CSE 151B/251B: Deep Learning

Programming Assignment 3

Winter 2024

Instructions

Due Tuesday, February 20^{th}, 2024 11:59 pm:

1. **Start early!** If you have any questions or uncertainties about the assignment instructions, please ask about them as soon as possible (preferably on Piazza, so everyone can benefit). We want to minimize any possible confusion about this assignment and have tried very hard to make it understandable and easy for you to follow.
2. Please hand in your assignment via Gradescope. We prefer a report written using LaTeXin NeurIPS format for each assignment.
3. You should submit your code on Gradescope along with your report. For your own purposes, keep your code clean with explanatory comments, as you may want to reuse it in the future!
4. You will be using PyTorch for this assignment. You should already know how to access the UCSD GPU server by this point but if you've forgotten, please see the material on Piazza to refresh your memory. Additionally, any updates or modifications to the assignment will be announced on Piazza.
5. Please work in teams of size 4-5 (no more than 5). In extraordinary circumstances we will allow you to do it smaller groups (no fewer than 3). Please discuss your circumstances with your TA, who will then present your case to me.
6. Please use software that allows you to code together (remotely), using pair programming (or trio or quad or quint programming!). The teammate who is driving should be switched every hour or so you *all* get a chance to code. Just writing the report is not enough, or just being a code tester is not enough! We want you all to have a good learning experience coding this!
7. **Important**: For the group report, include an informative title, author list, and an abstract. The abstract should summarize the problem, what you did to approach it, your initial results, and how you improved on those to achieve the best results. The report should be well organized with an introduction, background (if you review previous work), methods, results, and discussion for each programming part. Figures should be near where they are referenced, there should be informative captions on figures, clearly specified axes and figure keys, etc. A details of what to include in the the report along with rubric can be found at the end of the document.
8. **Note: Training this model will take you about 20-30 minutes**, and may take longer if the GPUs on Datahub aren't free for usage. Any hyperparameter or architecture change that you make will take an hour before you figure out whether that change was correct or incorrect. **So, start early!**

Learning Objectives

1. Understand the basics of a recurrent neural network (LSTM).
2. You will train a basic RNN/LSTM model using characters extracted from a music dataset provided to you and then run the network in generative mode to "compose" music.

Part I

Homework problems to be solved individually, and turned in individually

1. How do LSTMs deal with vanishing and exploding gradients? (2 points)
2. This question refers to Figure 1. Give an example of a domain or problem where the following types of RNN architectures could be used: (2 points each)
   - Figure 1b: many to one.
   - Figure 1c: one to many
   - Figure 1d: many in, then many out.
   - Figure 1e: Simultaneous many in, many out.

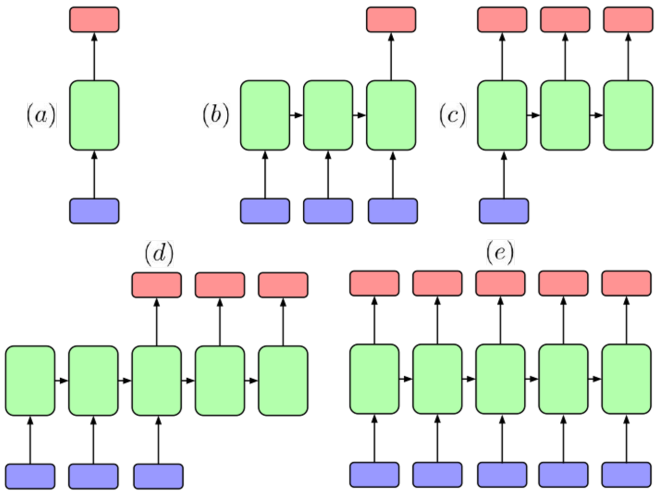

Figure 1: Basic Recurrent Network Architectures (except (a), which is feedforward!)

3. What is the function of each of the gates in an LSTM cell? (2 points)
4. **For CSE251B Students only, NO EXTRA CREDIT FOR CSE151B:** Please do Chapter 1, Problem 1.2 from Bishop Textbook (For reference, it is on page 44 of the PDF) (5 pts)

Character level LSTM for music generation (40 points)

In this assignment we will explore the power of Recurrent Neural Networks. In this problem, we will generate music in abc format. You will train a basic RNN/LSTM model using characters extracted from a music dataset provided to you and then run the network in generative mode to "compose" music. Please familiarize yourself with Pytorch before implementing. One of the good places to start with Pytorch is this playlist.

GitHub Classroom

The link to our GitHub Classroom assignment is https://classroom.github.com/a/LEtGBJXO. Following that link will lead to a button that says "Accept this assignment." Accepting this assignment will set up a Git repo with the starter code. You should have admin permissions on this repo, so you can share it with your teammates. *Only one person per team should accept the assignment* and then share the repository with the whole team by adding them as collaborators. On GitHub, this can be done under Settings -> Manage Access -> Add people.

Problem:

1. **Getting familiar with the data** In this part, we are going to see how we can convert music from ABC notation to a playable format(.midi in this case) online. Go to the website https://notabc.app/abc-converter/. Copy the text from **fur-elise.txt** file (found under the **Data folder** in pa3-starter code, which contains music in ABC notation) and hit Submit. Download the tune in midi format and play it on your computer.

   You will be generating the music in a similar format as the sample file, which is ABC format. A sample ABC file is shown in Fig 2.

2. **Read in data**. Read in the data from the train.txt (found under the **Data folder** in pa3-starter code) file. This file contains multiple tunes in ABC format, with each tune delineated by <start> and <end> tags.

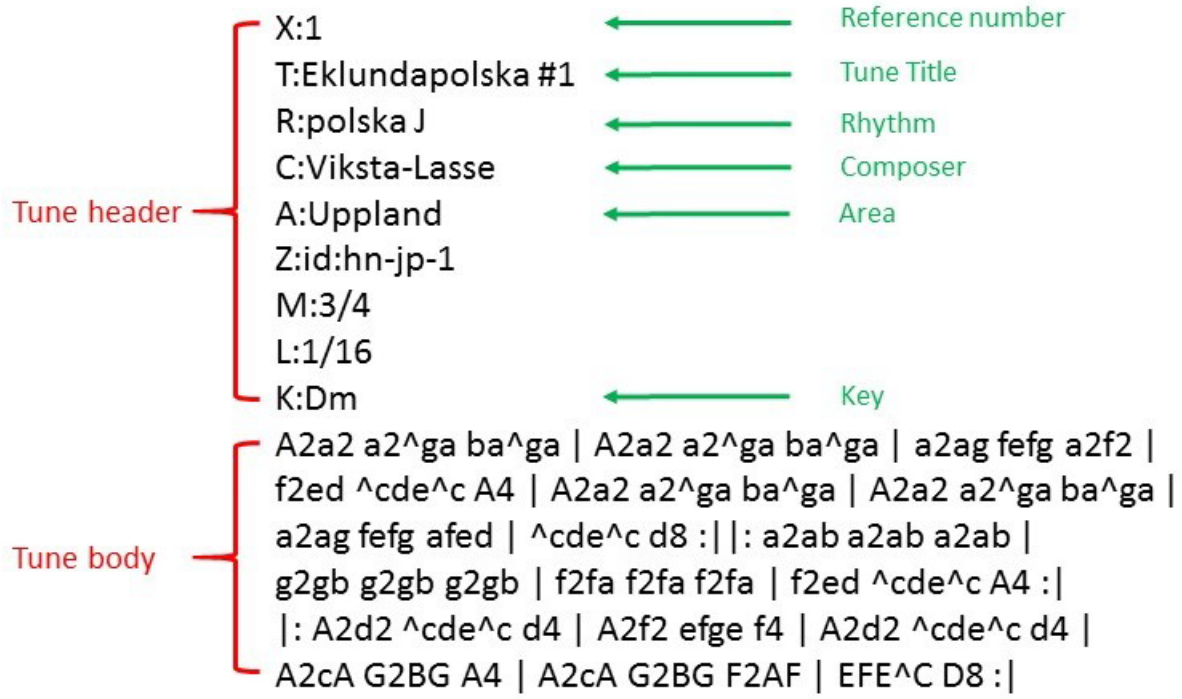

Figure 2: A music file in ABC notation.

3. **Train a network** First, train a LSTM architecture to learn the structure of an ABC notation music file through prediction. Your network will take in a slice of the training set - just, say, a batch of 25-30 characters, and given the first character as input, you will train it to predict the second character, etc. So, you are slicing *random* sequences from the training set - not aligned with the beginning of the file.

   Train one hidden layer LSTM based architecture. You should use the available Pytorch modules to build the LSTM. Try using around 150 neurons in your hidden layer. You should use the cross entropy loss.

   In the training stage, the network takes the ground-truth character of current step as input and predicts the next character. Then the next character (ground truth) is used as input, and it is trained to produce the subsequent character. This is sometimes called "teacher forcing." Teacher forcing works by using the teaching signal from the training dataset at the current time step, $target(t)$, as input in the next time step $x(t+1) = target(t)$, rather than the output $y(t)$ generated by the network. The training and validation dataset is provided to you. Evaluate your model on the training and validation set after every epoch. Include the training and validation loss curves in your report. A decent baseline should give you **1.9** cross-entropy loss on the validation data.

4. **Generate music** In the generation stage, the idea is to "prime" the network with a sequence, and then let the network run on its own, predicting the next character, and then using the network's output as the next input. There are at least two ways you could feed back the network output. One is to take the maximum output. We don't recommend this option. A second way is to flip an n-sided coin, assuming n outputs, based on the probability distribution at the softmax layer. You can make the network more or less deterministic by adjusting the temperature (T) parameter in the softmax, but start with T=1.

   Save the sequence to a file, and then use this to convert back to midi format, and play what is generated using https://notabc.app/abc-converter/.

   For your report: Generate 3 sample music pieces, one at $T = 1$, one at $T = 2$, and one at $T = 0.5$. They should be of reasonable length. Pick ones that sound the best to you for your report. Provide their ABC notation and music representation generated from https://notabc.app/abc-converter/ . Discuss your results. Report all your hyperparameters. Also upload the tunes generated in midi format and their ABC notations in .txt format on Gradescope. A sample output would look like the example shown in Figure 3. Figure 4 shows the music in standard musical notation.

```
X:44
T:Farscrisue FB cF2 d2Az|B3d fd :|
w: Laka Gan vout B'a
M:3/4
L:1/8
K:Bb
B>G | ABcB | G2cB MBABA | A4c/B/c/B/ cc | BA/G/ BB | G2B2 | cdcA | FABAA | B2z2 | B4z2 | B2A cBA | FAG:|
F2e|B3 GA BB B/G/B | G2B | A F/G/G/A/ B4
M:2/4
L:1/4
K:C
(AGA G2z2 | BBc2 c2A2|B2B2 B2c2 | d2ef (fce)f3g2e| [M:6/8]:
F2A G2B|c2c g2c a4|f2c2|cdc FGB|cBF BAA | B2F2 A2B2|c2z3 ebee|B2c2 bonastot eaut Fupteesafs2 sennc
e,>c/d/2|BA cc | c2G2|d ef/f/ | d2cBcB | B4 |]
```

Figure 3: Generated Music

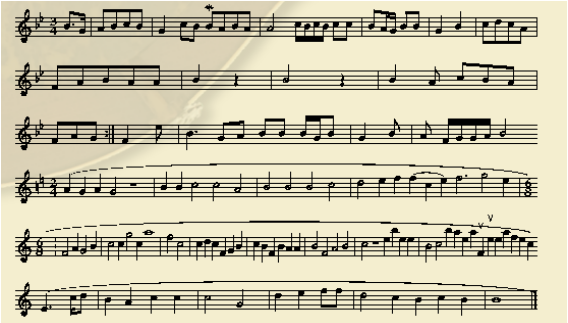

Figure 4: Music from Figure 3 in standard music notation.

5. **Hyperparameter Tuning**

   (a) Try building your architecture using RNN hidden units instead of LSTMs. Discuss your findings.

   (b) Try changing the number of neurons in your hidden layer for at least 3 different numbers, for ex. 150, 200 and 250. Now, again plot your training loss and validation loss vs number of epochs on data. What do you observe? Discuss your findings.

   (c) Use dropout with p = 0.1, 0.2, and 0.3, try generating one sample music for each. Also, plot your training loss and validation loss vs number of epochs for each. Does dropout increase or decrease the training speed? Does it improve the results? (This is a qualitative judgment). Discuss your findings.

6. **Feature Evaluation** For one of your generated music samples, do forward propagation through the network and note the activation of each neuron for each of the characters. Plot each of these activations as a heatmap and report the heatmaps for at least 3 neurons whose activation pattern you can interpret as signaling some feature of the music. Discuss your interpretation of the neuron.
   A example of one of these heatmaps is given in Figure 5. It basically shows some generated text from our trained network and shows how a neuron behaves for each of the character in that. In this case, the neuron had low activation for body of the music and high activation for header, which shows that it is able to recognize header of music in ABC format.

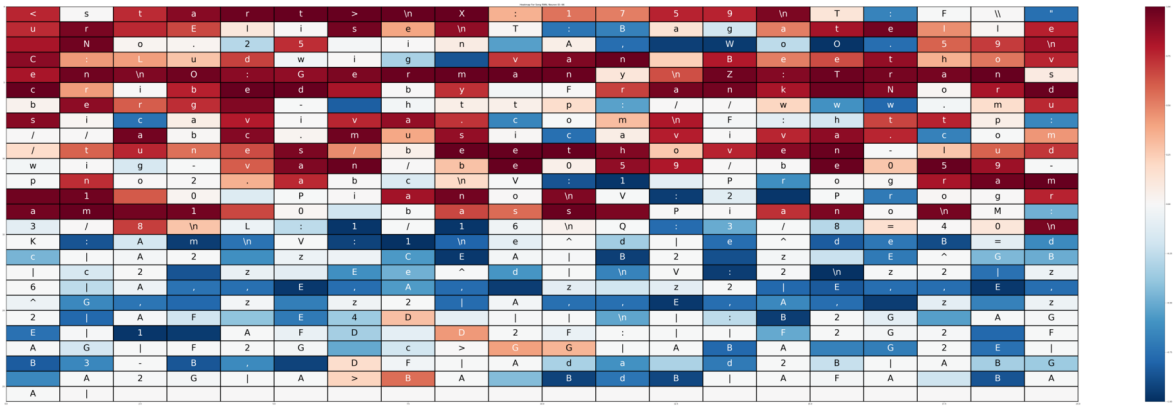

Figure 5: Heatmap showing that this particular neuron fires for the header.

What to include in your report

In addition to learning very useful and applicable skills in deep learning, this portion of the assignment will help you learn to write a scientific report. Please include an abstract and the following *7 sections*:

1. **Title:** (1 pt) A meaningful title for your report.

2. **Abstract:** (1 pt) The Abstract should serve as $\sim$ one paragraph synopsis of the work in your report, including the task (e.g. Character level LSTM for music generation), how you approached it, and a quick overview of your results. Please mention any key findings and interesting insights.

3. **Introduction:** (2 pt)
   The Introduction should describe the problem statement or task, why it's important, and any necessary background your audience may need to know.

4. **Related Work:** (2 pt)
   The Related Work section should review any work you used to inspire your approach - this includes previous research in the specific problem you address, as well as any core ideas you build upon. You should include citations in standard reference format with this itemized in a References section at the end of the report. This is where using LaTeX and BibTex can come in very handy.

5. **Methods:** (10 pt)
   In the Methods section, you should describe the implementation and architectural details of your system - in particular, this addresses how you approached the problem and the design of your solution. For those who believe in reproducible science, this should be fine-grained enough such that somebody could implement your model/algorithm and reproduce the results you claim to achieve.

   (a) **Training network using Teacher forcing (3 pt):** You should describe the LSTM architecture, the training procedure, the loss criterion, the optimizer you used.

   (b) **Song Generation (2 pt):** You should describe the approach you took to generate a song by priming the network with an initial sequence given to it. You should also discuss the details of incorporating Temperature (T) in generating the output.

   (c) **Hyper-parameter Tuning (2 pt):** Describe the RNN architecture you used in 5.a and briefly describe the approach you took in tuning your hyperparameters.

   (d) **Feature Evaluation (3 pt):** Describe why it is important to do feature evaluation. Describe the approach you took in generating the Heatmap of the activation's of each neurons for each of the characters.

6. **Results** (14 pt)
   In the Results section, you should demonstrate your baseline model's performance, performances for each of the parts of Q3, Q4, Q5 and Q6.

   You are expected to report the following things in your submission:

   (a) Report results for your baseline model, a 1-hidden layer LSTM with 150 neurons. **(2 pt)**

   (b) Report the three generated music samples with the above-mentioned three different Temperature (T) settings. Submit their .txt and .midi files as a part of your gradescope submission. **(3 pt)**

   (c) Report loss plots for hyperparameter tuning experiments namely: RNNs vs LSTMs, changing number of neurons in the hidden layer and varying dropouts. **(6 pt)**

   (d) Report heatmaps with appropriate scales, for three different neurons from your best-trained model. **(3 pt)**

7. **Discussion** (10 pt)
   Please discuss the following important points as well: How did the qualitative performance and loss of RNN vs LSTM models differ? Discuss the performance differences with respect to changes in the number of neurons and dropout. Also, draw insights from the heatmap of each of your neurons.

8. **Authors' Contributions and References** (Points to be subtracted if sufficient contribution missing)
   Each group member must write a small paragraph describing their contributions to the project. Do not forget to cite your references! You should not include any references that aren't cited somewhere in the paper.

UCSD GPU cluster

UCSD provides access to its GPU cluster through UCSD Datahub. You are free to use any other platforms but we won't be responsible for any glitches, bugs or delays in them.
(NOTE: The steps here might be outdated as the quarter proceeds. We advise you to check Piazza and their official website for any changes in the steps.)
Steps to access UCSD datahub are as follows.

1. **Method 1**

   (a) Log into UCSD Datahub using Single Sign On method.

   (b) There will be two notebooks visible for the class. Choose the one that allocates you GPU.

   (c) Once the notebook is spawned, A JupyterHub Dashboard will appear.

2. **Method 2**

   (a) ssh into dsmlp-login.ucsd.edu using *ssh username@dsmlp-login.ucsd.edu*. Before that, you will need to connect through VPN if you are not connected to on campus network.

(b) Use *launch-scipy-ml-gpu.sh* to launch a Jupyter Notebook with GPU access.

(c) More indepth detail available here.

More details on using Datahub is provided on this link

Start early on the PA. Datahub GPUs might get over utilized towards the end of PA and you may not be able to get access to GPU when you need it. Happy Coding !! :)