Fanghan Wu
1800 N Oak Street
Arlington, VA, 22209
612-913-2527
fw183@georgetown.edu

Hiring Manager
Blue Orange Digital

February 25, 2020

Dear Hiring Manager,

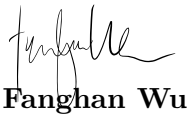
I write to apply for the position of Machine Learning Engineer. With a M.S. degree in Mathematics and Statistics from Georgetown University, I am proficient in Python, R, and SQL. Throughout my M.S. studies, I have focused on machine learning such as Neural Networks and Algorithms for Decision Making. For example, I led and completed a successful music project named "Schubot." In this project, we aimed to not only generate convincing music in the style of Schubert, but also provide inspiration to artists and composers. We implemented a neural networks model and produced music in the style of Schubert, with an 86% accuracy using only 6 pieces of music from Schubert.

Note: The results can be found on Soundcloud (link here).

I believe that the position of Machine Learning Engineer matches well with my interest, academic training and work experience and hope that I would have an opportunity to discuss with you in more details about my qualifications and how I may be able to contribute to your team's success.

Enclosed please find my resume and the music project "Schubot" mentioned above. Should you have any question, please feel free to contact me at fw183@georgetown.edu (e-mail) or 612-913-2527 (Cell).

Thank you in advance for your consideration and I look forward to hearing from you soon.



Fanghan Wu

Fanghan Wu

fw183@georgetown.edu | 612-913-2527 | LinkedIn: fanghan-wu | Github: Evishka
1800 N Oak Street, Arlington, VA

Career Objective

I am a recent graduate with a MS degree in mathematics and statistics. I am a highly motivated individual with good analytical and interpersonal skills. I am seeking a full-time OPT position as a data engineer or relevant to utilize my existing skill set.

Education

Georgetown University — Washington, DC
Master of Science in Mathematics and Statistics — *Graduated: May 2019*

St. Olaf College — Northfield, MN
Bachelor of Arts in Mathematics; Double Major in Philosophy — *Graduated: May 2017*

Budapest Semester in Mathematics — Budapest, Hungary
Honors — *Jan 2016 - May 2016*

Skills

- **Languages**: English (full proficiency), Chinese (full proficiency), German (conversational level)
- **Programming**: Python, SQL, R, LaTeX, Matlab, SAS, HTML, VB
- **Technologies**: AWS, GitHub, Docker
- **Libraries**: TensorFlow, PyTorch, Keras, Scikit-Learn, Numpy, Pandas, Spark, Jupyter

Projects

- **SchuBot**: A neural networks model for classical music generation in the style of Schubert
  - Implemented a 2-layer neural nets using keras to achieve an 86% accuracy on the dataset of Schubert's 6 moments musicaux
- **Handwritten Numbers Recognition**: To ensure accuracy of handwritten recognition for organization seeking greater automation, built and trained a softmax model from scratch for handwritten numbers recognition on the MNIST dataset and achieved a 91% accuracy
- **PM2.5 Forecast**: To provide a reliable forecast on air quality, built an ARIMA model to analyze and predict Beijing's PM2.5 level, based on PM2.5 records of Beijing over 5 years
- **eHarmony Matches Prediction**: To improve user satisfaction and successful matches for dating sites, trained an SVM model on a dataset from **eHarmony** to predict successful interaction between two users and achieved an 89% accuracy
- **Poker Hand Classification**: Analyzed one million instances of poker hands data and trained a classification model to classify 10 classes of poker hands, and achieved a 93% accuracy on hand prediction

Relevant Experience

Ridgeview Energy — Plano, TX
Business Analyst Intern — *Oct 2019 - present*

- Conducted research and statistical analysis for funding purposes
- Applied Python and SQL to automate data cleaning process, and analyzed historical trends and assessed growth potential for 12 companies in oil and gas industry

Ernst & Young — Beijing, China
Advisory Intern — *May 2018 - Aug 2018*

- Participated in writing and revising business reports of an e-commerce platform, making sample webpages and presenting business reports to accommodate frequent requirements of the client
- Assisted with the design and development of a B2C Android application, for customers to pay for car refuels and schedule pickups from the client's gas station convenient stores

St. Olaf College — Northfield, MN
Teaching Assistant — *Sep 2014 - Dec 2016*

- Graded homework assignments for 250 students with quick and frequent deadlines, provided feedback individually, developed plots to analyze student performance, and submitted weekly reports to professors
- Organized study groups twice a week for 8 to 10 students in order to answer inquiries and explain material to students
- Held one-on-one tutorial sessions upon request, to provide more personalized assistance such as additional exercises and presentation preparation

Additional Experience

- Worked as a **Graduate Teaching Assistant** at **Georgetown University** for 60 first-year graduate students to grade homework and answer email inquiries in a timely manner, and held office hours throughout the semester
- Led a group of 5 undergraduate students to optimize income for a car insurance company, generated an insurance plan that increased the income by 75%, and presented the project at **Traveler's Actuarial Competition 2015**
- Selected as a **Student Representative** at **Washington Statistical Society** to attend monthly meetings and help increase the diversity of the society. Held a seminar *Why Writing about Statistics is Hard* by Dr. Regina Nuzzo
- Selected as a **Program Representative** to represent the M.S. Program in Mathematics and Statistics in **Georgetown Graduate Students Government (GradGov)**; assisted in organizing and managing events, activities, and policies
- Coordinated with a 5-member team to discover potential clients for **BDA China** by analyzing financial statements of several car companies, and performing an hour-long interview with the CEO of a South Korean biobetter company

Schubot—Generating Classical Music Using RNN

Fanghan Wu & Fyona Sun

Project Introduction Neural networks are used to generate text based on the style of an author or to create an image in a certain artist's style. In this project we would like to apply a similar technique to train a neural network that can compose music in a certain type. Even after hundreds of years of study, reaching consensus on a formal theory for stylistic composition has proven to be difficult.

A neural-network based solution was developed by Hermann Hild, Johannes Feulner, and Wolfram Menzel that harmonizing chorales in the style of J.S.Bach. in 1992. They create several neural networks each trained for solving a specific task. The learning task is decomposed along two dimensions, the chord skeleton and its harmonic skeleton. A harmonic skeleton is first computed then refined. Another approach uses Hidden Markov Models (HMMs) instead of neural networks. In this approach chords are represented as lists of intervals and form the states of the Markov models. However this approach cannot generate high quality music pieces.

The most recent advances in generating Bach like music is the Coconet developed by Google. Their model is a straightforward convolutional neural network with batch normalization and residual connections. It takes a piece from Bach, randomly erase some notes, and train the model to learn the missing notes from context. Their goal is to for the neural network to complete several tasks including harmonizing melodies, creating smooth transitions, rewriting and elaborating existing music, and composing from scratch.

For the purpose of this project, we trained our model on the 6 moments musicaux, D. 780 (Op. 94) a collection of six short pieces for solo piano composed by Franz Schubert. Our goal is for the model to generate highly convincing classical pieces in the style of Schubert, which often takes years of practice for musicians.

- Our project is done in R studio with Keras and TensorFlow.

Approach There are several challenges for building the model. First of all, the traditional feed-forward network could not help us with the task, since it has no ability to store past information. We would like to find a network that can learn to predict notes at time t+1 using notes at time t as inputs. Second, we need to find a method to represent music that can not only capture the melody and texture of classical music, but also efficiently feed into a neural network model. Third we want to examine the performance of the model and the quality of generated music.

Our approach is based on a recurrent neural network model with character sequence. We first downloaded the 6 moments musicaux in MIDI format, and then converted those music pieces in a character based format as our input. In this way, we transformed a music generating project into a text generating project. There are several text based notations to represent music, ABC notation is one of those. ABC notation is a shorthand form of musical notation. In basic form it uses the letters A through G, letter notation, to represent the given notes, with other elements used to place added value on these – sharp, flat, the length of the note, key, ornamentation. Since the number of limited characters is finite and in tens, one hot representation is convenient to use directly for the characterization of an output of a limited quantity of possible outcomes. We performed the one-hot encoding and cut the text in semi-redundant sequences of maxlen characters as our inputs. The training goal is for the model to predict the next character based on the given sequence.

We feed in inputs one at a time, and let the network combine them using the state passed from each time step. However the memory is very short. Any value that is output in one time step becomes input in the next, but unless that same value is output again, it is lost at the next tick. To solve this, we can form a Long Short-Term Memory (LSTM) layer. LSTM introduces a "memory cell" value that is passed down for

multiple time steps, and being sent in parallel with the activation output.

installation

Data Summary

We acquired midi files from Schubert's 6 moments musicaux, and converted them to .txt format using ABC notations. In ABC notations, there are two parts for each piece of music. Lines in the first part begin with a capital letter followed by a colon, indicate various aspects of the music such as meters, time signature, default note length, and key. The second part represents the melody. In our very first approach, we used ABC notations of the 6 moments as is – the first moment followed by the second, and the third, etc. However, since the ratio between the first and the second parts in each moment is highly uneven, i.e. there are far more characters in the melody part than in the aspects part, and since there are only 6 pieces of music in total, our model learned about the melodies but not the aspect parts. However, in order to generate music in the style of Schubert, the model needs to generate everything – not only the notes themselves, but also other things such as keys, speed, and meters – we wanted to modify our input data such that there's enough information for the model to learn both the aspects and melodies. To solve this, we divided the 6 moments into smaller pieces, with each piece being roughly 400-600 characters. In this way, the ratio between the first and second parts become more even. The 6 moments were divided into 96 smaller pieces. Then, we duplicated the data into 500 of such pieces in total, and shuffled them. The finalized version of our input .txt data contains 212,591 characters in total, and 52 unique characters. We chose 256 as our batch size, so there are 16 batches.

```
#read the raw data and prepare for further uses
input <- readtext("~/Desktop/schubot/abc.txt")
input <- input[-c(1)]
input <- as.character(input)
#str_replace_all(input,"[\r\n]","")
input <- strsplit(input,"")
input<-unlist(input)
summary(input)
```

```
##    Length     Class      Mode
##    222463 character character
```

```
#make sequences from the data
maxlen <- 50
chars <- input %>%
  unique() %>%
  sort()
dataset <- map(
  seq(1,length(input)-maxlen-1, by=3),
  ~list(sentence=input[.x:(.x+maxlen-1)],next_char=input[.x+maxlen])
)
dataset <- transpose(dataset)

x <- array(0,dim=c(length(dataset$sentence),maxlen,length(chars)))
y <- array(0,dim=c(length(dataset$sentence),length(chars)))

for (i in 1:length(dataset$sentence)){
  x[i,,] <- sapply(chars,function(x){
    as.integer(x==dataset$sentence[[i]])
  })
  y[i,] <- as.integer(chars==dataset$next_char[[i]])
}
```

Cost Function, Performance metrics and model architecture

The loss function we chose is cross-entropy, which measures how well a probability distribution predicts a sample. Since the music score is a combination of finite characters and the output is one of those characters so it can be thought of as multi-class classification problem. Thus using Cross-Entropy as a loss function makes sense in this scenario.

The perplexity measure is used extensively in language models.

$$e^{-\frac{1}{N}\sum_{i=1}^{N} lnp_i}$$

The performance metrics contains accuracy and loss as shown in the image below.

```
$loss
 [1] 2.6012162 1.5783803 1.2544998 1.0298964 0.8868485 0.7919033 0.7249234 0.6676256 0.6274479 0.6068195
[11] 0.5822473 0.5631374 0.5506923 0.5391529 0.5223857 0.5230563 0.5120909 0.5091720 0.4974565 0.5022773
[21] 0.4988729 0.4895653 0.4900902 0.4808048 0.4716959 0.4718956 0.4808466 0.4858307 0.4758927 0.4734831
[31] 0.4671677 0.4697016 0.4702551 0.4673091 0.4721937 0.4576832 0.4610889 0.4569301 0.4572238 0.4354014
[41] 0.4758464 0.4776378 0.4727229 0.4735843 0.4644538 0.4649104 0.4507749 0.4621447 0.4627458 0.4680516
[51] 0.4856425 0.5193686 0.5058885 0.5263641 0.5063023 0.5172644 1.0244414 1.9406838 3.3298282 3.6956408
[61] 3.6768198 3.8132178 3.5120243 3.5015084 3.7438198 3.6567104 3.7786904 3.7285616 3.7227936 3.7262893

$acc
 [1] 0.2972885 0.5169168 0.6062783 0.6723644 0.7163182 0.7475405 0.7687976 0.7858625 0.7993281 0.8052705
[11] 0.8114670 0.8192019 0.8229565 0.8260336 0.8328229 0.8306209 0.8354200 0.8366480 0.8401344 0.8380595
[21] 0.8408542 0.8436490 0.8432961 0.8440865 0.8489562 0.8487868 0.8450463 0.8453569 0.8486316 0.8509605
[31] 0.8505371 0.8491679 0.8482504 0.8504947 0.8480246 0.8534024 0.8528661 0.8534589 0.8536706 0.8607422
[41] 0.8477564 0.8480246 0.8492950 0.8499019 0.8526402 0.8531625 0.8543763 0.8534871 0.8529508 0.8506641
[51] 0.8475729 0.8361399 0.8417576 0.8374949 0.8413624 0.8390193 0.7462842 0.5752679 0.3609045 0.2717828
[61] 0.2552119 0.2371025 0.2446963 0.2397984 0.2169040 0.2135164 0.1961551 0.1932898 0.1896905 0.1892811
```

```
$loss
  [1] 3.2669211 2.7095987 1.7049349 1.4543510 1.2839514 1.1506193 1.0396788 0.9565400 0.8963177 0.8352444
 [11] 0.7942973 0.7548167 0.7337539 0.7052535 0.6754570 0.6627712 0.6428223 0.6259756 0.6188816 0.6001842
 [21] 0.5998251 0.5945580 0.5734990 0.5776162 0.5612077 0.5620809 0.5574520 0.5537378 0.5492575 0.5463601
 [31] 0.5439112 0.5336167 0.5345084 0.5470423 0.5350768 0.5335698 0.5277438 3.0677231 0.6498552 0.5767778
 [41] 0.5631986 0.5435802 0.5689449 0.5531171 0.5414790 1.8042920 2.6602818 1.9801650 1.4156030 3.4934623
 [51] 4.5500111 3.1454717 2.6984371 2.4251559 2.4219704 2.5819154 2.6418500 2.7036555 2.6847348 2.5765663
 [61] 2.5415165 2.5763865 2.6011383 2.7667077 2.8328620 2.9342310 2.7783125 2.6589762 2.7594457 2.8178986
 [71] 2.8874138 3.0239563 3.0380683 3.0724737 3.0945921 3.0978863 3.0572778 3.0502549 3.1151535 3.0643592
 [81] 3.0421911 3.1839353 3.1300297 3.0911170 3.0712908 3.1467236 3.1769253 3.1349202 2.8366697 2.9082028
 [91] 2.8145692 2.9995714 3.0465476 3.0119250 3.0500961 3.1247156 3.1501152 3.1182438 3.1525785 3.1051718

$acc
  [1] 0.1521730 0.2600392 0.4855251 0.5546600 0.6021003 0.6391943 0.6723220 0.6977430 0.7184637 0.7361074
 [11] 0.7486697 0.7609497 0.7667367 0.7750505 0.7841122 0.7893489 0.7948961 0.8001327 0.8043672 0.8085875
 [21] 0.8093356 0.8114811 0.8165625 0.8182280 0.8226883 0.8223355 0.8245091 0.8233517 0.8270216 0.8277979
 [31] 0.8269510 0.8305080 0.8315384 0.8268240 0.8327523 0.8299716 0.8338814 0.5253010 0.8003867 0.8202465
 [41] 0.8240010 0.8282073 0.8252290 0.8272474 0.8294212 0.6460118 0.4957867 0.5731930 0.6702330 0.4691236
 [51] 0.2881138 0.3237964 0.3220320 0.3414823 0.3326182 0.2930399 0.2775982 0.2597146 0.2612531 0.2835547
 [61] 0.2932940 0.2833147 0.2824678 0.2430590 0.2302285 0.2082939 0.2445975 0.2728132 0.2482815 0.2357757
 [71] 0.2183296 0.1915113 0.1877285 0.1783421 0.1735148 0.1691956 0.1769870 0.1800923 0.1667396 0.1811086
 [81] 0.1836069 0.1725267 0.1655257 0.1728937 0.1767753 0.1632109 0.1569862 0.1629991 0.2208562 0.2171581
 [91] 0.2350135 0.1968891 0.1821390 0.1899445 0.1804028 0.1739100 0.1666549 0.1691815 0.1641989 0.1702824
```

The model is constructed by four different types of layers. A LSTM layer: a Recurrent Neural Network layer that takes a sequence as an input and can return either sequences (return_sequences=True) or a matrix. A Dropout Layer: a regularisation technique that consists of setting a fraction of input units to 0 at each update during the training to prevent overfitting. A Dense layer: a fully connected neural network layer where each input node is connected to each output node An Activation layer, that determines what activation function our neural network will use to calculate the output of a node.

In the last layer we use Softmax activations. The number of Softmax activation notes in last layer will be equal to the number of all unique characters in the training data. Each RNN can be a LSTM which contains 'tanh' activation unit at input-gate which is a differentiable function.

We constructed our single RNN layer exactly like this where we have 256 LSTM nodes in one layer of RNN. At each time step all of the RNN nodes generate output which will be an input to the next layer and also the same output will again be any input to the same RNN unit.

Since each LSTM layer require same input shape the 'return_sequences' has to be set to TRUE. Then each RNN unit will generate output for each character means at each time step. We want our RNN unit to

generate output as the next character when given input as previous character in the sequence. We want to experiment how many LSTM to be added units here so that each unit will learn different aspect of the sequence and create a more robust model overall.

We are also interested in seeing how the parameters in the dropout layer influence the performance of the model.

```
#model
model <- keras_model_sequential()
model %>%
  layer_lstm(256,input_shape=c(maxlen,length(chars)),return_sequences = T) %>%
      layer_dropout(0.5) %>%
      layer_lstm(256,input_shape=c(maxlen,length(chars))) %>%
      layer_dropout(0.5) %>%
  layer_dense(length(chars)) %>%
  layer_activation("softmax")
#load_model_weights_hdf5(model,"~/Math514_FyonaSun/model-.5-wt.h5",by_name = T)

model %>% compile(
  loss="categorical_crossentropy",
  metrics = "acc",
  optimizer=optimizer_rmsprop(lr=0.01)
)
```

training Our model contains dropout layers, LSTM layers, a dense layer and an activation layer. We chose softmax activation, because it's good for multi-class neural networks. There were several other things we concerned about in our trainings: dropout parameter, and diversity of the output. Since the dropout is not linearly correlated with performance/loss, we experienced different dropout values when trying to find the best model. Diversity levels indicate the divertness of the output. The smaller the diversity level is, the more diverse the input, but the more likely for errors to occur. On the other hand, although less diversity gives better performances, the model is prone to overfitting. By default, diversity is equal to 1. However, we tried different diversity levels just to experience with more varieties of outcomes.

```
#training
pred <- function(preds,temp = 1){
  preds <- log(preds)/temp
  exp_preds<- exp(preds)
  preds <- exp_preds/sum(exp_preds)
  rmultinom(1,1,preds) %>%
    as.integer() %>%
    which.max()
}

on_epoch_end <- function(epoch,logs){
  cat(sprintf("epoch: %02d ------------\n\n",epoch))
  for(diversity in c(0.5,1)){
    cat(sprintf("diversity: %f ------------\n\n",diversity))
    start_index <- sample(1:(length(text)-maxlen),size=1)
    sentence <- text[start_index:(start_index+maxlen-1)]
    generated <- ""
    for (i in 1:640){
      x <- sapply(chars,function(x){
        as.integer(x==sentence)
      })
      x<-array_reshape(x,c(1,dim(x)))
      preds<-predict(model,x)
```

```
      next_index<-pred(preds,diversity)
      next_char<-chars[next_index]
      generated<-str_c(generated,next_char,collapse = "")
      sentence<-c(sentence[-1],next_char)
    }
    cat(generated,file='m.5.txt',append = TRUE)
    cat("\n\n",file='m.5.txt',append = TRUE)
    print(generated)
  }
}

print_callback<-callback_lambda(on_epoch_end = on_epoch_end)

#history<- model %>% fit(
#  x,y,
#  batch_size=256,
#  epochs=1,
#  callbacks=print_callback
#)

save_model_hdf5(model,'model.h5')
save_model_weights_hdf5(model,'model-wt.h5')
```

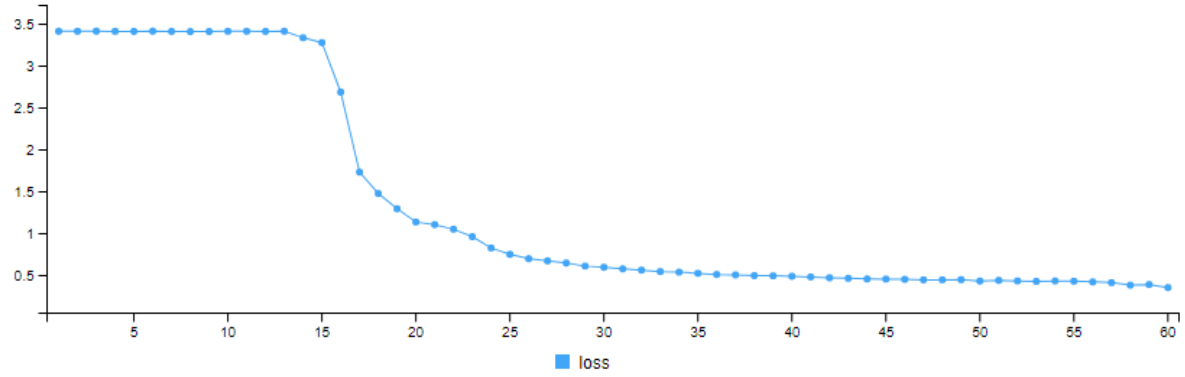
Model Tuning Our first model contained 4 layers - a dropout layer at dropout = 0.3, a LSTM layer with 512 nodes, and multiple diversity levels - 0.3, 0.5, 0.7, 1, and 1.2. We used the raw data of Schubert's 6 moments musicaux, and proceed with 1000 epochs. Our lowest loss was around the 800th epoch, with a loss around 0.6.

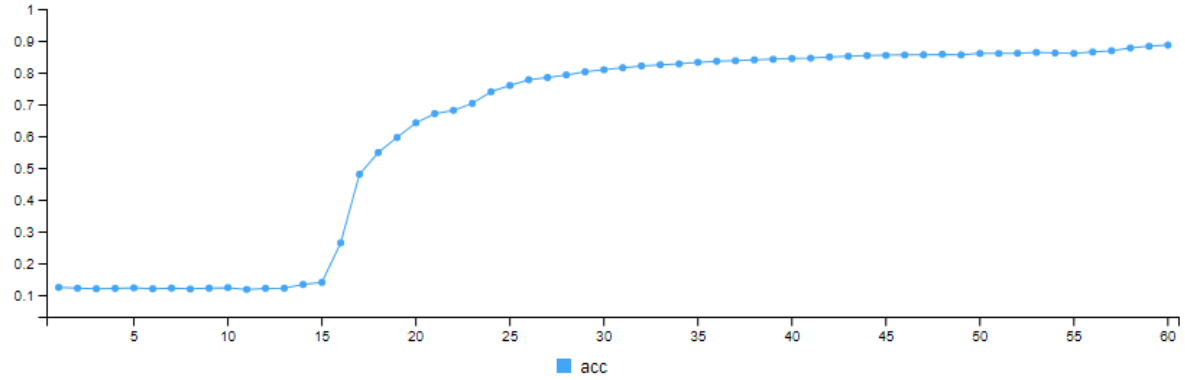
Then, as we modified the input data, we decided to increase the number of dropout and LSTM layers, as well as to decrease the number of epochs, as we had more input data after the modification. We trained three more models: Model 1: two layers of LSTM with 256 nodes, dropout = 0.5, diversity = 0.5, 1, epoch = 60 Model 2: two layers of LSTM with 256 nodes, dropout = 0.2, diversity = 0.3, 1, epoch = 70 Model 3: three layers of LSTM with 256 nodes, dropout = 0.2, diversity = 0.3, 1, epoch = 100 As a result, Model 2 achieved a much higher accuracy (0.86) than Model 1 (0.74). We therefore trained Model 3, however, it did not work as well as we expected, with only a 0.83 accuracy. Therefore, we concluded that Model 2 worked the best for us.

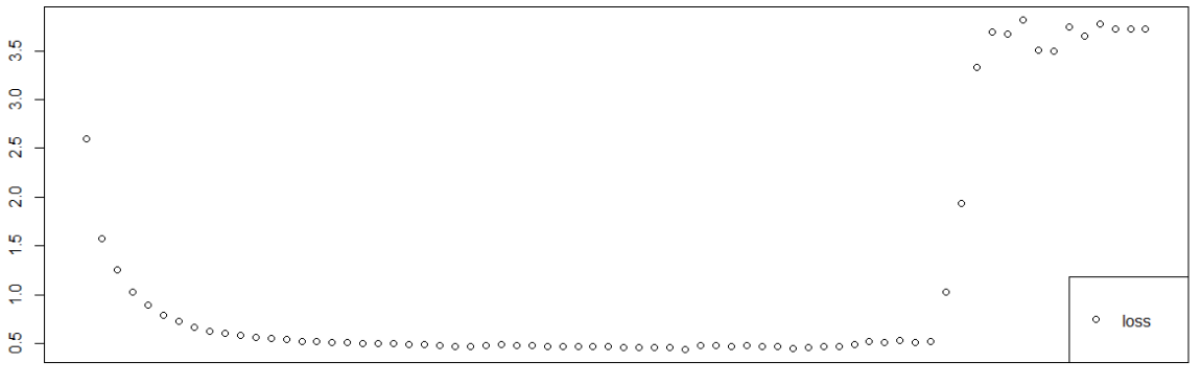
```
#layer_input(shape = c(1,50,length(chars)))
#one lstm layer with 0.3 droupout 1000 epoch
#model1<- load_model_hdf5("~/Math514_FyonaSun/model-2.h5")
#load_model_weights_hdf5(model1,"~/Math514_FyonaSun/model-2-wt.h5",by_name = T)
#2 lstm layer with 0.5 droupout 60 epoch
#model2<- load_model_hdf5("~/Math514_FyonaSun/mode-.5-60.h5")
#load_model_weights_hdf5(model2,"~/Math514_FyonaSun/model-.5-60-wt.h5",by_name = T)
#2 lstm layer with 0.2 droupout 70 epoch
#model3<- load_model_hdf5("~/Math514_FyonaSun/model-l2-0.2.h5")
#load_model_weights_hdf5(model3,"~/Math514_FyonaSun/model-l2-0.2-wt.h5",by_name = T)
#3 lstm layer with 0.2 droupout 100 epoch
#model4<- load_model_hdf5("~/Math514_FyonaSun/model-l3.h5")
#load_model_weights_hdf5(model4,"~/Math514_FyonaSun/model-l3-wt.h5",by_name = T)
```

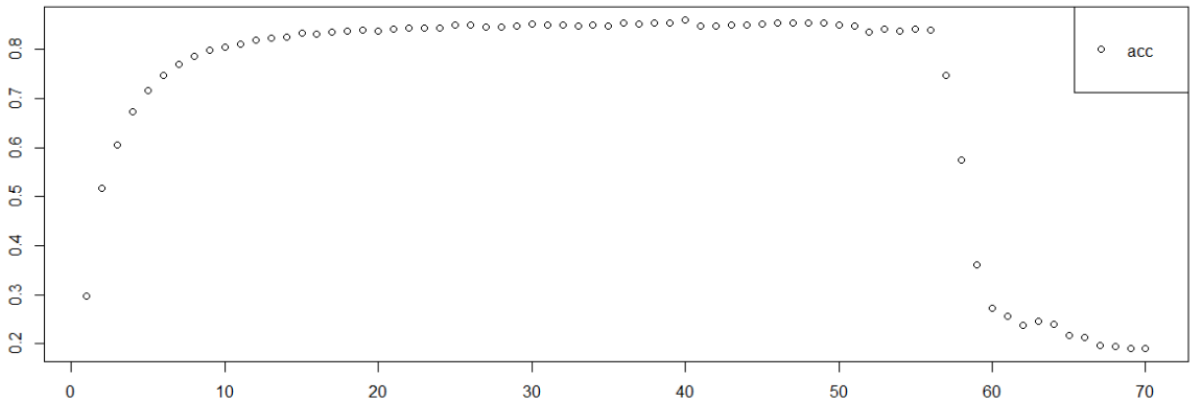
Training Curves

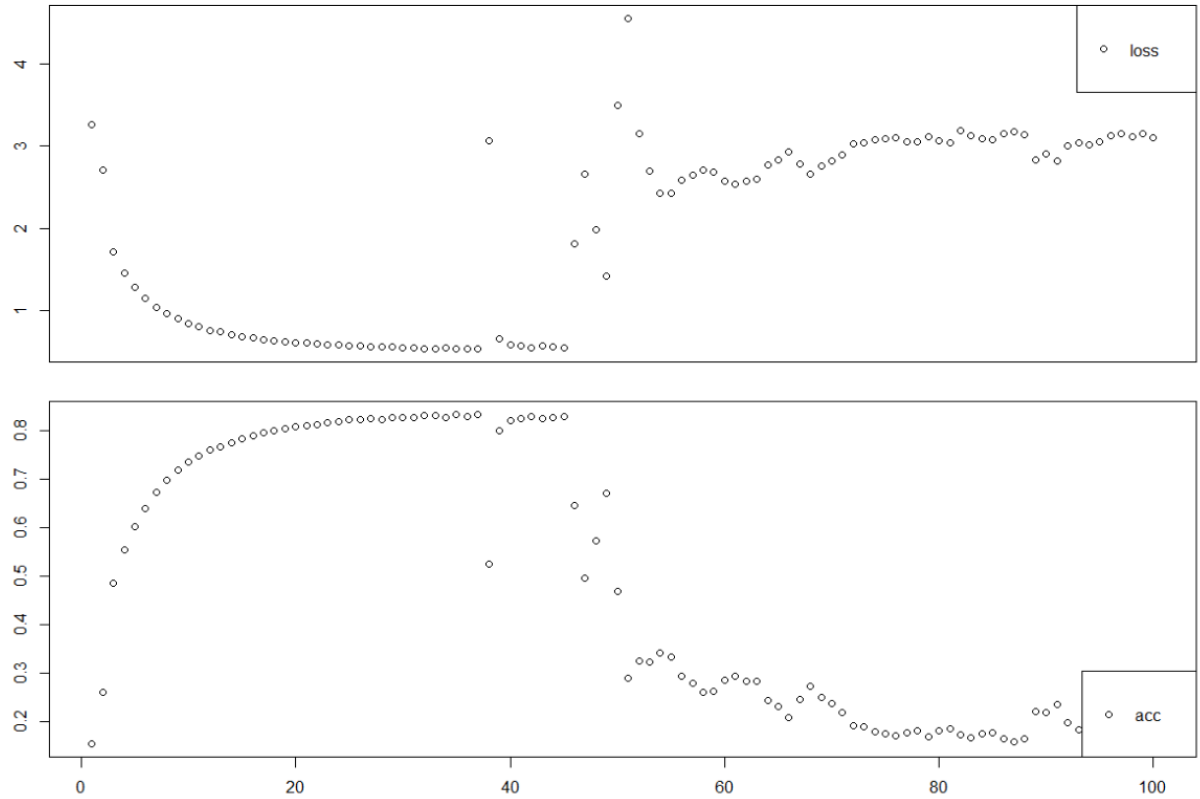
```
#plot(history)
#plot(history.5, main="2 lstm layer with 0.5 droupout 60 epoch")
```


loss


acc


loss


acc



Results and future work Here we apply the model to automatically generate a piece of music with length = 640, with zero initial state.

```
x <- array(0,dim=c(50,length(chars)))
generated <- ""
text = ''
# x <- array(0,dim=c(1,50,length(chars)))
x <- matrix(rexp(50*length(chars)), nrow=50)
x <- x/rowSums(x)
#x<- array(matrix(rexp(50*length(chars)), ncol=50))
x<-array_reshape(x,c(1,50,length(chars)))
new = which.max(x[,1,])
print(new)
```

```
## [1] 40
```

```
for(i in 1:640){
  if(i>50){
    y = x[,2:50,]
    inc = rep(0,length(chars))
    inc[new]=1
    x = rbind(y,inc)
    x=array_reshape(x,c(1,50,length(chars)))
  }
  else{
    x[,i,] = 0
    x[,i,new] = 1
    }
  res<- predict(model,x)
```

```
  new <-  which.max(res[,])
  text<- str_c(text,chars[new],collapse = "")
}
#print(text)
```

In summary, we have generated convincing music in the style of Schubert using only 6 pieces of music from Schubert. Further work can be done to take the performance of our model to the next level.

For example, we can give the model a non-empty initial state, either a character or a string of characters (a melody), and then let the model composing the rest. In addition, we can train the model with more compositions from Schubert to improve the matching accuracy of Schubert's style, and generate more diverse and interesting melodies.

Note: The results can be found on Soundcloud (https://soundcloud.com/user-547733484/sets/machine-learning).