Generating Music with LSTM Networks

Spencer Reitman
CSE154
UCSD
syreitma@ucsd.edu

Wonil Cho
CSE154
UCSD
woc026@ucsd.edu

Abstract

Recurrent neural networks are a well known way of modeling sequential data, and generating music is a great candidate to demonstrate this. In this paper, we explore various models of RNNs and how their various hyper-parameters can change their overall output. The main type of RNN we experimented with was an LSTM, or long short-term memory. This model outperformed a standard RNN with music generation, and it also was able to create music that was pleasant to the ear by setting the hyper-parameters correctly.

1 Introduction

Our main goal was to get an RNN, specifically an LSTM, to generate decently sounding music with training from midi files. The model itself was relatively easy to create, as the only layers in PyTorch are the LSTM layer and the linear, fully-connected layer. Once the model is set up, the hyper-parameters become the game-changers that can make or break a model. The main hyper-parameters we played around with were the number of hidden units, the learning rate, the chunk size, and the number of LSTM layers. Each change to these parameters affected both the overall loss of the validation/train sets and the ability for the model to effectively generate music.

2 Methods

Our baseline RNN model uses LSTM units with a tanh activation function, a single layer of 100 hidden units, using the Adam optimizer with a learning rate of 0.0001. The last layer is a softmax activation because we want to predict a probability distribution for every class, and as a result the objective function we optimize for is cross entropy loss. During training we backpropagate through 100 timesteps (chunk size), and used early stopping to halt training if the validation loss went up two times in a row.

We also experimented with different numbers of hidden units (50, 150, 200) in our LSTM layer (with all other hyperparameters remaining the same) and a vanilla RNN model (also using the same hyperparameters) as comparison to our original LSTM-based model.

When generating sequences with the trained model, we experimented with both random generation based on the softmax distribution and generation based on the maximum softmax output. In both cases, we passed the chosen output and the previous hidden layer activations back into the model to generate the next output.

3 Results

Training the baseline LSTM model resulted in a training for 27 epochs before it was stopped early. The training loss finished at 1.39 and the validation loss at 1.73, though we saved and used the model

from 2 epochs prior when the validation loss was still at 1.72. The negative log likelihood losses on the train, validation, and test set on that model was 1.40, 1.72, and 1.65, respectively.

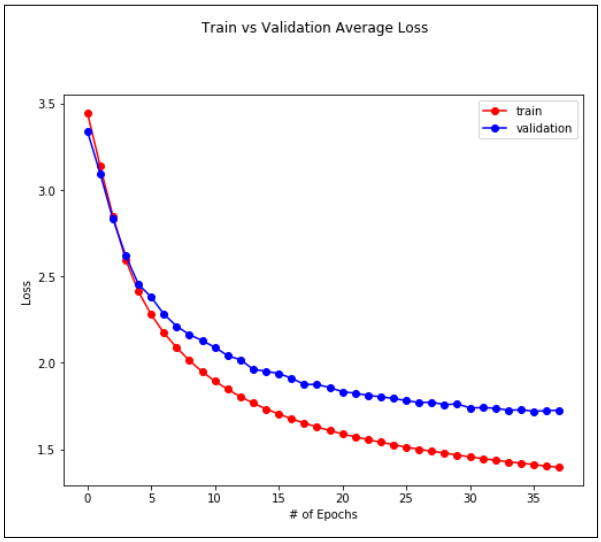

Figure 1: Baseline LSTM model with 100 hidden nodes training and validation loss over each epoch

The below sequences were generated by temperature based random sampling based on the model's softmax output, primed with only the start token. Playable music was generated with only the start token passed in at temperature values 0.5 and 1, but had significant trouble producing anything playable when temperature=2, unless it was provided a sizable portion of a file header to prime it first.

```
X:21
T:Carles O'Caroland Barn Stail
D:Jig Halle do mantiis An Stigand Botte de las ob la briglon maned #18
Z:id:hn-jig-16
M:6/8
K:A
ded dBd|BBA BAF|GEF E3:|
|: BAB A2A|GAF FAF|GAF DEF|FEE EFE|BAB ABA|B2G A2B||
```

```
X:11
T:Sona Fronhe
R:jig
C:Set Rogannonl" 2dol (16-
Z:Transstit et/ou corrivar piclhelae -elo7-020-09-178-49
K:Crtag.te Machier "Cahionde brvik
C:Planouse
f:arncy:
R:hougne
Q:D3l+or
C:Ladshosin Magtudor jie
H:Orio scoland ~ar dw ea gareN ob1y,  | canse durmobron?seam's as Mard deve@Houbhess ant ommieriren 15
W:Ch tte sess bantie  \'arnde jig Meashonpial ol adoye@bearsion de Med. cers wunthe "bim Citro llandomon.tisso #1057
Z:Transcrit et/ou Boric? par Michel BELLN:N -2007-15-26
Z:Pour toute bobservation mailto:galouvielle@free.fr
M:N/4
L:1/8
K:G
B2B2 | B2B2c2 | c2d2c2 | B2c2BB | A2)B2 |
c4 a2!C2 | A2b2e | dg f2 f2 | c2 ed _e2 | e2f e2df |
d2 d2d2e | d2 f2dc | d2ae f2e2 |
(3Bgf d4 AG ||
codc de ed dd | b2fg b4 | (A2) (fea) (gea) | b2f}ged |(3B)z (=f2) | d B3A | G4 |: FBFA | AdBBf | d2B2 :|
| T2e fge | cedc B2FG | c4 G2 | B2B B2cd | G4 A2D2 |
c3 (4z2e2 | d4c2 F2A2 | A4 | (4)B4]2 |  Bc4d2 | E6 | G4 |
|: |
```

Figure 2: Two example generated sequences at T=1

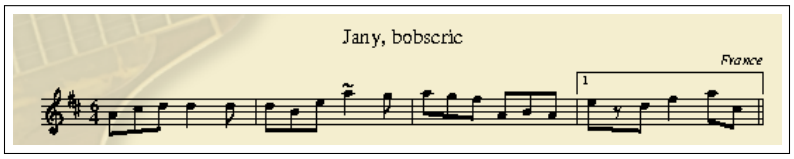

Figure 3: First T=1 example music representation

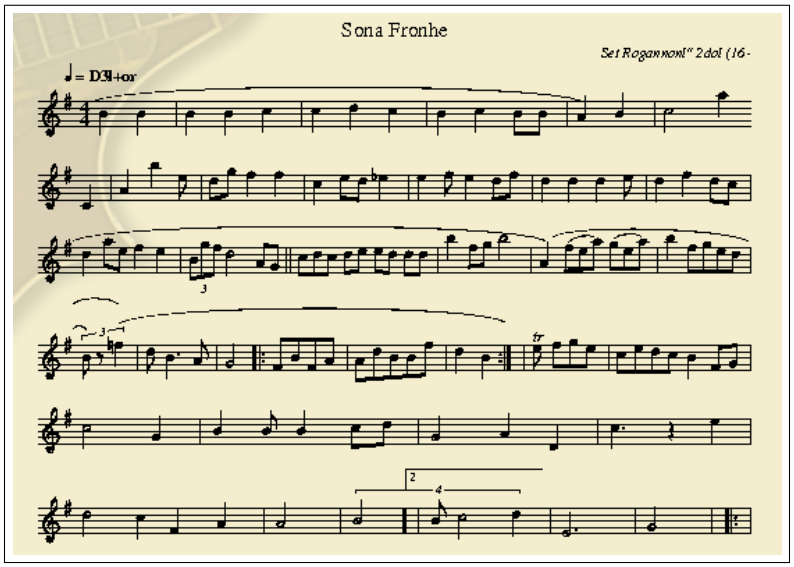

Figure 4: Second T=1 example music representation

```
X:1
T:Bour boi borke doure bowas Mapleted te the mancine de tha douris on Cha the Suporte de balte du le San so manterd ta douretins on mess an
crees the the de ou the lo bun nast ans Mart 1 160
Z:id:hn-hornpipe-11
M:C|
K:G
AG|A3A cedc|AAGB A2Bc|1 AGED E2 :|2 d2G2 A2B2||
```

```
X:21
T:Carles O'Caroland Barn Staíl
D:Jig Halle do mantiis An Stigand Botte de las ob la briglon maned #18
Z:id:hn-jig-16
M:6/8
K:A
ded dBd|BBA BAF|GEF E3:|
|: BAB A2A|GAF FAF|GAF DEF|FEE EFE|BAB ABA|B2G A2B||
```

Figure 5: Two example generated sequences at T=0.5

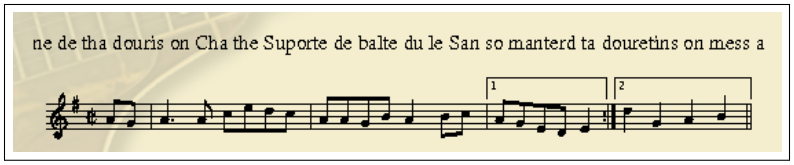

Figure 6: First T=0.5 example music representation

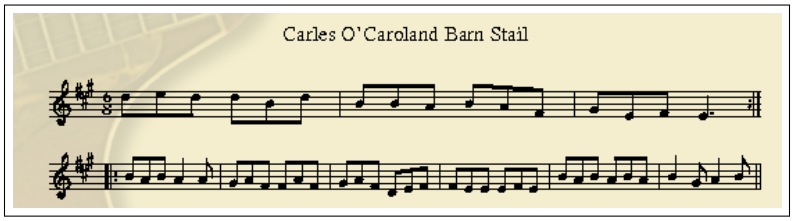

Figure 7: Second T=0.5 example music representation

```
X:11
T:Sona Fronhe
R:jig
C:Set Rogannonl" 2dol (16-
Z:Transstit et/ou corrivar piclhelae -elo7-020-09-178-49
K:Crtag.te Machier "Cahionde brvik
C:Planouse
f:arncy:
R:hougne
Q:D3l+or
C:Ladshosin Magtudor jie
H:Orio scoland ~ar dw ea gareN ob1y,  | canse durmobron?seam's as Mard deve@Houbhess ant ommieriren 15
W:Ch tte sess bantie  \'arnde jig Meashonpial ol adoye@bearsion de Med. cers wunthe "bim Citro llandomon.tisso #1057
Z:Transcrit et/ou Boric? par Michel BELLN:N -2007-15-26
Z:Pour toute bobservation mailto:galouvielle@free.fr
M:N/4
L:1/8
K:G
B2B2 | B2B2c2 | c2d2c2 | B2c2BB | A2)B2 |
c4 a2!C2 | A2b2e | dg f2 f2 | c2 ed _e2 | e2f e2df |
d2 d2d2e | d2 f2dc | d2ae f2e2 |
(3Bgf d4 AG ||
codc de ed dd | b2fg b4 | (A2) (fea) (gea) | b2f}ged |(3B)z (=f2) | d B3A | G4 |: FBFA | AdBBf | d2B2 :|
| T2e fge | cedc B2FG | c4 G2 | B2B B2cd | G4 A2D2 |
c3 (4z2e2 | d4c2 F2A2 | A4 | (4)B4]2 |  Bc4d2 | E6 | G4 |
|: |
```

```
X: ' WSeirkeixF.
O(Bcpa0503
M:2.8=B M6D
~O32Dc3 | E2B2d>McA8|D6e:|
[4H]|Enafc|Frfd^g,2:\
Sdme3jzlivaraUl (f2}dBt !tg}az tzz]F.G"d4r:
F2D1 GBc,8"C"^Ad#2{w,#2)FFz[G>}m"AFcd/ A2dezcA_A2:0F2[]:G>A DecC d|d4fz:|'Crrc]\j4O:EuAc|\
Vibg-h^gld^(fen|[
.bagf!m-f2 }2\d2gw)f "^f.)^dfz|"FmS|c
Ef,BE FD,GF |(K5Br}f}"ce""fddeB{^vTfhsBA{f}"c:z3v"G_MF]A6"|"R4:>d/!D C "2CG|AE"z3:"cz_gE m"c3/#^4e4]]:]^F,RheoSt!{ gP"(7cudet ^tne.=}0.e
S>!+eYC+free-! b2g {lap CgdB|P<BAA>ewAAB-|2F4:12=n_ecG?| ffedc|][eef2:|
```

Figure 8: Two example generated sequences at T=2

```
<start>
X:19
T:Dusty Miller, The
R:hop jig
D:Tommy Keane: The Piper's Apronan08nJ2x
S:1340
z:Ftly&
CBMar/ZO =dcrct/2.1
T:Idy#C47ywWz8y1
Zige:hn-Darak?ch
>:"_um LezKsilsAr<dwmaB 1avy=nlt>ex.rz1 8-G7 M/gy200
H:Mgmai)lloy
AU-5sScclaecens Heen?so ga7ss #?8
K:Govo \gnirs:bolo8
MN 8 1/=35
~1152d2 d2GBB| [B GcG)|=\{c4+cqeatam1x2 !dkpf a 0"g7drec "
 DC"FGFm1 Gf~]2
B>B7 A2 E^AFLBzB|e2d/2ec2 BA3B)|(3BccA|cc4( A D>B2FA|A BA/e/a/cF \6|
VF]f|]u"fcBd elf[Mi2|]c4A3|
c4d A|H(I{ijH_B'j)]iTf!7[Ac4A|B3f,en:||
S:mio)](16.G..!G6 FA A. | DFFA {cdfg |Gdcc|A2f3|aaE,C|aw|
P:B3m.Ge^cB|https://www.overleaf.com/project/5ddbc2f07a2b8700016ca1ba
p   =z>e2: g3ad:^[f17>n.v"jKe+ dmz^F^[2_d3-c|
Mf2+e2 [M::GurdouwcG FGn|?r
"k/es3!D2S (Ri(W<h(B)"Dd)r((AK)
Ho]+7fn gl"f{AA4+[7[Nu!^RufIdr2c2):.
>bA>a [z:]]fmuoc8|BBMA3]GEA[][K:DbVx#3] f Vagba4llo_.s
=n(edg))A E |G+Arz!
^F=ccBcAGB| c2c/G GB|ABcc.E3xB|~19)g3: H:]23y2zBA |G,B<p,MGA dlef|dB3=Bc2dAB^GEIGA|AGc2e2g[5144=dBB4^A4B3|FA|] C4 ]4!PD,,M2#CC4.~0,T,CIAhol>a2bca4:z20
e]If2u?ea|fg:f2e>gg3|c8|.f/f dcgce(c4pBz|l!+bbT>z bz)V+1 ^Iif3'd2g.g|!7nAan.d2f)c7.=E2|.~==B2G  G S|\9"1e>dB _ecBA F2A0An~Cmc}+}Q4Y d2{LB7!CuES !'}=e/c3-f,c?]| df=(f~L)B)c)|!Pg7?:!f/lao^![2Awb.Fretm?grnqu.tnd.\
dHenfg!}BfGd|efe'f a2g gb|g>f7g"b7"bIgb- b'[i7 V"D7d!e3s e{e}d70AmAA_B A4BF|dfe g>edcA| 9>F}:E2zzw:|bate/d/ Bz|
ABdfff|dSceAc[GcodG(:!"tA.prW.GddG f]f&d]
:r
w<end>
```

Figure 9: Sequence generated with T=2 with larger priming sequence

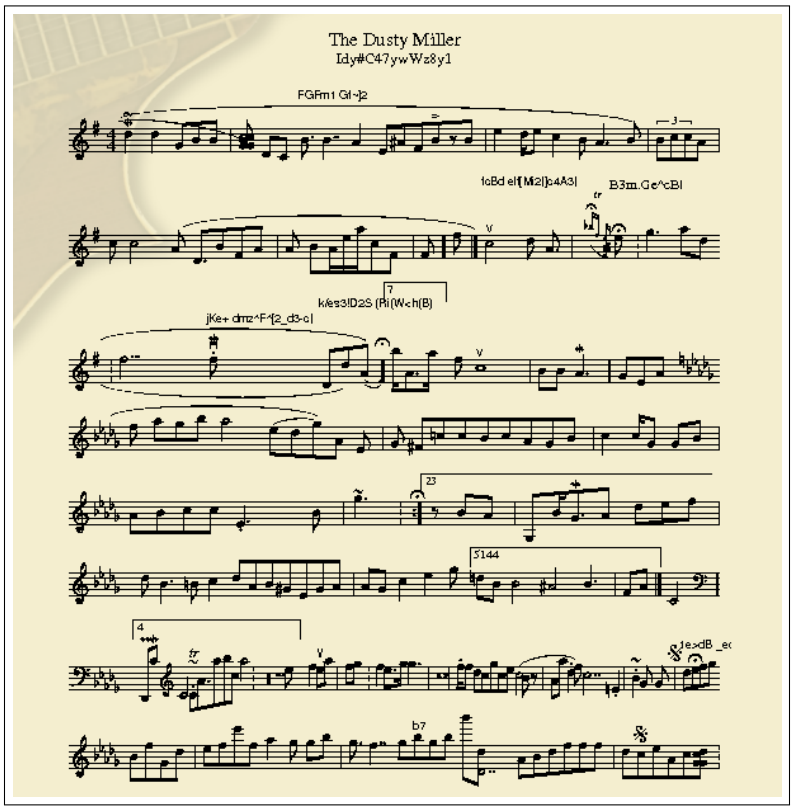

Figure 10: Music representation for T=2 with larger priming sequence

Plotting the activations of a hidden nodes at each time step upon forward propagating the first generated music sample in Figure 1 results in the heatmap shown below in Figure 11.

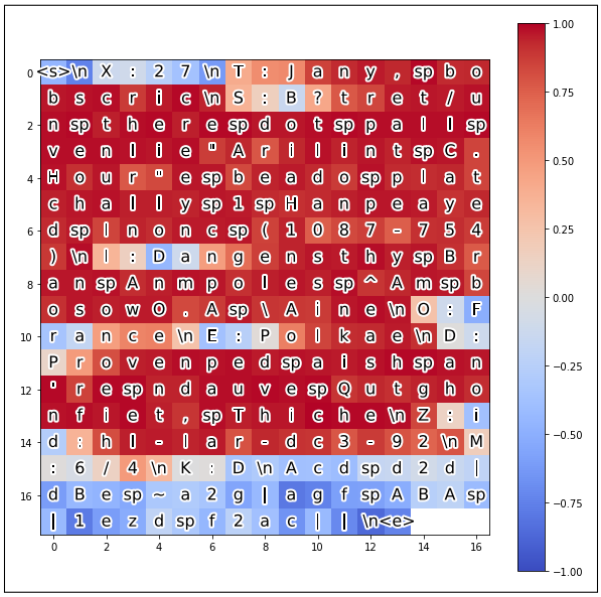

Figure 11: Heatmap of the 12th hidden node's activations; should be read from top left to bottom right

Experimentation with other numbers of hidden units resulted in the loss curves below in Figures 12-14.

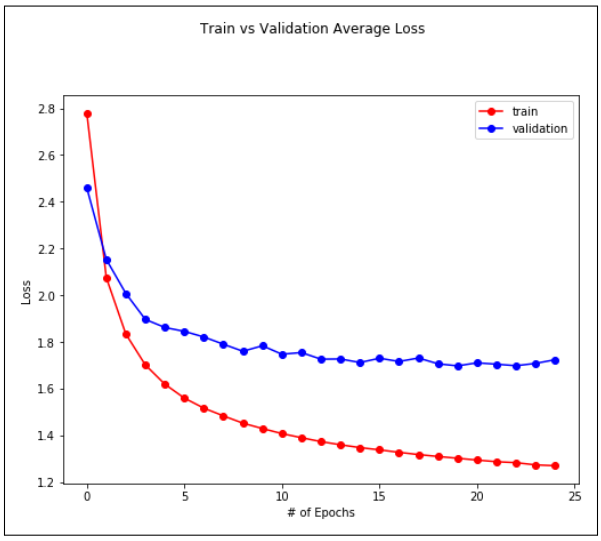

Figure 12: Baseline LSTM model with 50 hidden nodes training and validation loss over each epoch

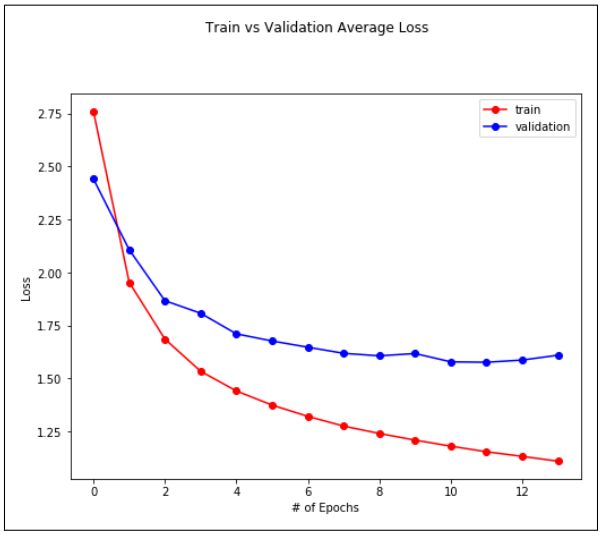

Figure 13: Baseline LSTM model with 150 hidden nodes training and validation loss over each epoch

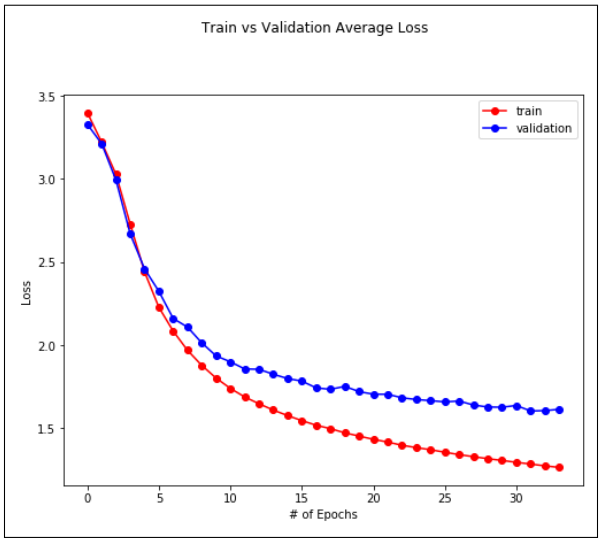

Figure 14: Baseline LSTM model with 200 hidden nodes training and validation loss over each epoch

Replacing the LSTM layer from our baseline model above with a vanilla RNN layer resulted in a 1.64 training loss and a 1.95 validation loss before being stopped at epoch 27. The best set of weights from 2 epoch prior had a validation loss of 1.94.

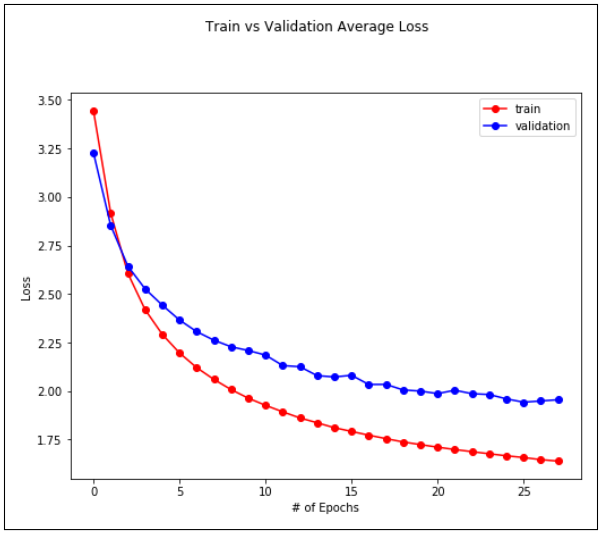

Figure 15: Vanilla RNN model training and validation loss over each epoch

4 Discussion

When generating sequences with the trained model, we found that temperature-based randomly generated sequences tend to produce more musical outputs compared to taking the maximum softmax output at each time step, which tends to start repeating character sequences near the tail end (as seen in Figure 17 when the sequence starts to repeat 'AF'). This is perhaps a result of the model's softmax output favoring the most common sub-sequence of characters in the body of the music file which ends up getting selected as the maximum in the generation at every time step.

```
X:155
T:Rourie bag bes The
R:hornpipe
D:Jie Fancond corerin: Caritie loman frinon Flase .curye 1598 (19)
Z:id:hn-jig-7
M:6/8
K:C
AGF FAd|cdc AGA|GAG FAG|fed BAF|AFA Bdef|ded ecd|e2cd d2e|
d2c Bdf|d2B AFG|ABG A2F:|2 AGG G3||
```

Figure 16: Example temperature generated sequence at T=0.7

```
X:1
T:Branle de tamboure Marche
R:hornpipe
H:Also played cart ans part on the the tambornain
Z:id:hn-jig-16
M:6/8
K:D
AG FE FA|BA AF AF|AF AF AF||
```

Figure 17: Example generated sequence when choosing maximum softmax output at each time step

When generating music using random sampling based on temperature, we found lower values of temperature, such as 0.5, 0.7, and 1, to generate playable music more consistently compared to higher values of temperature such as 2, as shown in figures 2, 5, 16, compared to figure 8. In order for T=2 to consistently generate playable music, we had to increase the length of the priming sequence, as seen in figure 9, but even the new generated music was more random than musical. We suspect this is because higher values of temperature normalizes the softmax output making the character choice at each time step more uniformly random, leading to far more errors in character choice. This is in contrast to lower values of temperature which does the opposite, "sharpening" the output and thereby promoting the choice of the most probabilistic output(s), leading to fewer mistakes. That said, lower temperature values essentially make the random sampling method closer to the maximum output method which may reduce the diversity of outputs too low to make a musical sequence. T=0.7 seems to be about just right in this regard; it avoids many of the inconsistencies that T=1 has, but still produces fairly musical output.

Tracking the activations of the hidden nodes at each time step upon forward propagating a music sample gives us an insight into what each feature each hidden node seem to be recognizing. In the case of the 12th hidden node in our baseline LSTM model, shown in Figure 11, the hidden node seems to be activating primarily for the header portion of the sample music file we provided it, specifically the T: to Z: sections, inclusive, which indicates that this node is able to recognize the header of music (from the T: to Z: section) in ABC format.

When changing the number of hidden nodes, we noticed that more hidden nodes meant lower loss, but could also mean overfitting. With more hidden nodes, we noticed that the generated music files looked a bit too "real" in that they were extremely similar to ones seen in the training data. For the hidden nodes we tested, 50 was much too low and the loss increased to 1.73 on the validation set. With 100 hidden nodes, our loss was relatively good but we knew there was room to improve. With 150 hidden nodes, the loss improved slightly and the generated music files were slightly more

consistent. Lastly, with 200 hidden nodes, the loss improved even more and the generated music files started to get much more realistic. We also tested models with more than 200 hidden nodes, but we noticed that a limit was eventually reached where loss wouldn't improve much and generated music files did not seem to improve either; thus 200 hidden nodes became a good spot right before overfitting.

The vanilla RNN model with its significantly higher loss, as shown in Figure 15, tended to be far more inconsistent in producing playable music compared to the baseline LSTM, generating sequences that were closer to what the LSTM model generated when T=2. Additionally, when generating using the maximum output, our trained RNN would generate a sequence that was unable to produce an end flag, causing it to loop indefinitely.

We experimented further with multiple LSTM layers and with dropout but didn't notice a significant difference in the loss nor the quality of the generated music.

5 Individual Contributions

Spencer Reitman's contributions to the project was in writing the dataloader, RNN model/trainer, and training/validation loss graphing code.

Wonil Cho's contributions to the project was in writing the music generation and heatmap code.