Assignment 5
RNN for Transliteration
CS7015 : Deep Learning

Siddharth Bhatia (EE16B120)
Anant Shah (EE16B105)

17 April 2019

1 Parameter Settings

The parameters settings which gave us the best result on the validation and test set are :

1. Learning Rate : $8e - 4$
2. Dropout Probability : 0.7
3. Batch Size : 12
4. Epochs : 20
5. Initialization : Xavier Initialization

This gave us an accuracy of 43.66% on the test data-set and gave us an accuracy of 42.9% on the validation data-set.

2 Comparison of Different Learning Rates

Rest of the hyper-parameters as as follows:

1. Batch Size: 12
2. Keep Probability: 0.5
3. Initialization: Xavier

The plots for the training and validation loss for different learning rates are shown in Figure 1 and Figure 2.

Training Loss: We observe that the training loss converges very slowly(but smoothly) for learning rate of 0.0001, as expected. Learning rate of 0.01 is relatively more noisier than the other two as expected, but is faster than 0.0001. Clearly, 0.0001 is too low and 0.01 is too high for this experiment, indicated by the comparison with 0.001 displaying steady decrease.

Validation Loss: Similar trends are observed as in the training loss curve, with 0.001 saturating pretty quickly.

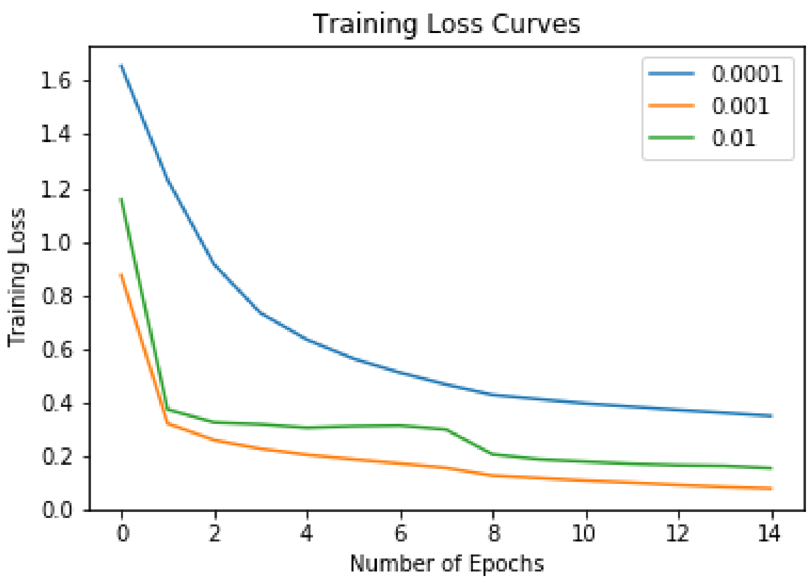

Figure 1: Training Loss for different learning rates

3 Comparison of Different Batch Sizes

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e-4$
2. Keep Probability: 0.5
3. Initialization: Xavier

The plots for the training and validation loss for different batch sizes are shown in Figure 3 and Figure 4.
Training Loss: We observe that the training loss converges faster as we decrease the batch size(in this experiment)(with minor changes between batch sizes of 30 and 10)
Validation Loss: Similar trends are observed as in the training loss curve. The validation curve of batch size 10 slowly starts increasing as number of epochs increases, indicating a case of over-fitting.

4 Comparison of Different dropout probabilities

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e-4$
2. Batch Size: 12
3. Initialization: Xavier

The plots for training and validation loss for different dropout probabilities are shown in Figure 5 and Figure 6.
Training Loss: The training loss converges faster with increasing probability. This is in line with our intuition that higher probability of keeping the weights can lead to more over-fitting, and hence lower training loss.

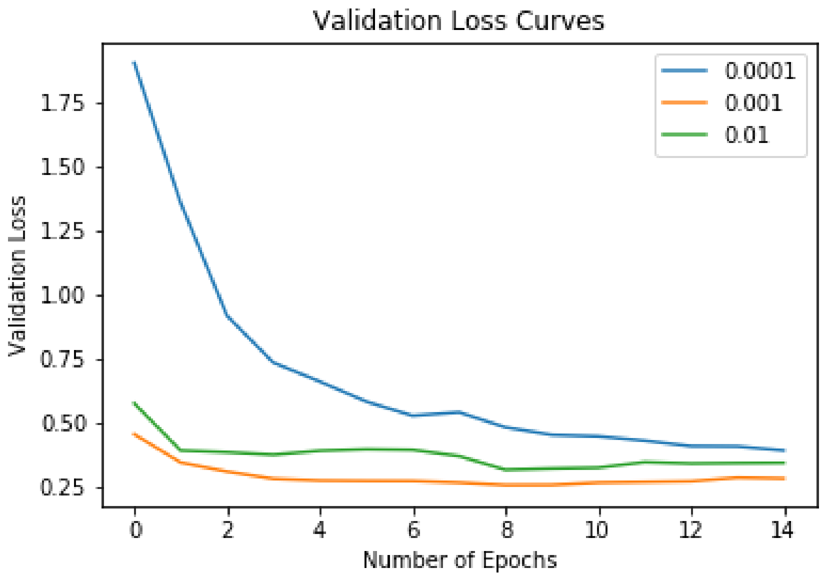

Figure 2: Validation Loss for different learning rates

Validation Loss: Probabilities 0.9 and 0.7 show clear signs of over-fitting(decreasing, and then increasing validation curves). probabilities 0.5 and 0.3 perform much better.

5 Comparison of Different Initialization

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e - 4$
2. Batch Size: 12
3. Keep Probability: 0.7

The plots for training and validation loss for different initialization techniques are shown in Figure 7 and Figure 8.

Training Loss: The training loss curves for both initialization are very close. Nothing much can be concluded from this.

Validation Loss: Again, the validation loss curves are very similar. This date is insufficient to make any conclusions.

6 Comparison with and without dropout

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e - 4$
2. Batch Size: 12
3. Initialization: Xavier

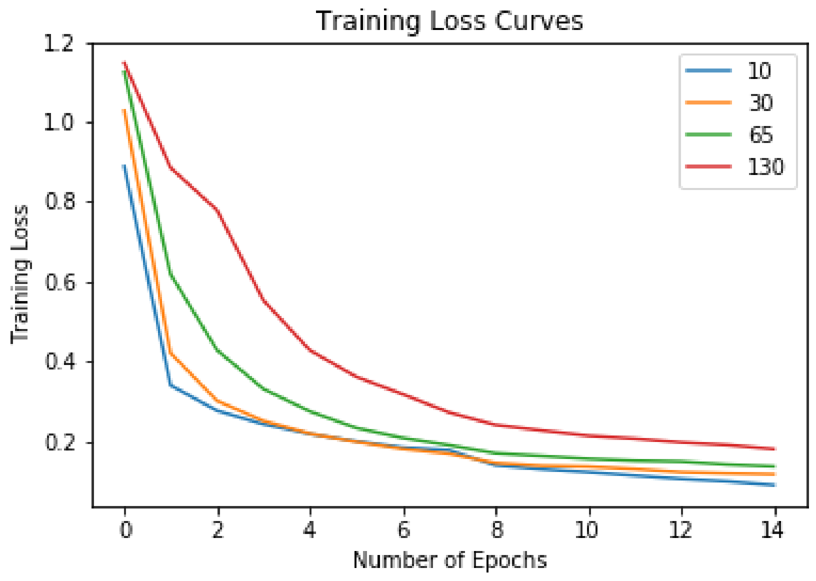

Figure 3: Training Loss for different batch sizes

The training and validation loss plots for the comparison of usage of dropout are shown in Figure 9 and Figure 10.

Training Loss: The training loss converges faster without dropout than with dropout. This is in line with our intuition that not using dropout (probability of keeping the weights = 1) can lead to more over-fitting, and hence lower training loss.

Validation Loss: Validation curve for the model without dropout decreases and then increases. This is clearly a sign of over-fitting, while p = 0.7, steadily decreases with some noise.

7 Comparison of Single-Layered and Double-Layered Decoders

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e-4$
2. Batch Size: 12
3. Keep Probability: 0.7
4. Initialization: Xavier Initialization

Double-Layered decoder allows for higher model complexity, for example, it allows the learning of hierarchical abstractions similar to a CNN or a deep feedforward network.

The training and validation loss for the comparison of number of decoder layers are shown in Figure 11 and Figure 12.

Training Loss: The training loss curves are close, with single-layered decoder performing slightly better than the double-layered one.

Validation Loss: Initially, the single-layered decoder converges faster than the double-layered one. But then, on further training, it slowly begins to increase, whereas double-layered decoder outperforms it.

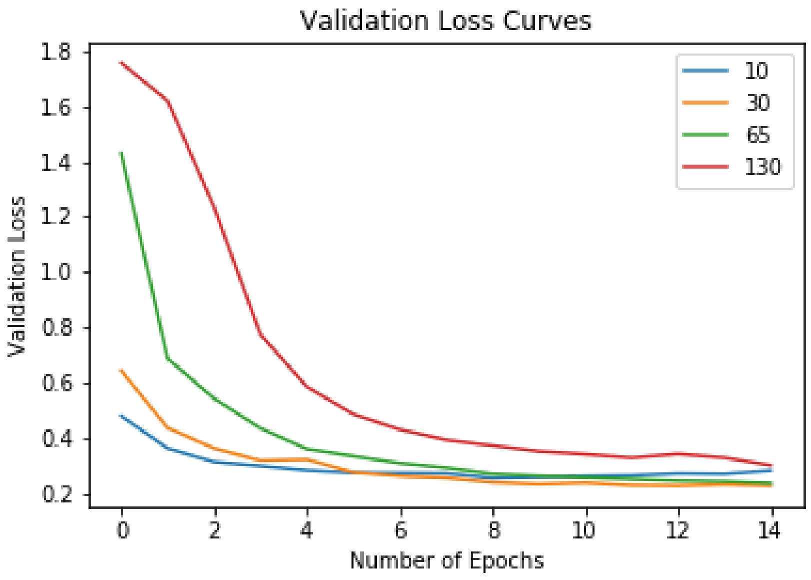

Figure 4: Validation Loss for different batch sizes

8 Comparison of Unidirectional and Bidirectional LSTMs

Rest of the hyper-parameters as as follows:

1. Learning Rate: $8e-4$
2. Batch Size: 12
3. Keep Probability: 0.7
4. Initialization: Xavier

The training and validation plots for the comparison of a uni-directional and bi-directional LSTM are shown in Figure 13 and Figure 14.

Bi-directional LSTMs allow for learning of non-causal relationships between the inputs, unlike an Unidirectional LSTM, which captures patterns in the flow of information occurring only in one direction. Bidirectional LSTMs can perform well in this case since it can be argued that letters in words recently later than the current letter do provide us with useful information about prediction of the current letter, enhancing the understanding of the context.

Training Loss: Bidirectional LSTM slightly outperforms Unidirectional LSTM, as both training loss curves steadily decrease.

Validation Loss: Bidirectional LSTM significantly outperforms Unidirectional LSTM. The validation loss curves for both of them first decrease and then increase, indicating over-fitting.

9 Accuracy's for different hyper-parameters

This section is to show a table which gives information about the validation accuracies for various hyper-parameters settings.

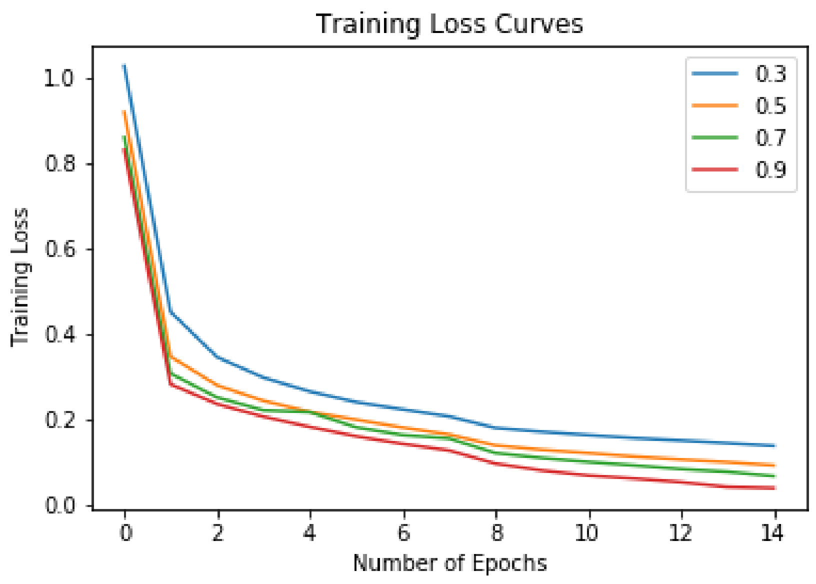

Figure 5: Training Loss for different dropout probabilities

Learning Rate	BatchSize	DropoutProbability	Initialization	Epochs	ValidationAccuracy
$1e-4$	12	0.5	Xavier	15	25.7%
$1e-3$	12	0.5	Xavier	15	40.06%
$1e-2$	12	0.5	Xavier	15	34.03%
$8e-3$	10	0.5	Xavier	15	41.11%
$8e-3$	30	0.5	Xavier	15	39.39%
$8e-3$	65	0.5	Xavier	15	34.46%
$8e-3$	130	0.5	Xavier	15	27.36%
$8e-3$	12	0.3	Xavier	15	38.25%
$8e-3$	12	0.5	Xavier	15	42.16%
$8e-3$	12	0.7	Xavier	15	39.05%
$8e-3$	12	0.9	Xavier	15	40.16%

10 Input and Output Dimension at each layer

Consider a batch-size of 20. The input vocabulary size is 47(including the extra characters added) while the output vocabulary size is 87(including the extra characters added).

Padding is carried out so that each word in a batch is of the same length. Let the maximum length of a word in a batch be 15. Thus each word in that batch is padded to size 15.

The input to the INEMBED layer will be a tensor of size 20*15*47.
The output of the INEMBED layer will be a tensor of size 20*15*256. This output is passed to the encoder. At each time step, the input to the encoder is a tensor of size 20*256. The output of the encoder at each time-step will be a tensor of size 20*256. Since this is a bi-directional LSTM, we concatenate the outputs from the forward LSTM and backward LSTM. Thus the output of the encoder at each time-step is a tensor of size 20*512.
This output is then passed to the attention mechanism. Now consider one time-step at the decoder. The input to the decoder at this time-step will be the output at the previous time-step and the output of the attention

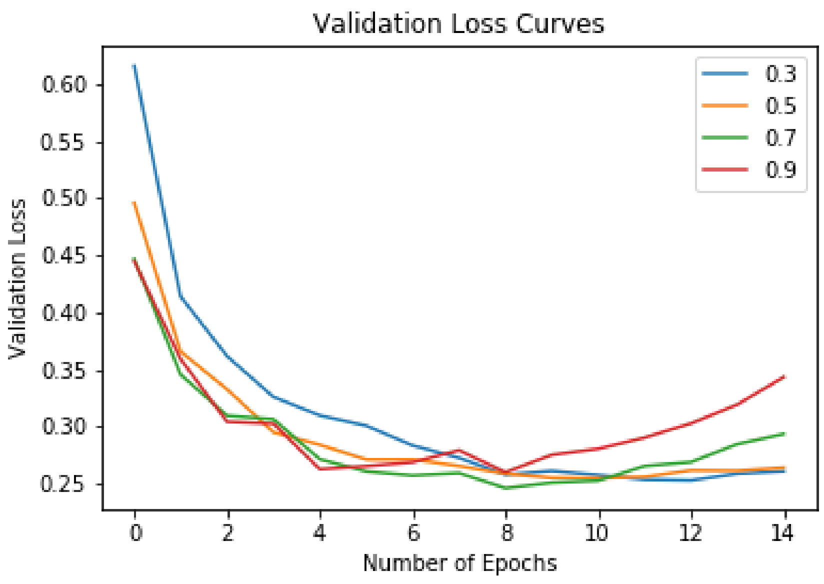

Figure 6: Validation Loss for different dropout probabilities

mechanism. Since the the output of the attention mechanism is just an aggregation of all the encoder outputs, the output of the attention mechanism is a tensor of size 20*512. This is concatenated with the output of the decoder at the previous timestep. Hence the input to the decoder at a time-step is a tensor of size 20*768.
The output of the first layer of the decoder at a time-step is a tensor of size 20*512.
The output of the second layer of the decoder at a time-step is a tensor of size 20*512.
Now, this output is passed through a feed-forward network to obtain the prediction of the letter at that time-step. Thus the output is a tensor of size 20 * 87.
This is passed through another feed-forward layer to obtain OUTEMBED. The size of the OUTEMBED vector is 20*256. Note that all of these sizes are for a certain time-step of the decoder.

11 Attention Mechanism

The attention mechanism that we used is Bahdanau Attention. This is an additive style of attention mechanism. This type of attention mechanism takes inputs as the encoder outputs, the previous state of the decoder and the output of the decoder at the previous timestep. The equations are as follows :

$$\alpha_{ts} = \frac{e^{score(d_t,h_s)}}{\sum_{k=1}^{S} e^{score(d_t,h_k)}}$$

These alphas are the weights which will be multiplied to the encoder outputs to get the context vector. Here, h_s are the outputs of the encoder, while d_t is the state of the decoder at the previous time-step.

$$c_t = \sum_s \alpha_{ts} h_s$$

$$score(d_t, h_s) = V^T tanh(W_1 d_t + W_2 h_s)$$

Attention mechanism allows the model to focus on relevant parts of the input. For example, the middle section of the word generally does not depend on the beginning or the ending of the word, hence focusing

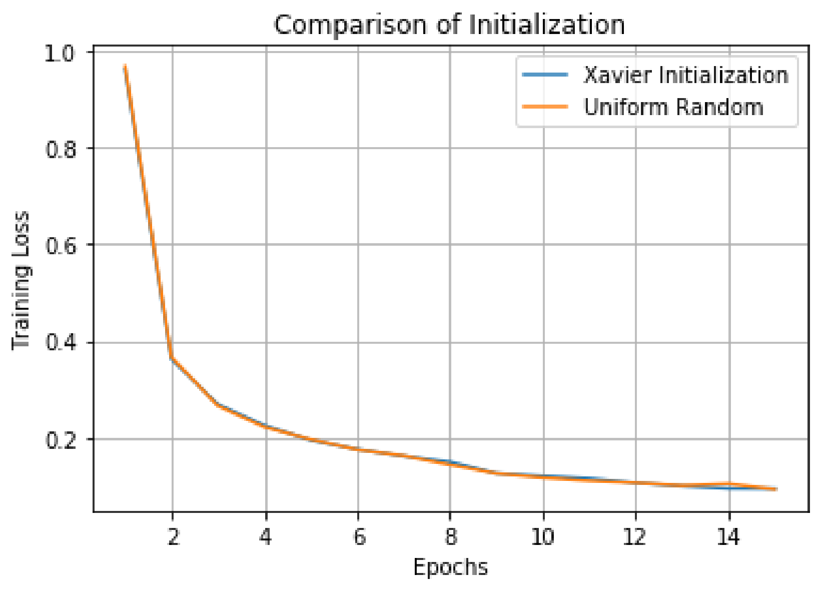

Figure 7: Training Loss for different initialization

on that portion of the word only would help during prediction. Attention mechanism allows this as the model learns on what parts of the input to focus.

The plots of the attention weights for a sequence pair is shown in Figure 15. The input sequence is the word ABDUH and the weights are shown for its translation obtained. We can observe that there is a one-one alignment for the Hindi translation of H, while a many-one alignment for the Hindi translation of A. there is also a many-one alignment for the translation of D. We do not observe any non-contiguous alignments.

12 Early Stopping

We have used validation loss(with a patience of 5) instead of validation accuracy for Early Stopping. One of the reasons for this discrepancy is the hard thresholding used for calculating the accuracy(since ultimately we are going to predict a single class) for an input even if the predicted probabilities of multiple classes are close in their values. This problem is aggravated in case of large number of classes. It is important not only to increase the probability of the correct class, but also to maximize the gap between the predicted probabilities of the correct class and that of the next lower predicted probability, which is captured in the validation loss, but not in validation accuracy.

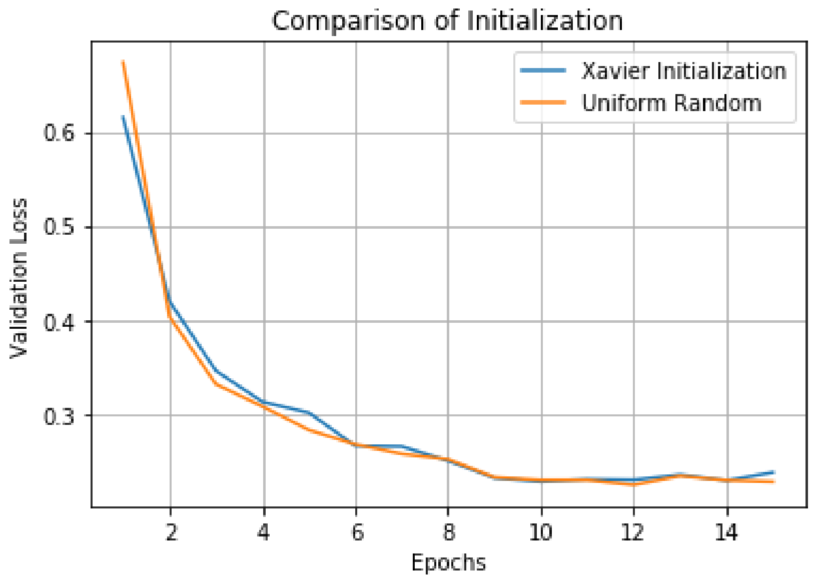

Figure 8: Validation Loss for different initialization

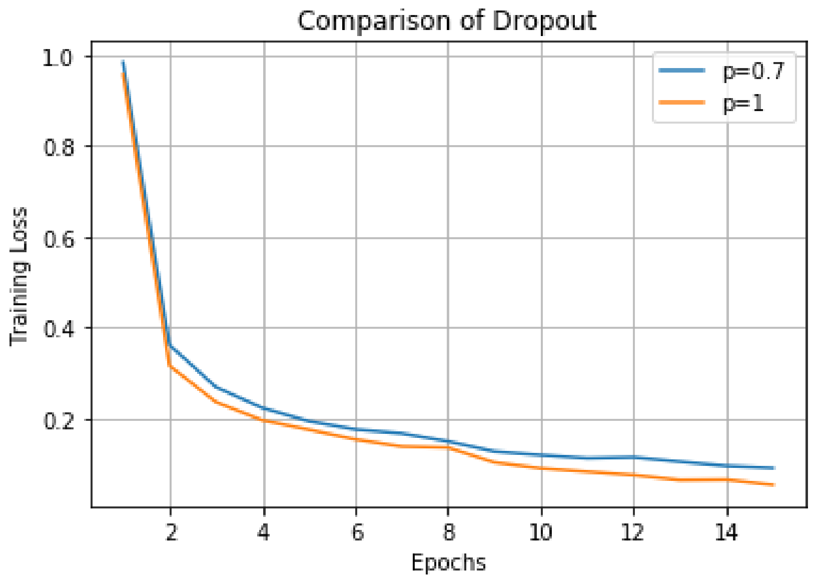

Figure 9: Training Loss

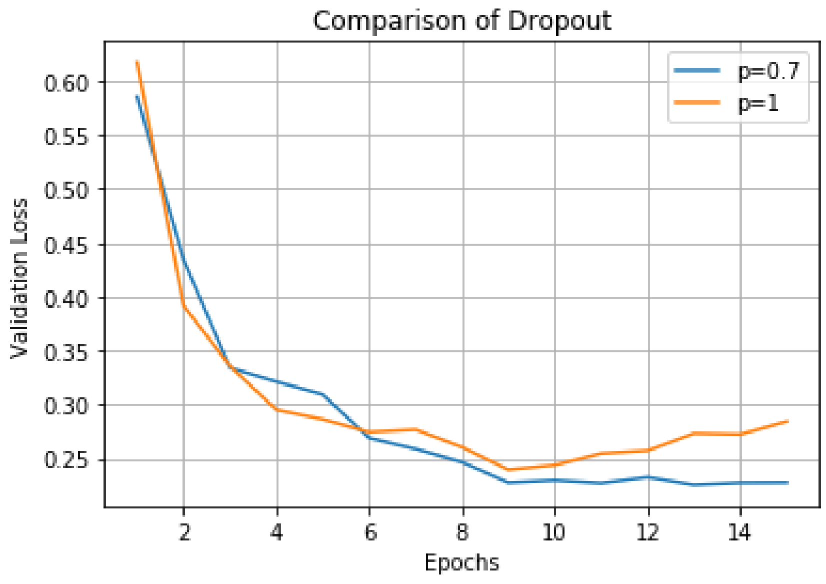

Figure 10: Validation Loss

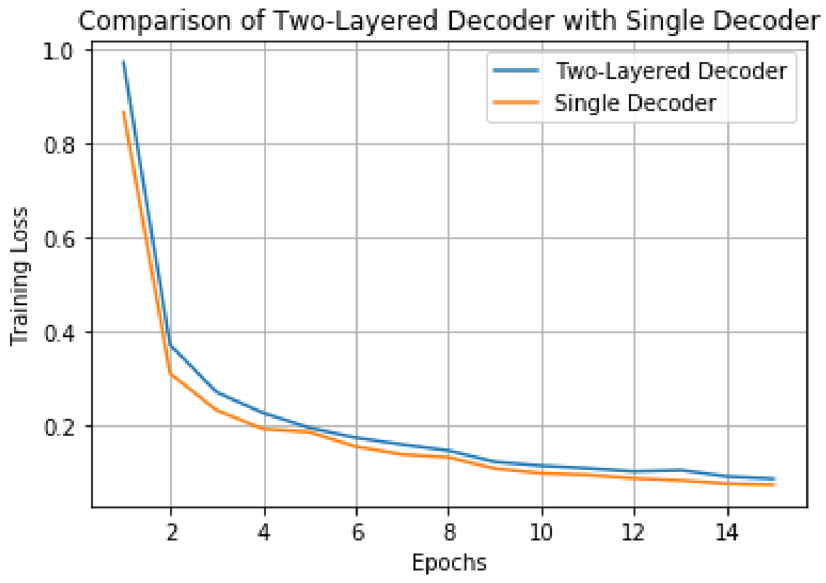

Figure 11: Training Loss for the number of layers in the decoder

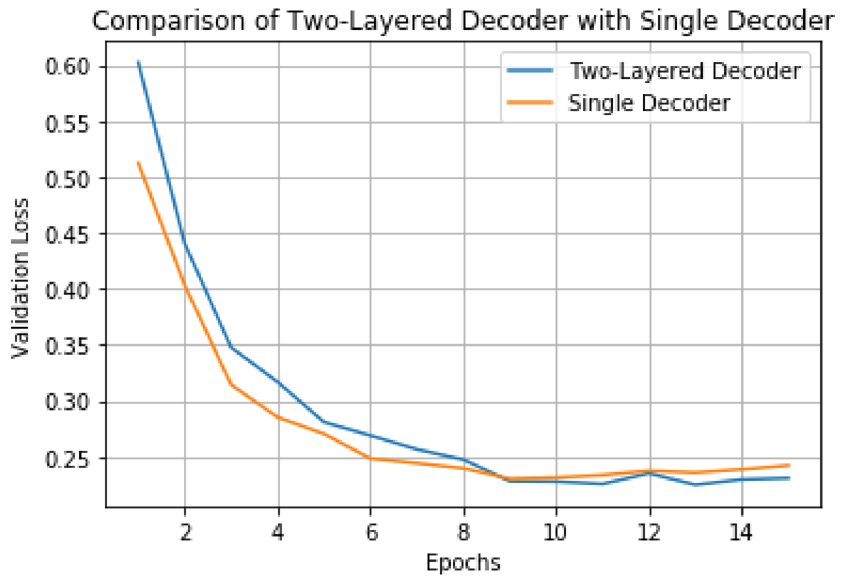

Figure 12: Validation Loss for the number of layers in the decoder

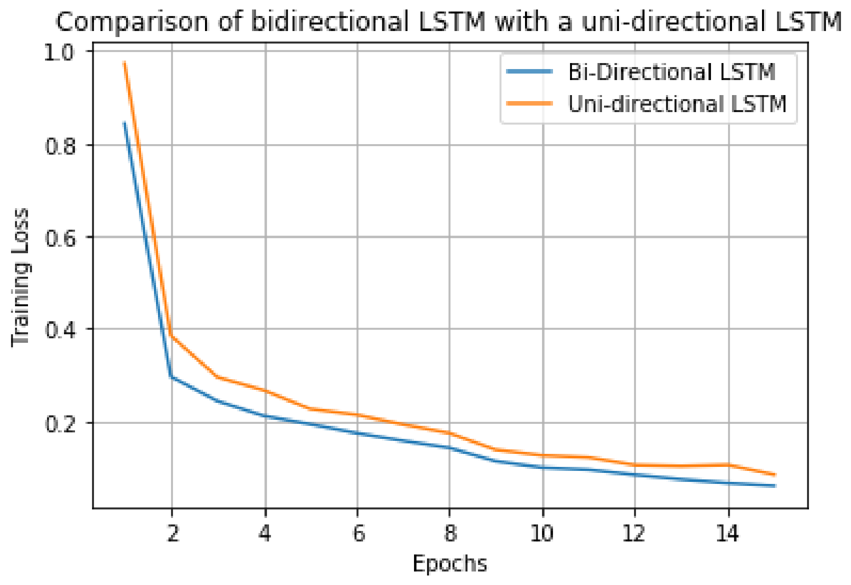

Figure 13: Training Loss

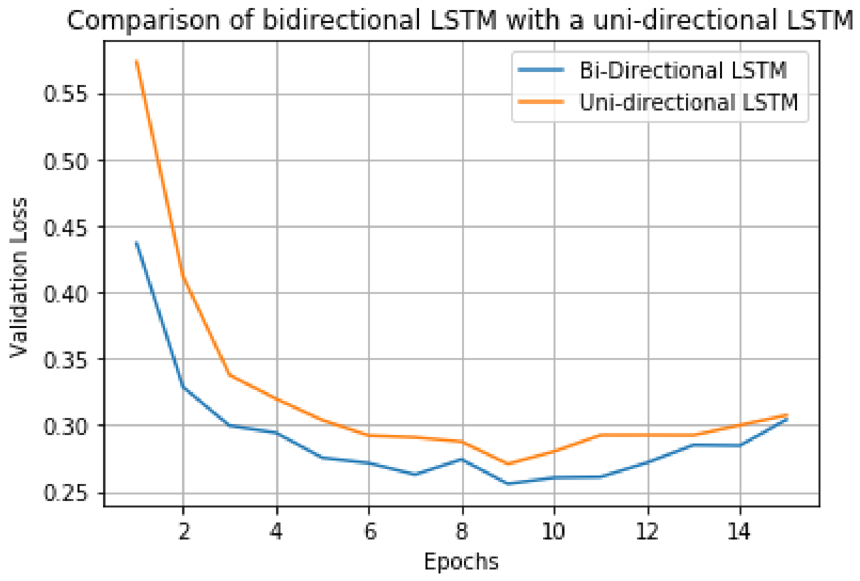

Figure 14: Validation Loss

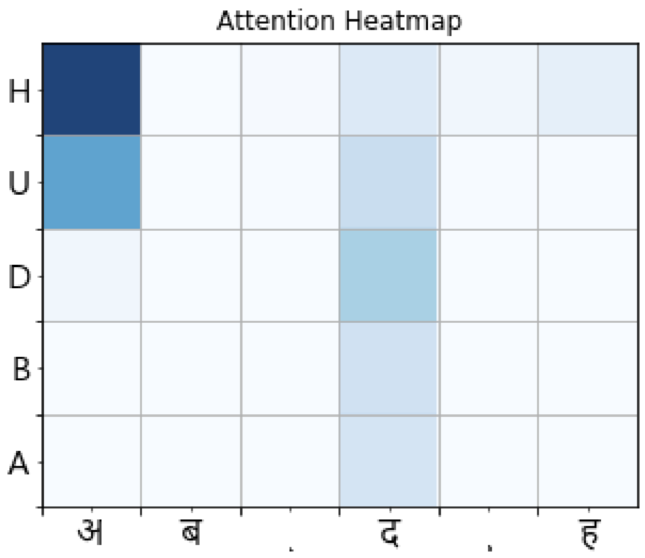

Figure 15: Plot of attention weights