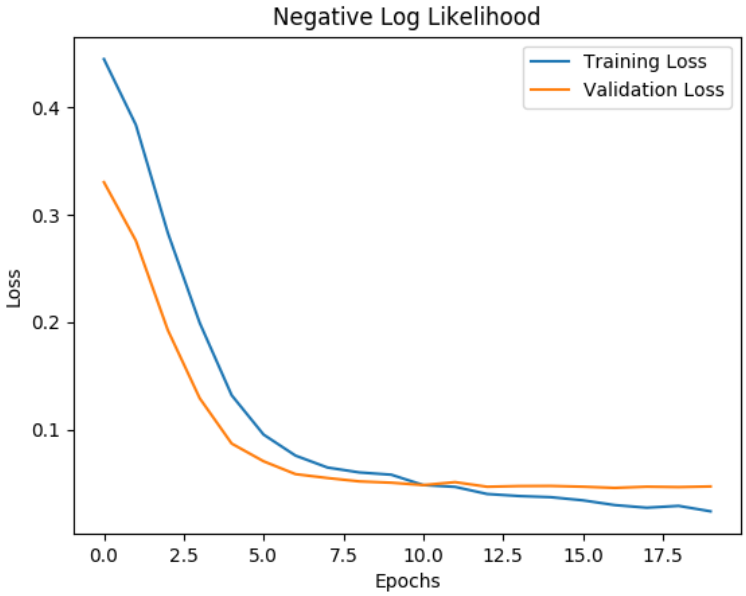
Programming Assignment 1

Shubham Choudhary, ME15B139
Dhruv Chopra, EE16B107

April 17, 2019

1. **Learning Curve**



2. **Best Hyperparameter Setting**

   - Learning Rate = 0.001
   - Batch Size = 64
   - Dropout rate = 0.75
   - Initialization = Xavier
   - Decoder Layers = 2

3. **Dimensions**

   Note: We have kept batch size 50 and encoder and max decoder lengths as 65.

- Inembed : Input(65 x 50 x 45) Output(65 x 50 x 256)
- Encoder : Input(65 x 50 x 256) Output(65 x 50 x 512) State(50 x 512)
- Outembed: Input(65 x 50) Output(65 x 50 x 256)
- Decoder : Input(65 x 50 x 1280) Output(65 x 50 x 256)
- Softmax : Input(65 x 50 x 256) Output(65 x 50 x 87)

4. **Unidirectional vs Bidirectional Encoder**
   On using a unidirectional encoder the accuracies reduced by about 2 percent. This was expected because a letter depends upon not only the letters before it but also the letters after it and this reflects in the encoding of the word, which is worse off in the case of unidirectional LSTM encoder.

5. **Attention Mechanism**

   - The current target hidden state is compared with all source states to derive attention weights
   - Based on the attention weights we compute a context vector as the weighted average of the source states.
   - Combine the context vector with the current target hidden state to yield the final attention vector
   - The attention vector is fed as an input to the next time step (input feeding). The first three steps can be summarized by the equations below:

$$\alpha_{ts} = \frac{exp(score(h_t, \bar{h_s}))}{\sum_{i=1}^{S} exp(score(h_t, \bar{h_i}))}$$

$$c_t = \sum_{j=1}^{S} \alpha_{ts} h_j$$

$$a_t = f(c_t, h_t) = tanh()W_c[c_t; h_t]$$

   Here, the function score is used to compared the target hidden state h_t with each of the source hidden states $\bar{h}_s$, and the result is normalized to produced attention weights (a distribution over source positions). Finally,

$$score(h_t, \bar{h_s}) = v_a^T tanh(W_1 h_t + W_2 \bar{h_s})$$

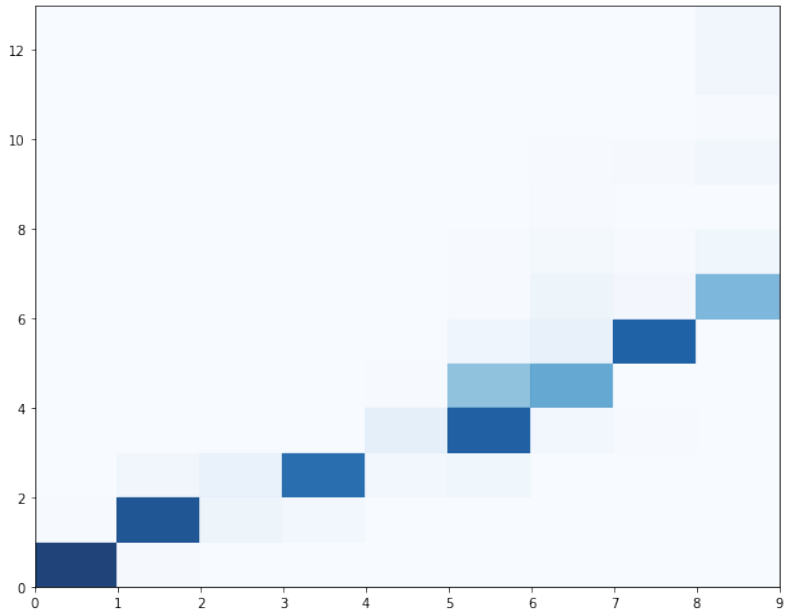
6. **Early Stopping Criterion**
   No, the early stopping criterion is not optimal because after a certain point although loss increases, the accuracy also increases. This is so because it predicts correct outputs with more confidence and it does so with the incorrect outputs as well. This increases the validation loss.

7. **Using valid accuracy as Stopping Criterion**
   This gives a better final accuracy, because we observed that many times the loss starts increasing(the network starts overfitting) but still accuracy improves upto a certain point before it starts reducing. Hence, using validation accuracy as a stopping criterion is a better option, as it gives better final results.

8. **Hyperparameter Tuning**
   Please refer to the tables below.

9. **Effect of using Attention Mechanism**
   Using attention mechanism boosted the accuracy by 4 to 5 percent. This increase was expected because attention is crucial to transliteration as character transliteration is a local attention task and not a distributed one.

10. **Visualization Plot**



   Most of the later Hindi characters have many-one mapping with the English characters.The first Hindi character has strong one-one mapping with the first English character. Non-contiguous mapping is seen for the 3rd and 5th hindi character.

11. **Single layer compared to two layered decoder**
   A two layered decoder gave an accuracy improvement of 1.5 to 2 percent over a single layered decoder. This is because, more the number of layers, more the number of hidden units and parameters and hence more the learning capacity and hence better the performance. However, using dropout for a two layered decoder was necessary to prevent overfitting(because more the number of parameters also means more overfitting).

12. **The effect of using dropout**
    We can observe from the hyperparameter tuning table using dropout with a keep probability of 0.75 gave us the best accuracy. This is because dropout prevents the network from overfitting to the training data.

13. **Beam Search**
    Beam Search performs better than greedy decoding giving roughly 2 percent increase in accuracies. Increasing the beam width increases its performance slightly upto a from widths 3-10 and rapidly from widths 1-3.

learning rate	0.1	0.01	0.001	0.0001
valid accuracy	0.4012	29.29	31.795	7.82
valid loss	0.2245	0.07	0.0743	0.1191

Table 1: Tuning learning rate

dropout keep prob	0.25	0.5	0.75	1
valid accuracy	35.908	36.209	36.6	35.5
valid loss	0.0843	0.0899	0.0877	0.0858

Table 2: Tuning dropout keep prob

initilization	Xavier	He
valid accuracy	36.2	35.607
valid loss	0.093	0.086

Table 3: Tuning initialization

batch size	32	64	128	256	512
valid accuracy	34.604	36.2	34.9	31.795	28.485
valid loss	0.0832	0.093	0.0812	0.0743	0.0689

Table 4: Tuning batch size