Generating Music with LSTM Networks

Garrett Gibo **Siqi Huang** **Zekria Sidiqi**

Abstract

Recurrent Neural Networks(RNN) have been powerful to explore data with temporal structure. In this assignment, we explore the music composition with Long Short Time Memory (LSTM) and vanilla RNN. For LSTM, we experiment with different temperatures and number of hidden units. We also compare vanilla RNN with same hyperparameters as that of LSTM.

1. Introduction

In our project to generate music using RNN, we experiment mostly with the LSTM model and compare the initial best LSTM model with a simple vanilla RNN. After obtaining a best LSTM model, we generate music with temperatures of 0.5, 1, and 2. Other than comparing the music generated with different temperatures, we also compare the LSTM model using different number of hidden units. We also experiment with maximum at each time-step with temperature of 0.7.

Except for all the experiments with LSTM model, we use vanilla RNN model using tanh and relu as the hidden layers and having the other hyperparameters as that of our best LSTM model. We observe that vanilla RNN model using tanh fails to generate music sample and the one using relu successfully generate music samples.

2. Generation of Sample Music

Figures 2.1 - 2.12 show the ABC notation and respective sheet music afterwards for songs generated using T=2, T=1, T=0.5 respectively. For this portion, the hyperparameters used were: 1 LSTM layer with 100 units, Adam optimizer with default learning rate of 0.001 and with weight decay or momentum.

Before this part we experimented with different number of hidden layers and different optimizers like RMSprop with regularization and momentum. But surprisingly the simplest LSTM model with 1 hidden layer 100 hidden units and Adam optimizer with the default learning rate 0.001 provided the lowest validation loss. So going forward we used this model architecture and these hyperparameters to generate the six songs.

Starting with T=2 the songs generated were always unplayable. However to see the effects of this temperature value we restricted the priming characters to 'X:1\nT:Ductie\nM:6/8\nL:⅛\nK:Bb' including the three import fields in the header of ABC files meter (M), length (L), and key (K) forced the model to generate a somewhat playable tune. We noticed with T=2 the songs didn't have great fluidity compared to T=1 and T=0.5.

For T=1 we noticed a playable song was generated for every generation so we removed the strict restriction on the header and let the model decide on values of M, L, K. For these two songs we primed the songs on ‘<start>\nX:1\nT:Contredanse\n’. This resulted in great tunes. The first sample song even had parallel tunes playing in the background towards the end of the song.

Using T=0.5 the songs generated were had the best fluidity. There the most cut offs in tunes in T=2 and a little in T=1 but none in T=0.5. But with T=0.5 it seemed like only a few keys were being used throughout the song compared to T=1 which kinda made it seem repetitive.

With these findings we infer that a temperature value between 1 and 0.5 will probably have a good balance of variation and fluidity.

3. Best Model of Music Generation

As our model stops after two consecutive increases in validation losses, our best model early stops at epoch 23. **Figure 3.1** shows the training and validation losses of our best model. We obtain a test loss of 1.383 for our best model. According to the plot from **Figure 3.1**, we observe that while the training losses drop smoothly, while the validation losses fluctuate a lot after epoch 5.

4. Experimentation of Different Number of Hidden Units

Figures 4.1 - 4.5 display the training vs validation losses for 50, 75, 150, 201, and 499 hidden units respectively. For this portion we used the model of Part 2, i.e. LSTM 1 hidden layer and Adam optimizer with lr=1e-3. Like in the previous parts, we let the model run for max 25 epochs while implementing early stopping with 2 consecutive increases in validation loss.

Looking at plots we can see there’s no real significant standout changes in the losses graphs. Besides that, there’s a pattern of average validation loss decreasing as we increased the number of hidden units. The lowest validation losses for each number of increasing hidden units were: 1.533, 1.485, 1.369, 1.353, and 1.312 respectively.

Although not required we also computed the average test loss for each number of hidden units and they came out to: 1.457, 1.485, 1.313, 1.283, and 1.263 also respectively. Surprisingly the average test losses were lower than the smallest validation loss compared to 100 hidden units which was slightly higher (by 0.003).

From this portion we learned that the higher number of hidden units increased the performance of the model all around with lower losses. So going forward to the next part, we used the model with 499 hidden units.

5. Experimentation of Time Step Maximum

Using the model from the last part with 499 hidden units, the same hyper parameters from before, and the following priming characters: '<start>\nX:1\nT:Co' we generated the two sample songs one using sampling with T=0.7 and the other using the maximum output. **Figure 5.1** shows the sheet music for the song using sampling from the softmax distribution method. We can see it looks like a great piece and the song itself was pleasing to listen to as well.

Figure 5.2 shows the sheet music of the song generated using the maximum output method method. The generation loop for this song was stuck in an infinite loop We then stopped it manually and printed the song it had generated thus far. We witnessed it was repeating the same three keys | B2B B2B | over and over again.

The results from this part show that sampling from the softmax distribution instead of taking the maximum output greatly improves the quality of generated songs. We can also make unique music every time using the same priming characters which is neat as well.

6. Experimentation with Vanilla RNN

In the experimentation with a vanilla RNN using tanh as the hidden layer, we keep the hyperparameters from part a), which it has the number of hidden units is 100 and Adam Optimizer with learning rate of 0.001. We experiment vanilla RNN with two different hidden layers, tanh and relu. Also, we set the model to stop when the validation losses continuously increase twice.

Figure 6.1 shows the training versus validation loss curves for the vanilla RNN using tanh as the hidden layer and **Figure 6.2** shows the training versus validation loss curves for vanilla RNN using relu as the hidden layer. Both vanilla RNN stops around 20 epochs earlier than that of our best model shown from part (a). Both also have a pretty good performance in the decrease of validation losses. However, we observe that vanilla RNN with hidden layer of relu has a more flunctuative validation losses curve than the vanilla RNN with hidden layer of tanh and our best model. In terms of the performance of generating music samples, the vanilla RNN using tanh hidden layer fails to generate music sample while the vanilla RNN using relu hidden layer generates a pretty good music sample as seen in **Figure 6.3** and **Figure 6.4**.

7. Feature Evaluation

Figure 7.1 shows the feature evaluation for the LSTM model**.** The x and y axes denote the order in which to read the characters starting at the top left corner.

Identifying which neuron to use for the evaluation was primarily a visual test to confirm which parts of the music that a specific neuron performed well on. Our specific feature evaluation shows that it doesn't fire for the header of the music, fires for the body, and fires for the end of the song.

8. Author's Contributions

All three members of the group contributed to the completion of the project. All of the three members researched how recurrent neural networks and LSTM work on a theoretical level in order to implement them. Siqi mainly worked on the implementation for the vanilla RNN. Zekria worked on experimenting with the LSTM model to optimize the performance. Garrett worked on identifying feature evaluation All members contributed to putting the project together.

9. Figures and Tables

```
<start>
X:1
T:Ductie
M:6/8
L:1/8
K:Bb
Paf?ub>it Ffoorge chau\m, wiwe MJo]
B>2 FdBy  I
D>DE E>DAh):|t gAB d3d | f2ec cEb>=A/=G |LA,39W:Ym~!d BFE AG>FG-|: "CF
frf"AYuc'a+zlin]:|[1 bg{a}E2b9|d'3)g2f|dec)ed/z/ |1\
pm.f!f/>K/G/A/B/e/f/ ac/>n(Gs=A | cdmz}uf=(4>[K)[(G/1 BAG/B) L CFdLG.::
! et8!<:A>AA(2A4)m+HF^| 5
<end>
```

Figure 2.1 The above is the ABC notation for T=2 song 1 .

Figure 2.2 The above is the sheet music for T=2 song 1 .

```
<start>
X:1
T:Ductie
M:2/8
L:1/8
K:Bb
Ne]+acg/= .a2tebn8!K    73]evf|I !fe>=Bt c<c>A | ((UAA)"B+f2f !=f'3ete! |1 M6D2]
(B2G>{Be).^ce|{Lg7)few^dfBAG.F.FMZ>g/2FF |.HzckAc (3f}Mz2u2d2c| =zB4zzz2:|f2Nabf[2g(3hblz
[K0Muz2z[e U[m zY^m7=.[2.ue3|]e<.d2=eghn3^08"E3.A3|]recBcBAB[ALBGj?Z5
<nwe>
DD|E4G/E/:|c2G/e8|FA:|g3,>qG::
M:Vey4B
eBccADG| eFrGad2||tG[L:] FE/2F/4 e.ecAg +Swozmgq+'Fing6Tx+ @GTofd>t- _twoar-)f/a/wn-
[PB/Zt/an-o!der]Z!I!tEz F>>FG/>AMu~!+!z2g/2uf!M2_d/2cf f/2=e-!iY|9fe d>d|gdPHg gFD | c+>e
A>E :|x
pz !F!N!Ot xn'vmbou-j5
b3
<end>
```

Figure 2.3 The above is the ABC notation for T=2 song 2 .

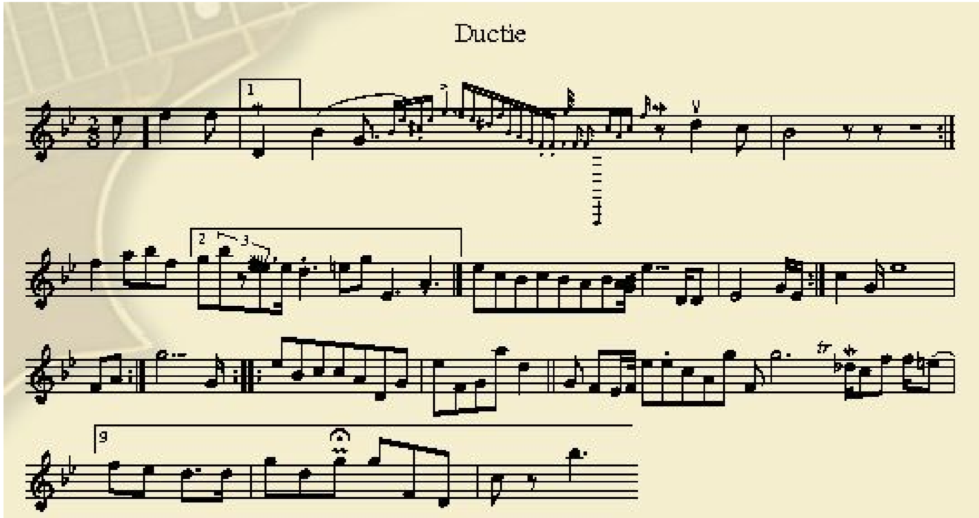


Figure 2.4 The above is the sheet music for T=2 song 2 .

```
<start>
X:1
T:Contredanse
O:France
A:Provence
R:Alour
C:Trad.
M:2/4
L:1/8
K:Em
F |: B/2B/2A/2B/2 | AB/2d/2 | eB B/c/2 |1 dd/c/ d/2c/2:|2
c/2/2dc/2 g |
d/2B/2c/2d/2 | cB/2B/2 |
dd/2f/2g/2f/2 | gd/2f/2f/2d/2 | \
P:ine
"G"e2 e/2f/2e/2d/2|
"Bb"BA/2G/2A/2B/2 ||\
"C"A3/2A/2 "F7"Ac/2A/2|1 "G7"d2 AB/2A/2| B2"D7"Bd | "Cm"c2c |
"A"d2 cd | "Am"c3z |
ce Ad | "Em"e3 "GAG |
"F"df ed | de de | "G"dBf | "d"d3 :|
<end>
```

Figure 2.5 The above is the ABC notation for T=1 song 1 .

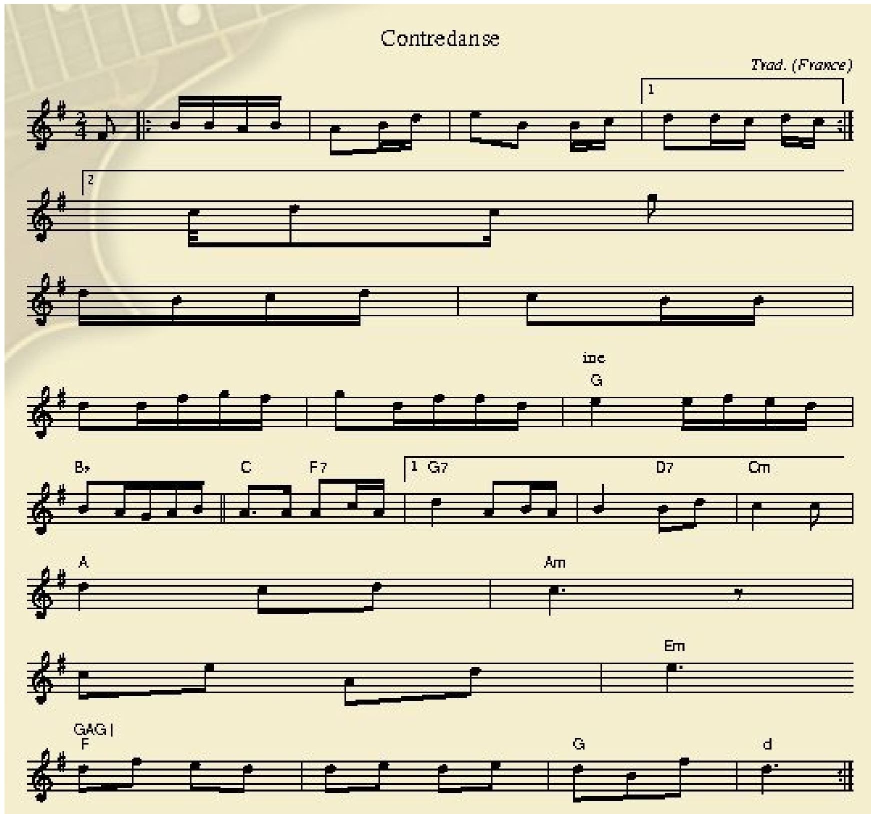


Figure 2.6 The above is the sheet music for T=1 song 1 .

```
<start>
X:1
T:Contredanse
R:mazurka
D:Paur Holland
C:Juch Jig, : orramh the Encoonaian Ryadyney: The Dilio Stain
H:Chorple Farte farchte\'i Degrava
N:970
Z:id:hn-barllany
Z:id:hn-Diny B
M:6/8
Q:3/8=160
K:Gm
a | | dGGB | A2GF | E2EF | D2GE | FGGB | AGAG | EFGB | dcBA | GFGF |
D3/2/2D |FB)dc/A/ | B2GG | BGFG | C2D ||
P:A] | c/c/BA G>A | fffg | fdB | AGAG |
T:Revbet desso aiom, E4 The
W:Toumbley tradai/men
A2c2c2|f2d2c2|B3cd2| f3dcd | e2A2A2 |]
<end>
```

Figure 2.7 The above is the ABC notation for T=1 song 2 .

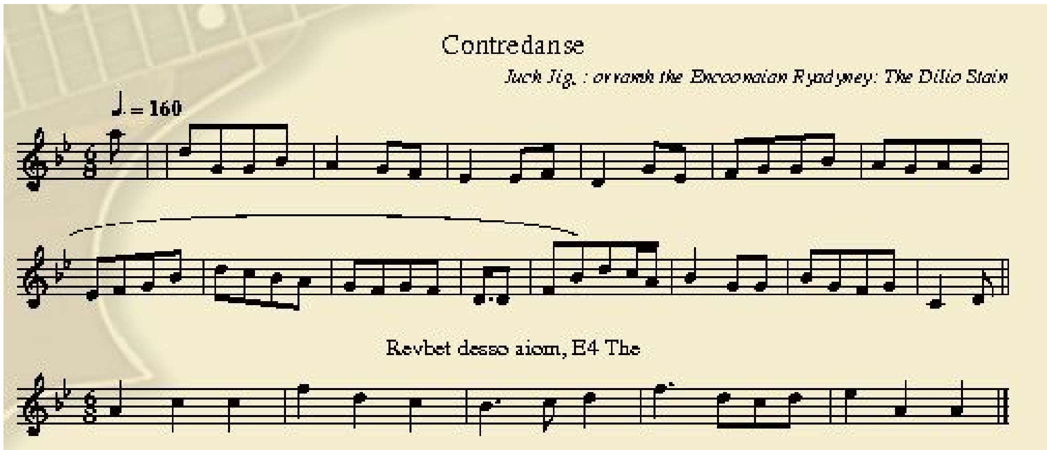


Figure 2.8 The above is the sheet music for T=1 song 2 .

```
<start>
X:1
T:Les  versoo
T:Tropanc Stain Roin
T:Sin Dannan's Jig
R:jig
D:Molly Garghan: Crath Stain Sligo
Z:id:hn-hornpipe-50
M:C|
K:G
gf||
gfgf edBd|edcB ABdc|BAGA BcdB|cAGF G2:|
<end>
```

Figure 2.9 The above is the ABC notation for T=0.5 song 1 .

Figure 2.10 The above is the sheet music for T=0.5 song 1 .

```
<start>
X:1
T:Les  maille
R:Schottische
Z:id:hn-hornpipe-70
M:C|
K:G
GG|(3Bcd ed BAG2|BddB GBAG|FAdc A2:|
|:dB|ABAG E2EF|G2G2 G2:|
<end>
```

Figure 2.9 The above is the ABC notation for T=0.5 song 2 .

Figure 2.10 The above is the sheet music for T=0.5 song 2 .

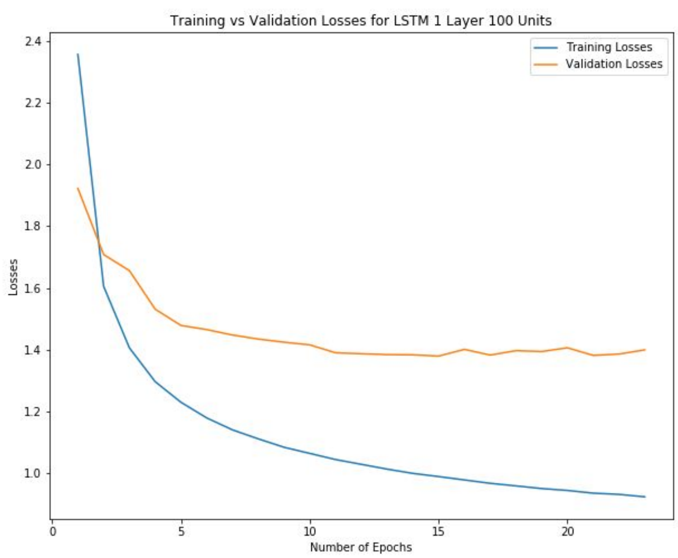


Figure 3.1 The above is the training and validation losses of our best model used in parts A & B.

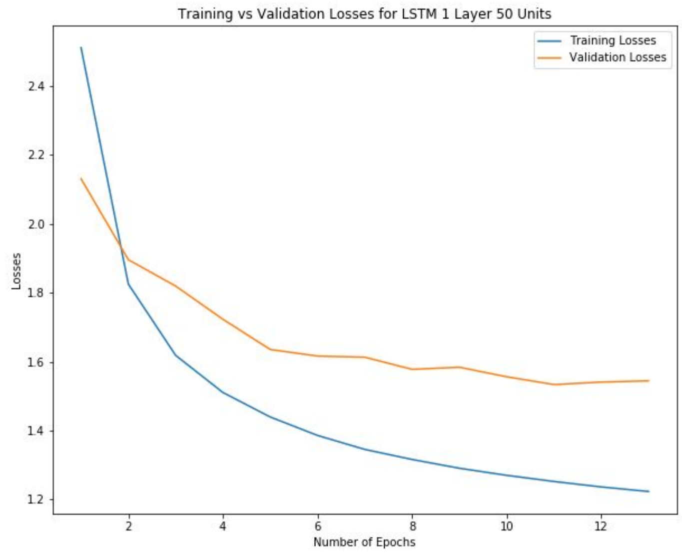


Figure 4.1 The above figure is the LSTM model with 50 number of hidden units.

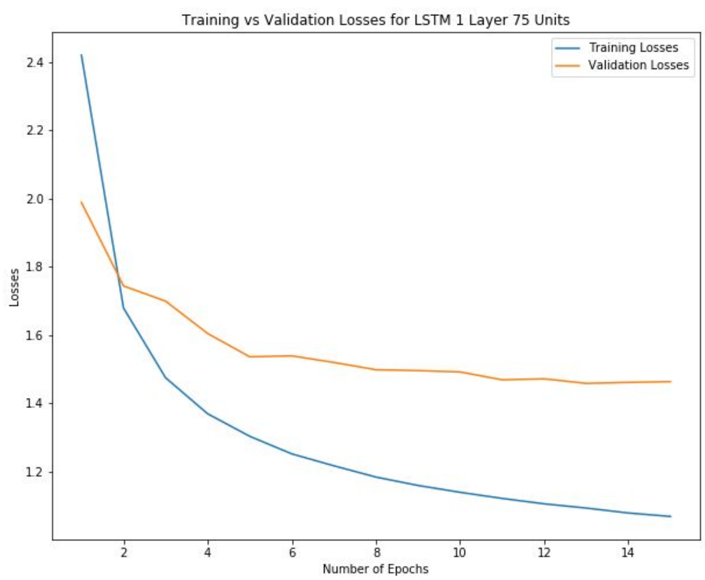


Figure 4.2 The above figure is the LSTM model with 75 number of hidden units.

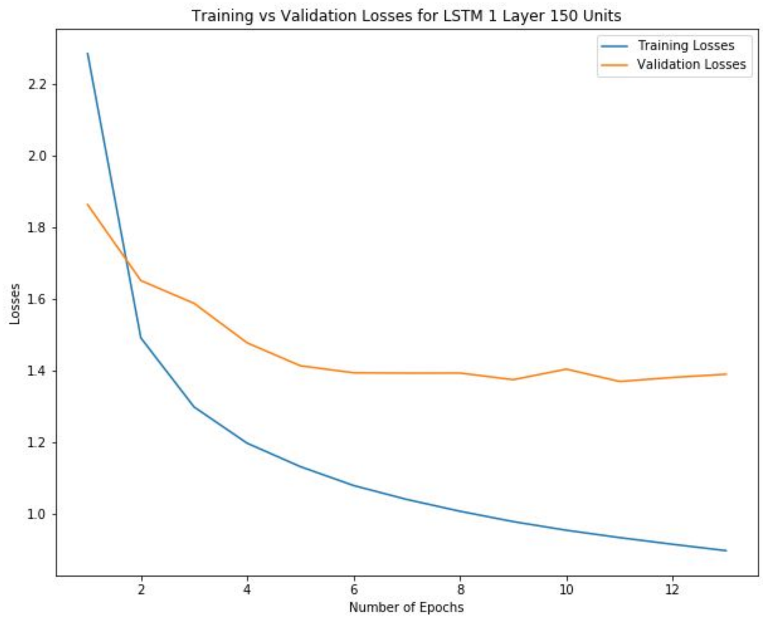


Figure 4.3 The above figure is the LSTM model with 150 number of hidden units.

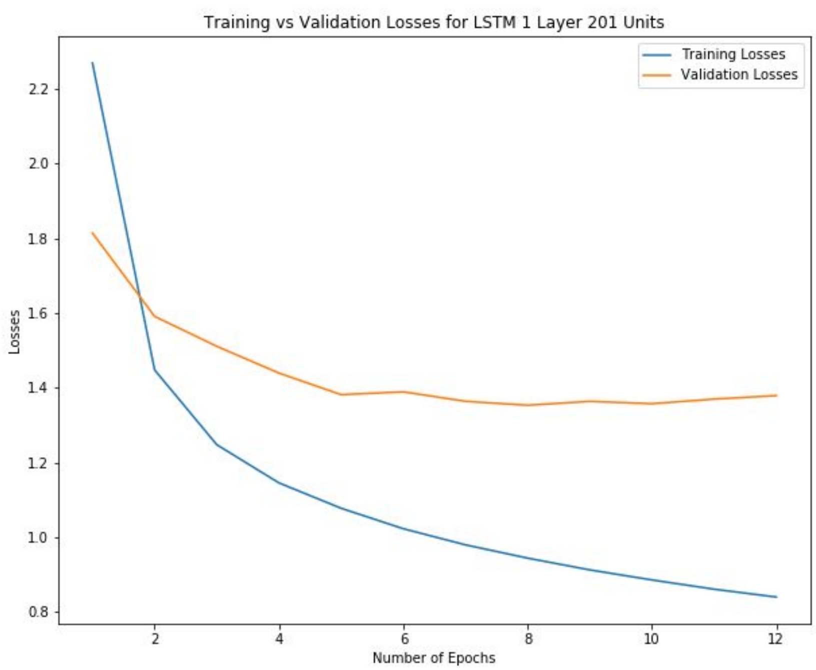


Figure 4.4 The above figure is the LSTM model with 201 number of hidden units.

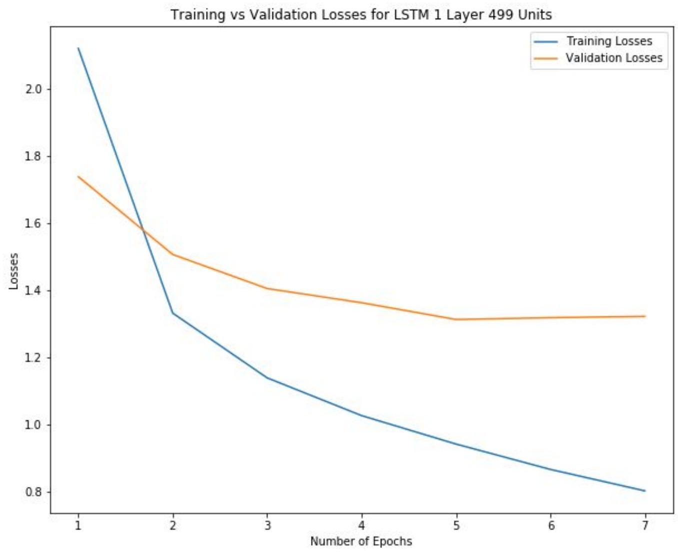


Figure 4.5 The above figure is the LSTM model with 499 number of hidden units.

Figure 5.1 The above is the sheet music of the sample song generated by using T=0.7

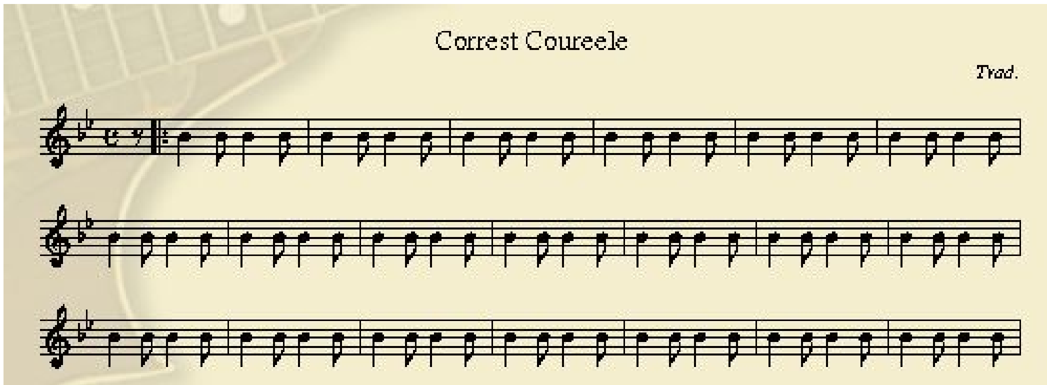


Figure 5.2 The above is the sheet music of the sample song generated using max outpu

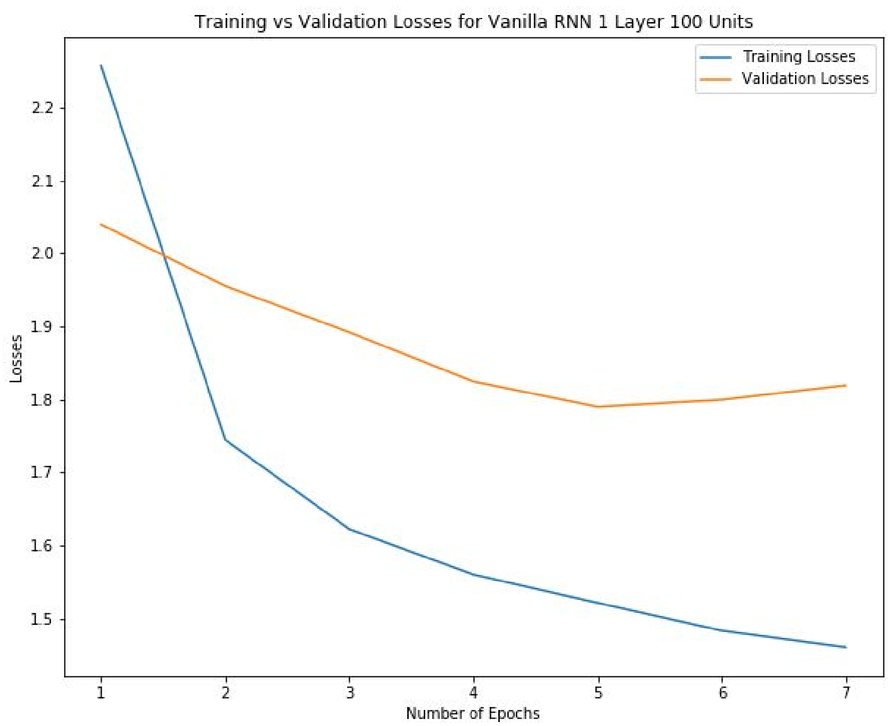
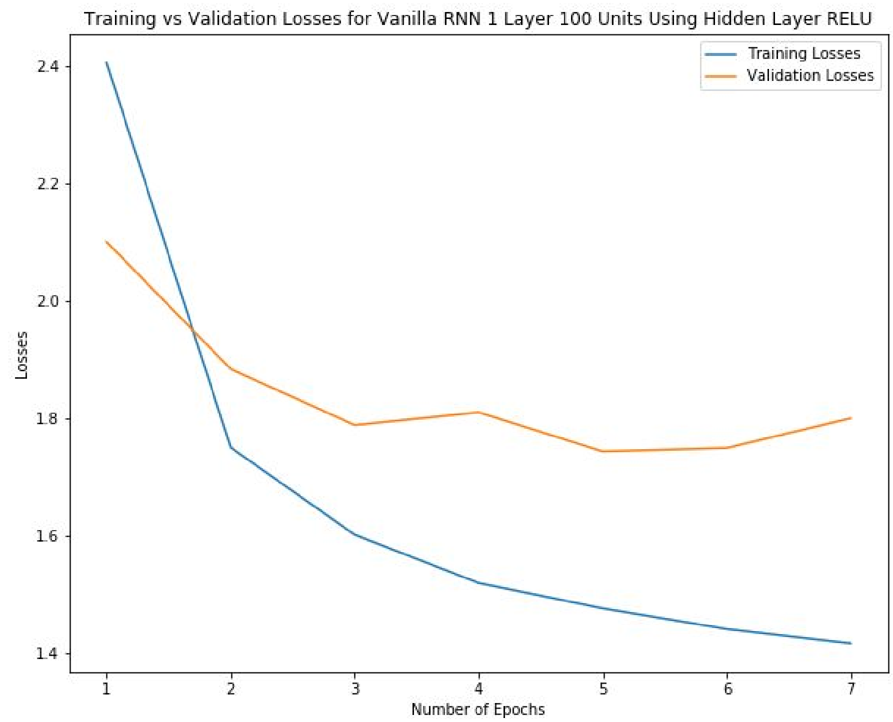


Figure 6.1 The above is the training and validation losses of vanilla RNN using tanh as the hidden layer.




```
<start>
X:1
T:Contredanse
O:France
A:Provence
C:Trad.
R:Ball Aloutine Grin Guran er Magur
C:Trad.
A:S.
Z:id:hn-1738)
Z:id:hn-portp Horeet
R:Ballan Bers bar
M:C|
K:D
V:1 Gortan
R:Ball de doubs ? lonnes
R:arts the parsto
Z:id:hn-pornpipe
R:Bolle
C:Japle
Z:id:hn-192 Brot et/ou corrig? par Michel BELLON - 2005-07-15
M:C|
K:G
DA G2 | B3 | d>d dc | df/2d/2 f/2/3/2 c/e/) f/2/) fa | g2 | e/2f/2 e/2d/2 d/2B/2 e/2A/2B/2 c/2e/2 e/2/)(d/B/A/G/ BA|B
A BA|Bd c/|B/B/) d>B | d2 de | d2 B2 | B2 c>d | d/d/) (3B/B/B/) - | (d/2//2 d2 | fe/2d/2 | de/f/ dd|de f/2e/2c/2 e/2/
f/d/ d/B/) dB/B/ dB | d2 cB/B/ | AG/2A/2 A3|GEA | d2 d2 | f3 | d2 de | c2 | c3- | d>d ed|ed ed/f/ | g2 | c2- | B2 | c
B | BG/B/ | A2 (3ABA | BB/B/ dB/G/ BG | AB | dB/B/ | AB/d/ BA | B2BB | B2 Bd | c4 | B/d/e/2e/2 g/2e/2 e/2/3/2g/2 d/2
c/2 A2 | EG G2||
|:2et (1589)
O:France
Z:Transcrit et/ou corrig? par Michel BELLON - 2005-07-15
M:C|
K:D
B/A/ | AB/B/ AB | c4 | B4 | B3/2B/2 c/2 c/2B/2 | de/f/ | (3c/d/ d/d/ | B2 | BAG | cd | B/2c/2 A/2G/2 E3 | EF E2 | (G
G/2G/2 A4 | A2 | B2 | A2 A2 | d2 B2 | B4 | d/2/) ed | d3/2e/2 c2 dB/d/ dd | cd | d3/2d/2 B/A/G/A/ df/2 | de fe|1 cB B
A AB/d/ | d>d | Bc/B/ B/c/d/ B/A/ | d2 | dB/2B/2 c/2B/2B/2 c/2/2)
<end>
```

Figure 6.3 The above is the ABC notation of the music generated with vanilla RNN using relu.

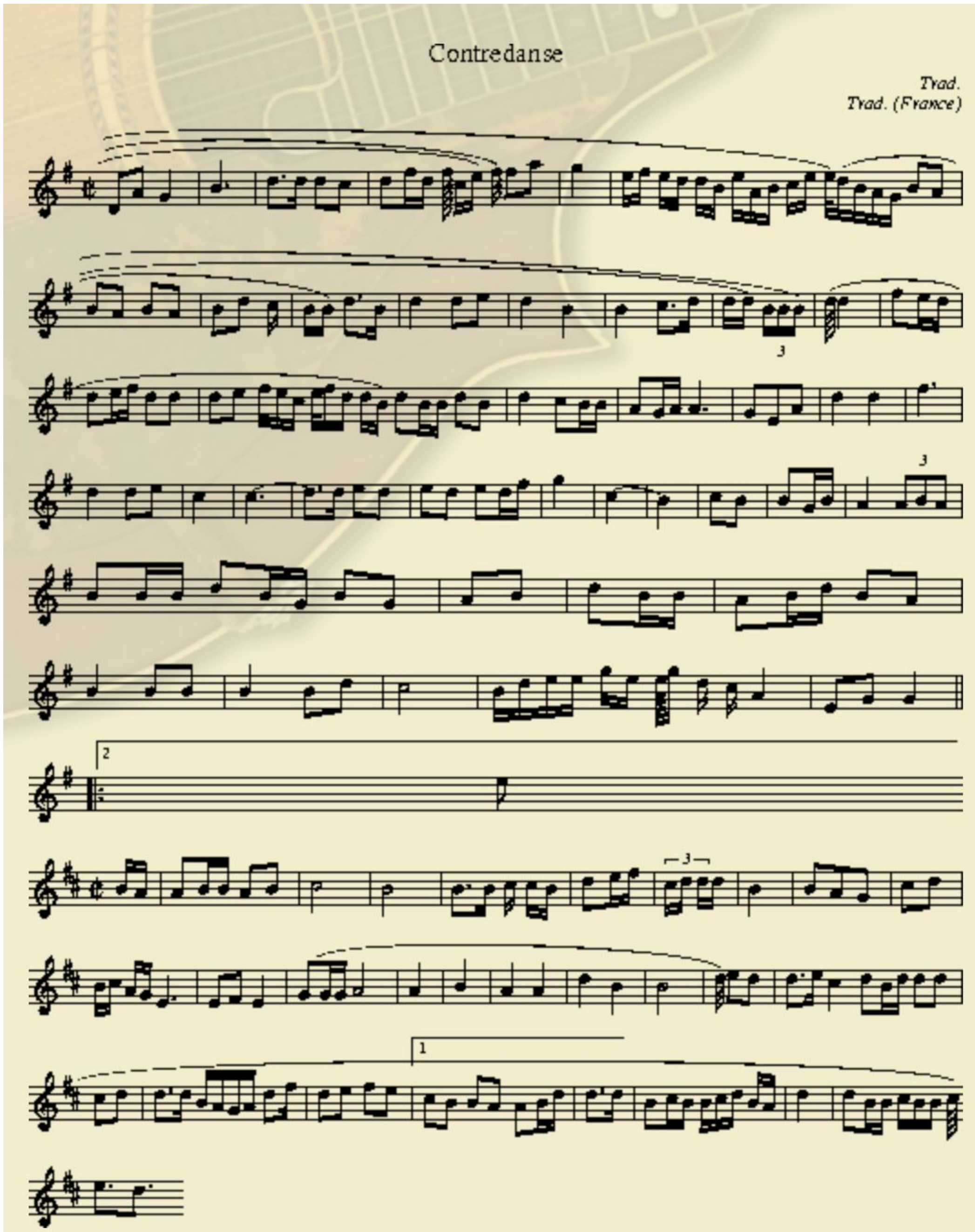


Figure 6.4 The above is the sheet music of the music generated with vanilla RNN using relu.

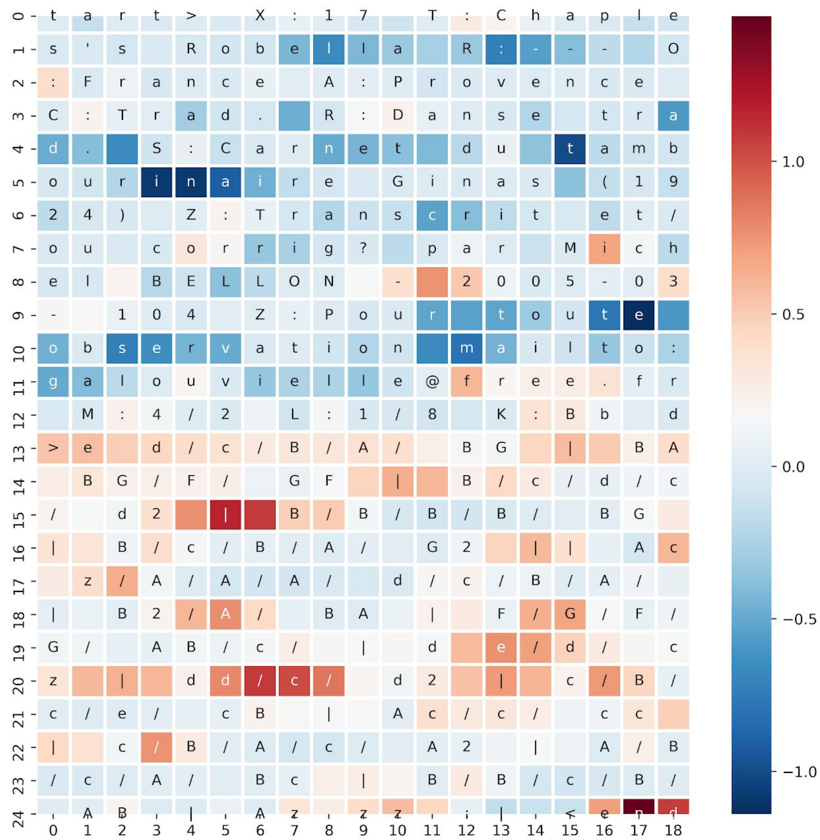


Figure 7.1 The above is the feature map for our network