Report for Programming Assignment-3

April 17, 2019

Checklist :

☑ checked.

☐ unchecked.

☒ not done.

The task you need to ensure before submission.

- ☑ We have read all the instruction carefully and followed them to our best ability.
- ☑ We have written the name, roll no in report.
- ☑ Run sanity_check.sh.
- ☑ We will be submitting only single submission on behalf of our team.
- ☑ We have not included unnecessary text, pages, logos in the assignment.
- ☑ We have not used any high level APIs(Keras, Estimators for e.g.).
- ☑ We have not copied anything for this assignment.

Team Mate 1 : Ojas Mehta
Roll No : CS18M038
Team Mate 2 : Avani Shukla
Roll No : CS18M052
Team Name : 101010101

1. A plot of the learning curve showing iterations on the x-axis and negative log likelihood over labels on the y-axis. Make a single plot showing both the training loss and the validation loss. [5 marks]

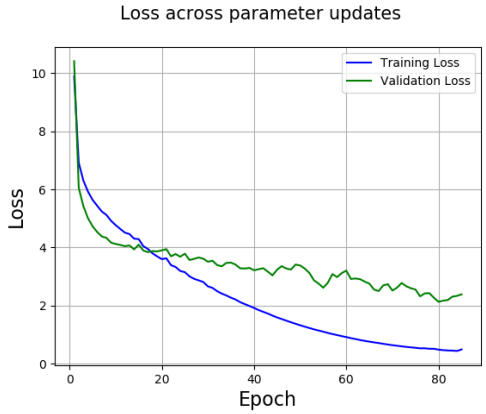

Figure 1: LSTM with Bidirectional Two-layered

2. Report the parameter setting which gave the best results and the performance on the test data of the model that performs best on the validation data. (10+15 marks from Kaggle leaderboard)

   The best accuracy that out model has got on public leaderboard is **64.333**.
   Bidirectional Encoder, Two layer decodered with LSTM Cells.
   LSTM Cells initialized using Xavier initialization.
   Bahadanau Attention using outputs of first layer of decoder.
   Decoders used without Teacher forcing.
   AdamOptimizer with 0.001 learning rate. Early Stopping critia with patience parameter 10.
   Ran till 77 epochs with batch size = 500.
   Dropout probability used was 0.

3. Report a table with validation accuracies with different hyperparameter settings. [15 marks]

   Note:
   All the accuracies are calculated by comparing whole word. So, a prediction is correct only if entire word matches otherwise not.

hyper parameter	hyper parameter value	validation accuracy
Learning Rate	0.01	32.562
	0.001	39.4182
	0.0001	34.571
Initialization	Xavier	39.4182
	Uniform Random	0.0
Batch Size	500	39.4182
	250	36.5095
	100	35.6068
Dropout probability	0.0	39.4182
	0.2	34.3029
	0.4	33.7011

4. Write down the dimensions of the input and output at each layer (for example, the input to INEMBED layer is 19 x 50 x 256) [6 marks]

NOTE:
Assuming Max sequence length of each batch(english) = 19
Assuming Max sequence length of each batch(Hindi) = 23
Hindi vocab size (out) = 72
Batch size = 500
Used **time major** in encoder and decoder both.

Layer	Input	Output
INEMBEDED	19 ×500	19 × 500 × 256
ENCODER	19 × 500 × 256	500 × 256
DECODER(two-layered)	500 × 512	23 × 500 × 512
Projection Layer(FF layer)	23 × 500 × 512	23 × 500 × 72
SoftMax Layer	23 × 500 × 72	23 × 500
Out Embedded Layer	23 × 500	23 × 500 × 512

5. Did you try using only a unidirectional LSTM for the encoder? What was the effect? [6 marks total (including implementation/code for unidirectional and bidirectional LSTM)]

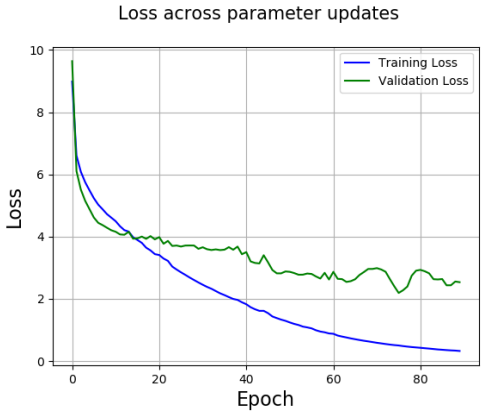

Figure 2: Unidirectional Encoder

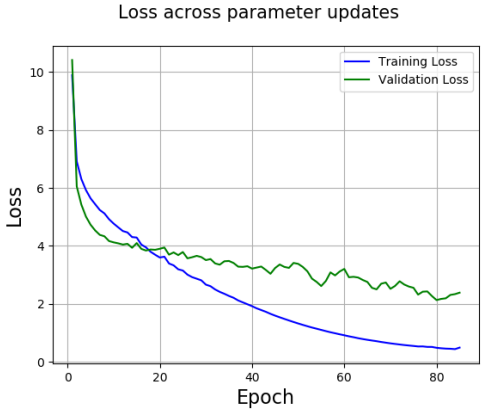

Figure 3: Bidirectional Encoder

6. Explain the attention mechanism you used with equations. [4 marks + 4 for proper implementation/code]
   We have used Badanau's Attention mechanism.The attention model is described as the following series of equations.

   - Each decoder output is conditioned on the previous outputs and some $\mathbf{x}$, where $\mathbf{x}$ consists of the current hidden state (which takes into account previous outputs) and the attention "context", which is calculated below. The function g is a fully-connected layer with a nonlinear activation, which takes as input the values y_{i-1}, s_i, and c_i concatenated.

$$p(y_i \mid \{y_1, ..., y_{i-1}\}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

- The current hidden state s_i is calculated by an RNN f with the last hidden state s_{i-1}, last decoder output value y_{i-1}, and context vector c_i.

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

- The context vector c_i is a weighted sum of all encoder outputs, where each weight a_{ij} is the amount of "attention" paid to the corresponding encoder output h_j.

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

- where each weight a_{ij} is a normalized (over all steps) attention "energy" e_{ij}

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}$$

- where each attention energy is calculated with some function a (such as another linear layer) using the last hidden state s_{i-1} and that particular encoder output h_j

$$e_{ij} = a(s_{i-1}, h_j)$$

7. What was the effect of using attention mechanism? [2 marks]
   With attention validation loss decresed till 2 while without it it only reached till 3.
   So, clearly attention improved the prediction on validation data.

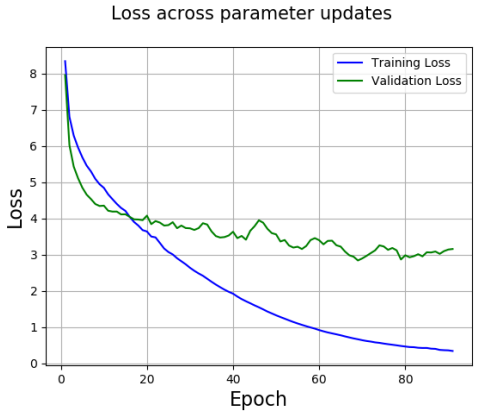

Figure 4: Without Attention

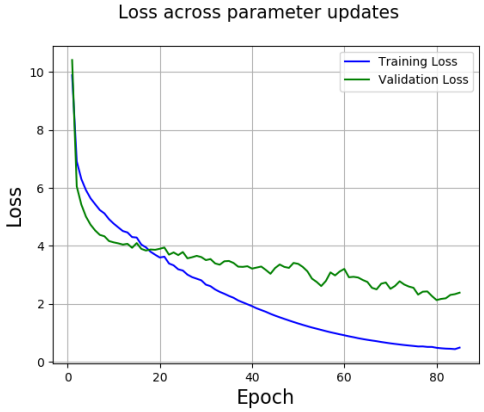

Figure 5: With attention

8. Plot a visualization of the attention layer weights for a sequence pair. Do you see meaningful character alignments? Do you see one-one, one-many, many-one alignments? Do you see non-contiguous alignments? [7 marks including correct working code]

   A matched with one character of hindi(one-one alignment).
   S matched with two characters of hindi(one-many alignment).
   LL matched with one character of hindi(many-one alignment).

   - one-one alignment

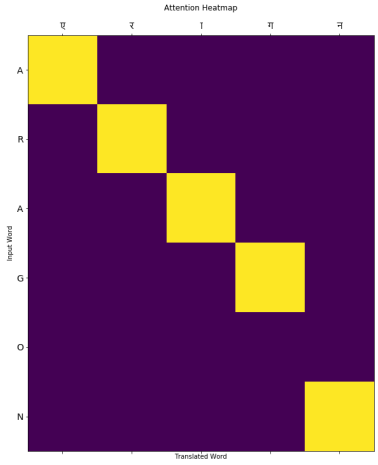

Figure 6: one-one alignment

- one-many alignment

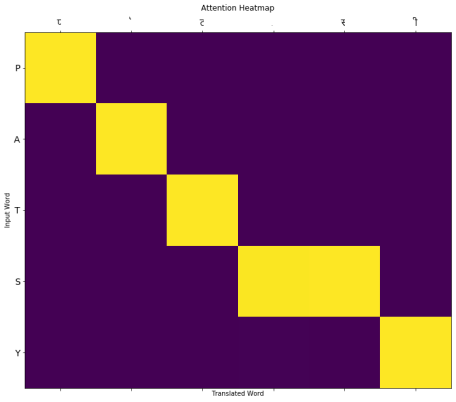

Figure 7: one-many alignment

- many-one alignment

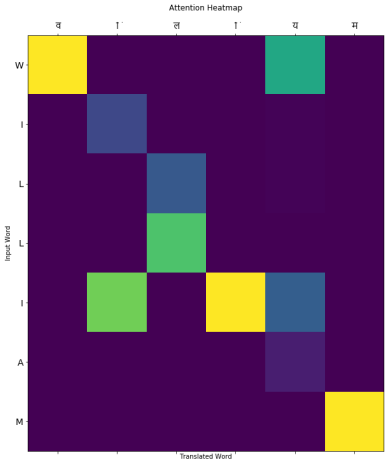

Figure 8: many-one alignment

9. What was the effect of using 2-layered decoder as compared to single decoder? [7 marks including correct code]
   There doesn't seem to be much difference between one or two layered decoder except that one layered decoder tool less time to compute.

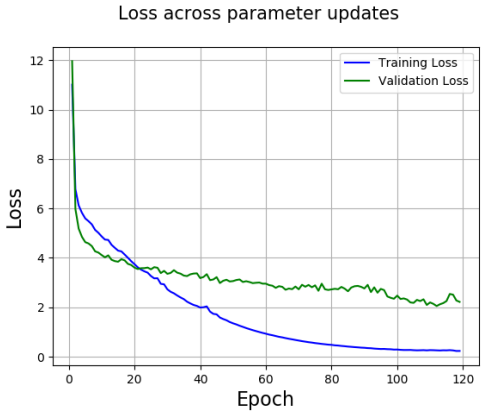

Figure 9: single layer decoder

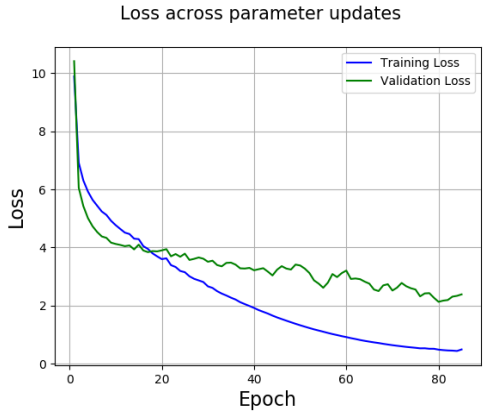

Figure 10: Two layer decoder

10. What was the effect of using dropout? [2 marks]
    Due to early stopping criteria being 5, the model with dropout convered on higher loss than the one without dropout.
    So, applying no dropout seems better.

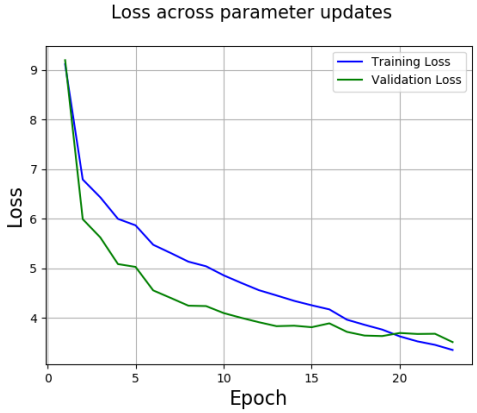

Figure 11: with dropout probability 0.4

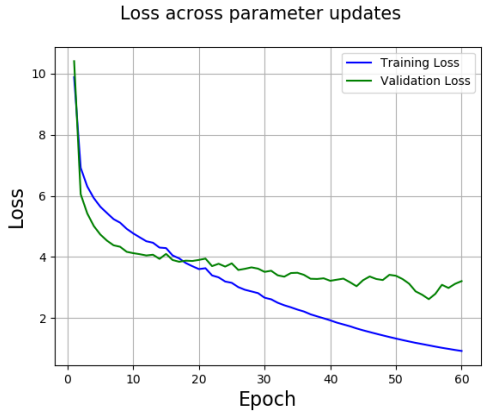

Figure 12: without dropout

11. Is the early stopping criterion optimal? Try training for a few epochs beyond the early stopping epoch. Does the validation loss reduce? [3 marks including implementation]
No the early stopping criteria of 5 is not optimal. Because with 5, model stopped at loss 3 but with criteria as 20 it went till loss 2 and gave much better accuracy.

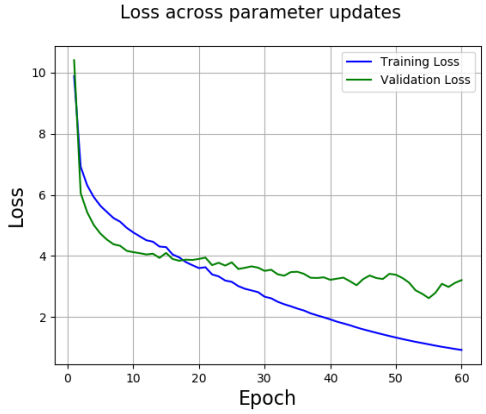

Figure 13: with early stopping criteria 5

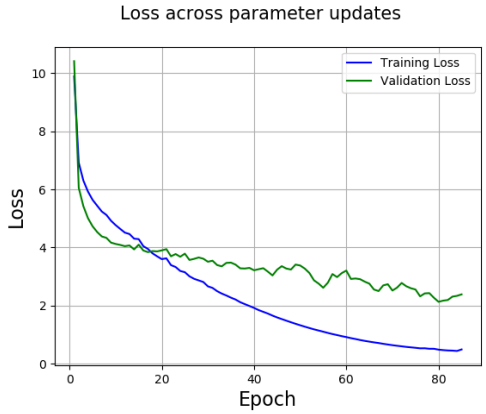

Figure 14: with early stopping criteria 20

12. Instead of using validation loss as early stopping criterion, you can also try using validation accuracy as stopping criterion. How do the two approaches compare? [2 marks]
Clearly, validation accuracy is better for early stopping from the below two figure and gave minimum loss.

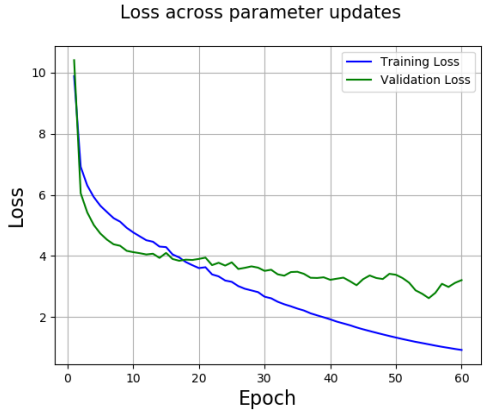

Figure 15: Validation Loss as Early stopping criteria

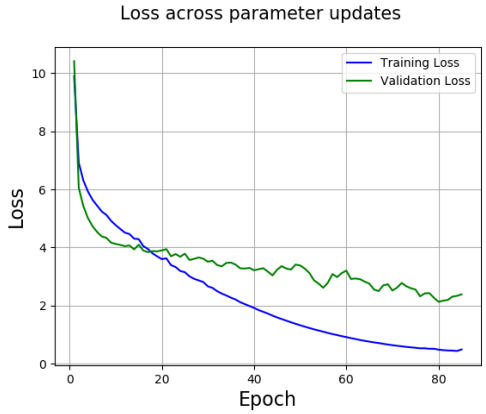

Figure 16: Validation Accuracy as Early stopping criteria

Extra credit Please elaborate any additional creative ideas you've implemented

(a) Data pre-processing Removed the symbols from English that doesn't come in Hindi .
Separated words at underscore so single word which consisted of many words got splitted as for prediction we were given only single words.
Thus increasing training data and also reducing unnecessary extension to max sequence length.

(b) Randomized Teacher Forcing We sent desired actual output as input for next time step with teacher forcing probability and fed prediction of previous time step otherwise during training.

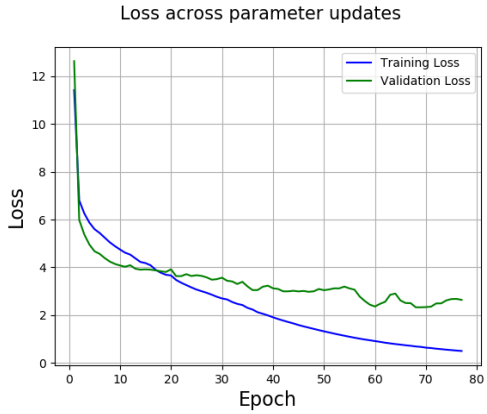

Figure 17: Randomized teacher forcing

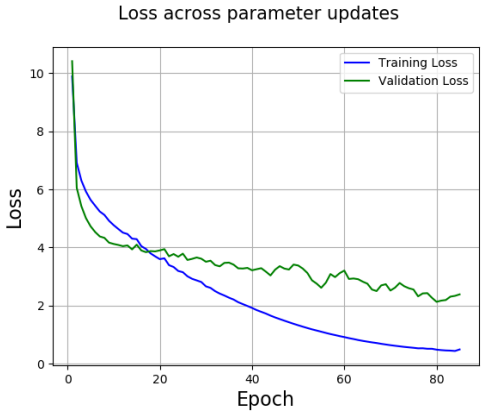

Figure 18: Without Any teacher forcing

observation Dropout was of no help
Early stopping criteria 5 was not suffiencint and had to increase the criteria for getting better accuracy.
Xavier initialization is better than uniform initialization.
Learning rate 0.001 on adam gave best accuracy.
Validation accuracy should be used for early stopping.
Bidirectional encoder is better than unidirectional encoder.