2': 197, '8': 198, '8.0': 199, '8.0.1': 200, '8.0.2': 201, '8.0.3': 202, '8.1'
: 203, '8.10': 204, '8.10.1': 205, '8.10.1.4': 206, '8.10.2': 207, '8.10.3': 208, '8.11':
209, '8.11.0': 210, '8.11.1': 211, '8.11.2': 212, '8.11.2.4': 213, '8.11.3': 214, '8.9':
215, '9': 216, '9.0': 217, '9.0.2': 218, '9.0.3': 219, '9.0.3.5': 220, '9.0.4': 221, '9.
1': 222, '9.1.3': 223, '9.1.4': 224, '9.10': 225, '9.10.2': 226, '9.11': 227, '9.11.2':
228, '9.11.2.5': 229, '9.11.3': 230, '9.11.4': 231, '9.2': 232, 'A1': 233, 'A2': 234, 'A3
': 235, 'A4': 236, 'A5': 237, 'A6': 238, 'B-1': 239, 'B-2': 240, 'B-3': 241, 'B-4': 242,
'B-5': 243, 'B-6': 244, 'B1': 245, 'B2': 246, 'B3': 247, 'B4': 248, 'B5': 249, 'B6': 250,
'C#2': 251, 'C#3': 252, 'C#4': 253, 'C#5': 254, 'C#6': 255, 'C#7': 256, 'C2': 257, 'C3':
258, 'C4': 259, 'C5': 260, 'C6': 261, 'C7': 262, 'D1': 263, 'D2': 264, 'D3': 265, 'D4': 2
66, 'D5': 267, 'D6': 268, 'D7': 269, 'E-2': 270, 'E-3': 271, 'E-4': 272, 'E-5': 273, 'E-
6': 274, 'E-7': 275, 'E1': 276, 'E2': 277, 'E3': 278, 'E4': 279, 'E5': 280, 'E6': 281, '
E7': 282, 'F#1': 283, 'F#2': 284, 'F#3': 285, 'F#4': 286, 'F#5': 287, 'F#6': 288, 'F1':
289, 'F2': 290, 'F3': 291, 'F4': 292, 'F5': 293, 'F6': 294, 'F7': 295, 'G#1': 296, 'G#2':
297, 'G#3': 298, 'G#4': 299, 'G#5': 300, 'G#6': 301, 'G1': 302, 'G2': 303, 'G3': 304, 'G4
': 305, 'G5': 306, 'G6': 307},
      dtype=object)
```

Input Sequences

In [64]:

```
#Preparing input sequences

sequence_length = 120

network_input = []
network_output =[]

for i in range(0,len(Notes_Chords)-sequence_length,1):
    #input and output at a single time interval
    seq_in = Notes_Chords[i:i+sequence_length]
    seq_out = Notes_Chords[i+sequence_length]
    #Converting the seq_in and seq_out to int to make it more understandable
    network_input.append([Notes_Chords_to_int[note] for note in seq_in])
    network_output.append(Notes_Chords_to_int[seq_out])
print("Length of Network Input :",len(network_input))
print("Length of Unique Notes :",len(Notes_Chords))
```

```
Length of Network Input : 82984
Length of Unique Notes : 83104
```

In [65]:

```
import copy
network_input_copy = copy.deepcopy(network_input)
```

In [66]:

```
#reshaping it in (m,Tx,Features)
network_input= np.reshape(network_input,(len(network_input),sequence_length,1))

#normalising input for better training (getting < 1)
#total notes is length of unique Notes_chords as each item of network input is mapped (Se
e : Notes chord to int)
network_input = network_input/float(total_unique_notes)
```

In [67]:

```
network_input.shape
```

Out[67]:

```
(82984, 120, 1)
```

Training the Model

```
    I have used epochs = 2 to get the output faster . Train with more than 30 opchs for
better result.
    The outputs units I have used are having the max value 32(2^4) and reduced further.
    Try with higher powers like (2^8) for getting better results. (Takes hours to train)
```

In [68]:

```
import tensorflow as tf
from tensorflow import keras
from keras import utils
```

In [69]:

```
#Converting network output from vector to a matrix of dimension total_unique_notes
#i.e one-coded vector for each unique note - using to_categoriacal
network_output = keras.utils.to_categorical(network_output)
```

```
In [81]:

#model
#return_sequences=True for including previous activation a<t-1> for training in a<t> too
,or else we will get only the final activation a<Tx>
#Dropout - to prevent overfitting
m,Tx,feature = network_input.shape
model = keras.Sequential([
    keras.layers.LSTM(units=16,input_shape=(Tx,feature),return_sequences = True),
    keras.layers.Dropout(rate = 0.1),
    keras.layers.LSTM(units=32,return_sequences = True),
    #keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate = 0.1),
    keras.layers.LSTM(units = 16),
    keras.layers.Dense(units = 16),
    #keras.layers.Activation('relu'),
    #keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate = 0.1),
    #keras.layers.Dropout(rate =0.05),
    keras.layers.Dense(units = total_unique_notes),
    keras.layers.Activation('softmax')
])

model.summary()

Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_24 (LSTM)               (None, 120, 16)           1152
_________________________________________________________________
dropout_24 (Dropout)         (None, 120, 16)           0
_________________________________________________________________
lstm_25 (LSTM)               (None, 120, 32)           6272
_________________________________________________________________
dropout_25 (Dropout)         (None, 120, 32)           0
_________________________________________________________________
lstm_26 (LSTM)               (None, 16)                3136
_________________________________________________________________
dense_16 (Dense)             (None, 16)                272
_________________________________________________________________
dropout_26 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_17 (Dense)             (None, 308)               5236
_________________________________________________________________
activation_8 (Activation)    (None, 308)               0
=================================================================
Total params: 16,068
Trainable params: 16,068
Non-trainable params: 0
_________________________________________________________________

In [82]:

model.compile(loss ="categorical_crossentropy",optimizer = 'rmsprop')

In [83]:

class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epoch,logs ={}):
        if(logs.get("loss") < 2.0):
            print(" \n Loss is low so cancelling training ")
            self.model.stop_training = True

callbacks = myCallback()


model.fit(network_input,network_output,epochs =30,batch_size=64,callbacks=[callbacks])

Epoch 1/30
1297/1297 [==============================] - 132s 102ms/step - loss: 4.7874
```

```
Epoch 2/30
1297/1297 [==============================] - 131s 101ms/step - loss: 4.6852
Epoch 3/30
1297/1297 [==============================] - 131s 101ms/step - loss: 4.6731
Epoch 4/30
1297/1297 [==============================] - 130s 101ms/step - loss: 4.6638
Epoch 5/30
1297/1297 [==============================] - 153s 118ms/step - loss: 4.6530
Epoch 6/30
1297/1297 [==============================] - 137s 106ms/step - loss: 4.6443
Epoch 7/30
1297/1297 [==============================] - 131s 101ms/step - loss: 4.6299
Epoch 8/30
1297/1297 [==============================] - 135s 104ms/step - loss: 4.6146
Epoch 9/30
1297/1297 [==============================] - 142s 110ms/step - loss: 4.5954
Epoch 10/30
1297/1297 [==============================] - 144s 111ms/step - loss: 4.5759
Epoch 11/30
1297/1297 [==============================] - 137s 106ms/step - loss: 4.5639
Epoch 12/30
1297/1297 [==============================] - 144s 111ms/step - loss: 4.5533
Epoch 13/30
1297/1297 [==============================] - 142s 110ms/step - loss: 4.5512
Epoch 14/30
1297/1297 [==============================] - 138s 107ms/step - loss: 4.5552
Epoch 15/30
1297/1297 [==============================] - 142s 109ms/step - loss: 4.5601
Epoch 16/30
1297/1297 [==============================] - 148s 114ms/step - loss: 4.5528
Epoch 17/30
1297/1297 [==============================] - 141s 109ms/step - loss: 4.5404
Epoch 18/30
1297/1297 [==============================] - 1144s 882ms/step - loss: 4.5290
Epoch 19/30
1297/1297 [==============================] - 133s 103ms/step - loss: 4.5172
Epoch 20/30
1297/1297 [==============================] - 132s 102ms/step - loss: 4.5022
Epoch 21/30
1297/1297 [==============================] - 133s 103ms/step - loss: 4.4902
Epoch 22/30
1297/1297 [==============================] - 132s 102ms/step - loss: 4.4793
Epoch 23/30
1297/1297 [==============================] - 132s 102ms/step - loss: 4.4646
Epoch 24/30
1297/1297 [==============================] - 134s 103ms/step - loss: 4.4555
Epoch 25/30
1297/1297 [==============================] - 133s 103ms/step - loss: 4.4456
Epoch 26/30
1297/1297 [==============================] - 133s 102ms/step - loss: 4.4353
Epoch 27/30
1297/1297 [==============================] - 134s 103ms/step - loss: 4.4259
Epoch 28/30
1297/1297 [==============================] - 133s 102ms/step - loss: 4.4153
Epoch 29/30
1297/1297 [==============================] - 133s 103ms/step - loss: 4.4081
Epoch 30/30
1297/1297 [==============================] - 133s 102ms/step - loss: 4.4000

Out[83]:

<tensorflow.python.keras.callbacks.History at 0x15a053adfc8>

Music Generation
In [84]:

random_start = np.random.randint(0,len(network_input)-1)
int_to_Notes_Chord = dict((number,note) for number,note in enumerate(Unique_Notes_Chords)
)
#print(int_to_Notes_Chord)

# Shape of random pattern = (32,1)
```

```python
random_pattern = network_input_copy[random_start]
prediction_output = []

# Predicting 300 notes
for i in range(300):
    prediction_input = np.reshape(random_pattern, (1,len(random_pattern),1))
    #normalising
    prediction_input = prediction_input/ float(total_unique_notes)

    prediction = model.predict(prediction_input)

    index_note = np.argmax(prediction) #returns the index of most high prob
    resulting_note = int_to_Notes_Chord[index_note]

    #list of outputs
    prediction_output.append(resulting_note)

    random_pattern.append(index_note)
    random_pattern = random_pattern[1:len(random_pattern)]

print(prediction_output)
```

```
['9', '4.9', '4.9', '9', '9', '4.9', '9', '4.9', '9', '4.9', '4.9', '0', '0', '0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
'0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0',
'0', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3',
'0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3
', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '
0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3'
, '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0
.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3',
'0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3
', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '
0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3'
, '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0
.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3',
'0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3
', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '
0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3'
, '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0.3', '0
.3', '0.3', '0.3', '0.3', '0.3', '0.3']
```

In [85]:

```python
from music21 import instrument, note, stream, chord
offset = 0
# offset is the gap between each notes in music
final_output_notes = []

for pattern in prediction_output:
    #if pattern is chord
    if ("." in pattern) or pattern.isdigit():
        notes_in_chord = pattern.split(".")
        notes =[]
        for note_new in notes_in_chord:
            int_note = int(note_new)
            new_note = note.Note(int_note)
            new_note.storedInstrument = instrument.Piano()
            notes.append(new_note)
        new_chord = chord.Chord(notes)
        new_chord.offset = offset
        final_output_notes.append(new_chord)

    #if pattern is note
    else:
        new_note = note.Note(pattern)
        new_note.offset = offset
        new_note.storedInstrument =instrument.Piano()
        final_output_notes.append(new_note)
    #need to increase offset in order to not stack the notes
    offset+=1

print(final_output_notes)

final_musical_stream = stream.Stream(final_output_notes)

final_musical_stream.write('midi','output1.mid')
```

```
[<music21.chord.Chord A>, <music21.chord.Chord E A>, <music21.chord.Chord E A>, <music21
.chord.Chord A>, <music21.chord.Chord A>, <music21.chord.Chord E A>, <music21.chord.Chor
d A>, <music21.chord.Chord E A>, <music21.chord.Chord A>, <music21.chord.Chord E A>, <mu
sic21.chord.Chord E A>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord
.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <m
usic21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.
Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <mu
sic21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.C
hord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <mus
ic21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Ch
ord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <musi
c21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Cho
rd C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music
21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chor
d C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music2
1.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord
C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.
chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C
>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.c
hord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>
, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.ch
ord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>,
<music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chor
d.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <
music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord
.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <m
usic21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.
Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <mu
sic21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.C
hord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <music21.chord.Chord C>, <mus
ic21.chord.Chord C>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.ch
ord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.
Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chor
d C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C
E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->,
<music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <mus
ic21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21
.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.cho
rd.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.C
hord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <musi
c21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.
chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chor
d.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Ch
ord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord
C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E-
>, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->, <
music21.chord.Chord C E->, <music21.chord.Chord C E->, <music21.chord.Chord C E->]

Out[85]:

'output1.mid'
```

```
  File "<ipython-input-2-25ac14ea1975>", line 1
    Music creation by LSTM Network
                ^
SyntaxError: invalid syntax
```

In [2]:

```
#For understanding MIDI files MIDI
import music21
from music21 import converter, pitch, interval
import os
import glob
```

In [4]:

```
#Dataset - Midi files - Music Instrument Digital interface(MIDI)
song_dir = "Schubert_dataset"
song_list = os.listdir(song_dir)
song_list
```

Out[4]:

```
['schubert_D850_1.mid',
 'schubert_D850_2.mid',
 'schubert_D850_3.mid',
 'schubert_D850_4.mid',
 'schubert_D935_1.mid',
 'schubert_D935_2.mid',
 'schubert_D935_3.mid',
 'schubert_D935_4.mid',
 'schub_d760_1.mid',
 'schub_d760_2.mid',
 'schub_d760_3.mid',
 'schub_d760_4.mid',
 'schub_d960_1.mid',
 'schub_d960_2.mid',
 'schub_d960_3.mid',
 'schub_d960_4.mid',
 'schuim-1.mid',
 'schuim-2.mid',
 'schuim-3.mid',
 'schuim-4.mid',
 'schumm-1.mid',
 'schumm-2.mid',
 'schumm-3.mid',
 'schumm-4.mid',
 'schumm-5.mid',
 'schumm-6.mid',
 'schu_143_1.mid',
 'schu_143_2.mid',
 'schu_143_3.mid']
```

In [7]:

```
#getting in musical format
from music21 import note, chord, instrument
import numpy as np

Notes_Chords = []
for song in glob.glob(song_dir + '/*.mid'):
    global score
    score = converter.parse(song)
    notes_to_parse = None
    parts = instrument.partitionByInstrument(score)
    if parts:
        notes_to_parse = parts.parts[0].recurse()
    else:
        notes_to_parse = score.flat.notes

    for element in notes_to_parse:
        if isinstance(element,note.Note):
            Notes_Chords.append(str(element.pitch))
        if isinstance(element,chord.Chord):
            Notes_Chords.append(".".join(str(n) for n in element.normalOrder))
total_notes = len((Notes_Chords))
print("Total Length of the Notes List :",len(Notes_Chords))
print("Different Notes :",np.array(Notes_Chords))
```

```
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-7-9a4862b06148> in <module>
      6 for song in glob.glob(song_dir + '/*.mid'):
      7     global score
----> 8     score = converter.parse(song)
      9     notes_to_parse = None
     10     parts = instrument.partitionByInstrument(score)

~\anaconda3\lib\site-packages\music21\converter\__init__.py in parse(value, *args, **keyw
ords)
   1135         return parseData(value, number=number, format=m21Format, **keywords)
   1136     elif not isinstance(value, bytes) and os.path.exists(valueStr):
-> 1137         return parseFile(valueStr, number=number, format=m21Format,
   1138                          forceSource=forceSource, **keywords)
   1139     elif not isinstance(value, bytes) and os.path.exists(common.cleanpath(valueSt
r)):

~\anaconda3\lib\site-packages\music21\converter\__init__.py in parseFile(fp, number, form
at, forceSource, **keywords)
   1008     v = Converter()
   1009     fp = common.cleanpath(fp, returnPathlib=True)
-> 1010     v.parseFile(fp, number=number, format=format, forceSource=forceSource, **key
words)
   1011     return v.stream
   1012 

~\anaconda3\lib\site-packages\music21\converter\__init__.py in parseFile(self, fp, number
, format, forceSource, storePickle, **keywords)
    549         else:
    550             environLocal.printDebug('Loading original version')
[illegible]
```

```
--> 551                 self.parseFileNoPickle(fp, number, format, forceSource, **keywords)
    552                 if writePickle is True and fpPickle is not None and storePickle is T
rue:
    553                     # save the stream to disk...

~\anaconda3\lib\site-packages\music21\converter\__init__.py in parseFileNoPickle(self, fp
, number, format, forceSource, **keywords)
    483             self.subConverter.keywords = keywords
    484             try:
--> 485                 self.subConverter.parseFile(fp, number=number, **keywords)
    486             except NotImplementedError:
    487                 raise ConverterFileException('File is not in a correct format: %s' %
fp)

~\anaconda3\lib\site-packages\music21\converter\subConverters.py in parseFile(self, fp, n
umber, **keywords)
   1046             '''
   1047             from music21.midi import translate as midiTranslate
-> 1048             midiTranslate.midiFilePathToStream(fp, self.stream, **keywords)
   1049 
   1050         def write(self, obj, fmt, fp=None, subformats=None, **keywords):  # pragma: n
o cover

~\anaconda3\lib\site-packages\music21\midi\translate.py in midiFilePathToStream(filePath,
inputM21, **keywords)
   2149         mf.read()
   2150         mf.close()
-> 2151         return midiFileToStream(mf, inputM21, **keywords)
   2152 
   2153 

~\anaconda3\lib\site-packages\music21\midi\translate.py in midiFileToStream(mf, inputM21,
quantizePost, **keywords)
   2309         # create a stream for each tracks
   2310         # may need to check if tracks actually have event data
-> 2311         midiTracksToStreams(mf.tracks,
   2312                             ticksPerQuarter=mf.ticksPerQuarterNote,
   2313                             quantizePost=quantizePost,

~\anaconda3\lib\site-packages\music21\midi\translate.py in midiTracksToStreams(midiTracks
, ticksPerQuarter, quantizePost, inputM21, **keywords)
   2048                 # such as the time sig, tempo, or other parameters
   2049                 # environLocal.printDebug(['found midi track without notes:'])
-> 2050                 midiTrackToStream(mt,
   2051                                   ticksPerQuarter,
   2052                                   quantizePost,

~\anaconda3\lib\site-packages\music21\midi\translate.py in midiTrackToStream(mt, ticksPer
Quarter, quantizePost, inputM21, **keywords)
   1655         # need to build chords and notes
   1656         notes = getNotesFromEvents(events)
-> 1657         metaEvents = getMetaEvents(events)
   1658 
   1659         # first create meta events

~\anaconda3\lib\site-packages\music21\midi\translate.py in getMetaEvents(events)
   1602                 metaObj = midiEventsToTempo(e)
   1603             elif e.type in (MetaEvents.INSTRUMENT_NAME, MetaEvents.SEQUENCE_TRACK_NAM
E):
-> 1604                 metaObj = midiEventsToInstrument(e)
   1605             elif e.type == ChannelVoiceMessages.PROGRAM_CHANGE:
   1606                 metaObj = midiEventsToInstrument(e)

~\anaconda3\lib\site-packages\music21\midi\translate.py in midiEventsToInstrument(eventLi
st)
    698         try:
    699             if isinstance(event.data, bytes):
--> 700                 i = instrument.fromString(event.data.decode('utf-8'))
    701             else:
    702                 i = instrument.instrumentFromMidiProgram(event.data)

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa9 in position 10: invalid start by
```

```
te
```

In [6]:

```
#to get instrumets used in the midi file
from music21 import instrument
for song in glob.glob(song_dir + '/*.mid'):
    parts = instrument.partitionByInstrument(score) #or use (parts = score.parts.stream(
))
    for p in parts:
        global music_instrument
        music_instrument = p.partName
        print("Instruments: ",music_instrument)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-68dd2da1b0f2> in <module>
      2 from music21 import instrument
      3 for song in glob.glob(song_dir + '/*.mid'):
----> 4     parts = instrument.partitionByInstrument(score) #or use (parts = score.parts.
stream())
      5     for p in parts:
      6         global music_instrument

NameError: name 'score' is not defined
```

In [9]:

```
#Finding the unique notes or chords,then sorted mapping with a number
Unique_Notes_Chords = sorted(set(item for item in Notes_Chords))
print(Unique_Notes_Chords)
total_unique_notes = len(Unique_Notes_Chords)
print("Total Unique Notes:",total_unique_notes)

#Mapping every unique note with an integer
Notes_Chords_to_int = dict((note , number) for number, note in enumerate(Unique_Notes_Ch
ords))

#Notes_Chords_to_int
np.array(Notes_Chords_to_int)
```

```
[]
Total Unique Notes: 0
```

Out[9]:

```
array({}, dtype=object)
```

In [10]:

```
#Preparing input sequences

sequence_length = 120

network_input = []
network_output =[]

for i in range(0,len(Notes_Chords)-sequence_length,1):
    #input and output at a single time interval
    seq_in = Notes_Chords[i:i+sequence_length]
    seq_out = Notes_Chords[i+sequence_length]
    #Converting the seq_in and seq_out to int to make it more understandable
    network_input.append([Notes_Chords_to_int[note] for note in seq_in])
    network_output.append(Notes_Chords_to_int[seq_out])
print("Length of Network Input :",len(network_input))
print("Length of Unique Notes :",len(Notes_Chords))
```

```
Length of Network Input : 0
Length of Unique Notes : 0
```

In [11]:

```
import copy
```

```
network_input_copy = copy.deepcopy(network_input)
```

In [12]:

```
#reshaping it in (m,Tx,Features)
network_input= np.reshape(network_input,(len(network_input),sequence_length,1))

#normalising input for better training (getting < 1)
#total notes is length of unique Notes_chords as each item of network input is mapped (Se
e : Notes chord to int)
network_input = network_input/float(total_unique_notes)
```

In [13]:

```
network_input.shape
```

Out[13]:

```
(0, 120, 1)
```

Training the Model

```
I have used epochs = 2 to get the output faster . Train with more than 30 opchs fo
r better result.
The outputs units I have used are having the max value 32(2^4) and reduced further.
Try with higher powers like (2^8) for getting better results. (Takes hours to trai
n)
```

In [14]:

```
import tensorflow as tf
from tensorflow import keras
from keras import utils
```

In [15]:

```
#Converting network output from vector to a matrix of dimension total_unique_notes
#i.e one-coded vector for each unique note - using to_categoriacal
network_output = keras.utils.to_categorical(network_output)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-7185083fc175> in <module>
      1 #Converting network output from vector to a matrix of dimension total_unique_note
s
      2 #i.e one-coded vector for each unique note - using to_categoriacal
----> 3 network_output = keras.utils.to_categorical(network_output)

~\anaconda3\lib\site-packages\tensorflow\python\keras\utils\np_utils.py in to_categorical
(y, num_classes, dtype)
     73   y = y.ravel()
     74   if not num_classes:
---> 75     num_classes = np.max(y) + 1
     76   n = y.shape[0]
     77   categorical = np.zeros((n, num_classes), dtype=dtype)

<__array_function__ internals> in amax(*args, **kwargs)

~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py in amax(a, axis, out, keepdims, i
nitial, where)
   2665     5
   2666     """
-> 2667     return _wrapreduction(a, np.maximum, 'max', axis, None, out,
   2668                           keepdims=keepdims, initial=initial, where=where)
   2669

~\anaconda3\lib\site-packages\numpy\core\fromnumeric.py in _wrapreduction(obj, ufunc, met
hod, axis, dtype, out, **kwargs)
     88                 return reduction(axis=axis, out=out, **passkwargs)
     89
---> 90     return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
     91
     92

ValueError: zero-size array to reduction operation maximum which has no identity
```

In [17]:

```
#model
#return_sequences=True for including previous activation a<t-1> for training in a<t> too
,or else we will get only the final activation a<Tx>
#Dropout - to prevent overfitting
m,Tx,feature = network_input.shape
model = keras.Sequential([
    keras.layers.LSTM(units=16,input_shape=(Tx,feature),return_sequences = True),
    keras.layers.Dropout(rate = 0.1),
    keras.layers.LSTM(units=32,return_sequences = True),
    #keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate = 0.1),
    keras.layers.LSTM(units = 16),
    keras.layers.Dense(units = 16),
    #keras.layers.Activation('relu'),
    #keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate = 0.1),
    #keras.layers.Dropout(rate =0.05),
    keras.layers.Dense(units = total_unique_notes),
    keras.layers.Activation('softmax')
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm (LSTM)                  (None, 120, 16)           1152      
_________________________________________________________________
dropout (Dropout)            (None, 120, 16)           0         
_________________________________________________________________
lstm_1 (LSTM)                (None, 120, 32)           6272      
_________________________________________________________________
dropout_1 (Dropout)          (None, 120, 32)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 16)                3136      
_________________________________________________________________
dense (Dense)                (None, 16)                272       
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 0)                 0         
_________________________________________________________________
activation (Activation)      (None, 0)                 0         
=================================================================
Total params: 10,832
Trainable params: 10,832
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
model.compile(loss ="categorical_crossentropy",optimizer = 'rmsprop')
```

In [20]:

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epoch,logs ={}):
        if(logs.get("loss") < 2.0):
            print(" \n Loss is low so cancelling training ")
            self.model.stop_training = True
```

```
callbacks = myCallback()


model.fit(network_input,network_output,epochs =30,batch_size=64,callbacks=[callbacks])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-20-6a94123e330e> in <module>
      8 
      9 
---> 10 model.fit(network_input,network_output,epochs =30,batch_size=64,callbacks=[callba
cks])

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in _method_wrapp
er(self, *args, **kwargs)
    106   def _method_wrapper(self, *args, **kwargs):
    107     if not self._in_multi_worker_mode():  # pylint: disable=protected-access
--> 108       return method(self, *args, **kwargs)
    109 
    110     # Running inside `run_distribute_coordinator` already.

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in fit(self, x,
y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, cl
ass_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_b
atch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1047          training_utils.RespectCompiledTrainableState(self):
   1048       # Creates a `tf.data.Dataset` and handles batch and epoch iteration.
-> 1049       data_handler = data_adapter.DataHandler(
   1050           x=x,
   1051           y=y,

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\data_adapter.py in __init__(
self, x, y, sample_weight, batch_size, steps_per_epoch, initial_epoch, epochs, shuffle, c
lass_weight, max_queue_size, workers, use_multiprocessing, model, steps_per_execution)
   1102       self._steps_per_execution_value = steps_per_execution.numpy().item()
   1103 
-> 1104     adapter_cls = select_data_adapter(x, y)
   1105     self._adapter = adapter_cls(
   1106         x,

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\data_adapter.py in select_da
ta_adapter(x, y)
    963 def select_data_adapter(x, y):
    964   """Selects a data adapter than can handle a given x and y."""
--> 965   adapter_cls = [cls for cls in ALL_ADAPTER_CLS if cls.can_handle(x, y)]
    966   if not adapter_cls:
    967     # TODO(scottzhu): This should be a less implementation-specific error.

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\data_adapter.py in <listcomp
>(.0)
    963 def select_data_adapter(x, y):
    964   """Selects a data adapter than can handle a given x and y."""
--> 965   adapter_cls = [cls for cls in ALL_ADAPTER_CLS if cls.can_handle(x, y)]
    966   if not adapter_cls:
    967     # TODO(scottzhu): This should be a less implementation-specific error.

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\data_adapter.py in can_handl
e(x, y)
    620     handles_y = True
    621     if y is not None:
--> 622       handles_y = ListsOfScalarsDataAdapter._is_list_of_scalars(y)
    623     return handles_x and handles_y
    624 

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\data_adapter.py in _is_list_
of_scalars(inp)
    628       return True
    629     if isinstance(inp, (list, tuple)):
--> 630       return ListsOfScalarsDataAdapter._is_list_of_scalars(inp[0])
    631     return False
    632 
```

```
IndexError: list index out of range
```

In [21]:

```
<tensorflow.python.keras.callbacks.History at 0x15a053adfc8>
```

```
  File "<ipython-input-21-3e8af8d4ac20>", line 1
    <tensorflow.python.keras.callbacks.History at 0x15a053adfc8>
    ^
SyntaxError: invalid syntax
```

In [ ]:

```
midicsv [ -u -v ] [ infile [ outfile ] ]
```