Assignment 2: Sentence Representations (Report)

Viraj Kamat
112818603

Section 1

Implementing DAN: Within the init function of the DanSequenceToVector class, four dense layers (fully connected) were defined with parameters of activation as Relu and use_bias as True. The number of layers to be defined is governed by n_layers parameter.

The call function is where the layers are utilized. The first step was to compute the sequence mask and the dropout. The dropout was first computed by taking a mask with a Bernoulli distribution of 0.2 and then performing a logical 'and' with the sequence mask. The resultant mask was a tensor with boolean True/False values which meant when applied upon our sentence matrix (vector_sequence) would remove those words where the mask has a value of False. When the training parameter was set to False the dropout was excluded and only the sequence mask was applied.

As with Deep Averaging networks we first take an average of our word vectors and apply non-linear transformations upon it. In the call function, for all the sentences in the batch using the tf.reduce_mean function once the average of the sentence is computed we call upon iteratively each of the dense layers defined within the init function.
Each iteration performs a non-linear transformation upon the averaged vector and discerns some information of the sentence. At each layer we save the state of this layer in an array. The final output of the these sequential non-linear transformations is the combined_vector, the combined_vector along with layer representations at each stage is returned.

Implementing GRU: Implementation of GRU was straight forward, we defined four dense layers (based on the num_layers parameter) within the init function of the GruSequenceToVector class using the keras.layers.GRU function. We set the return_sequences and return_state values to True, this enables capturing the cell state at each layer which are stored in an array and the layer representations which matters only for the last layer. The layer_representations are the cells states captured in an array and the combined vector is the output of the last layer. Within the call function of the GruSequenceToVector class we simply call each of the layers defined within the init function iteratively and we return the layer representations and output of the last layer.

Implementing the Probing Model: The probing model was a simple linear classifier that used a softmax function to classify results. This was implemented using a single tensorflow layer within the init function of the ProbingClassifier class. Then within the call function the input parameter was first fed into the pretrained model and the output of the last layer was captured and fed into the linear classifier that we defined in the init function. This classified output known as logits was then returned.

Section 2.1: Learning Curves

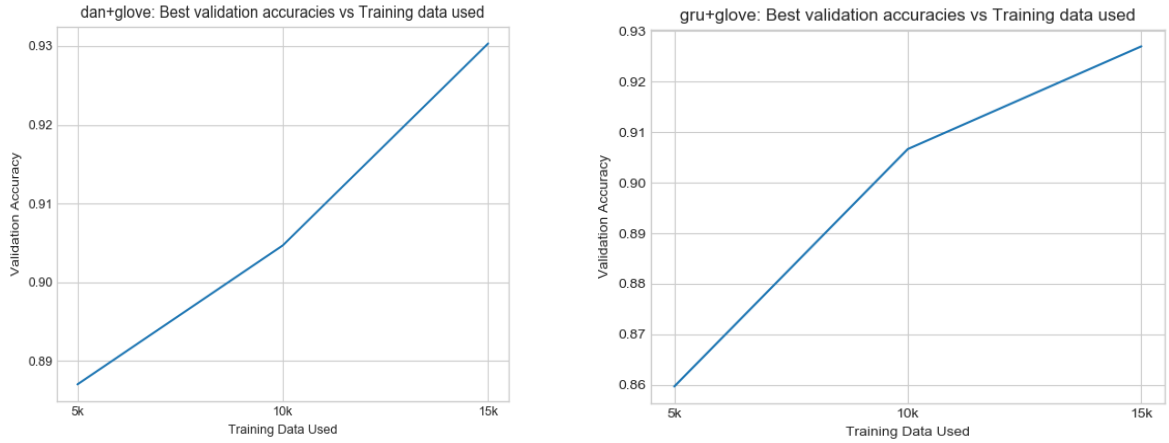

Figure 1: DAN VS GRU on increasing training data

We can see that as the training data was increased both DAN and GRU showed improved accuracy. With each step of choosing a larger dataset the accuracy improves, however GRU approaches higher accuracy quicker and this evident from the above graphs. Thus we can say that GRU is better than DAN at sentiment analysis.

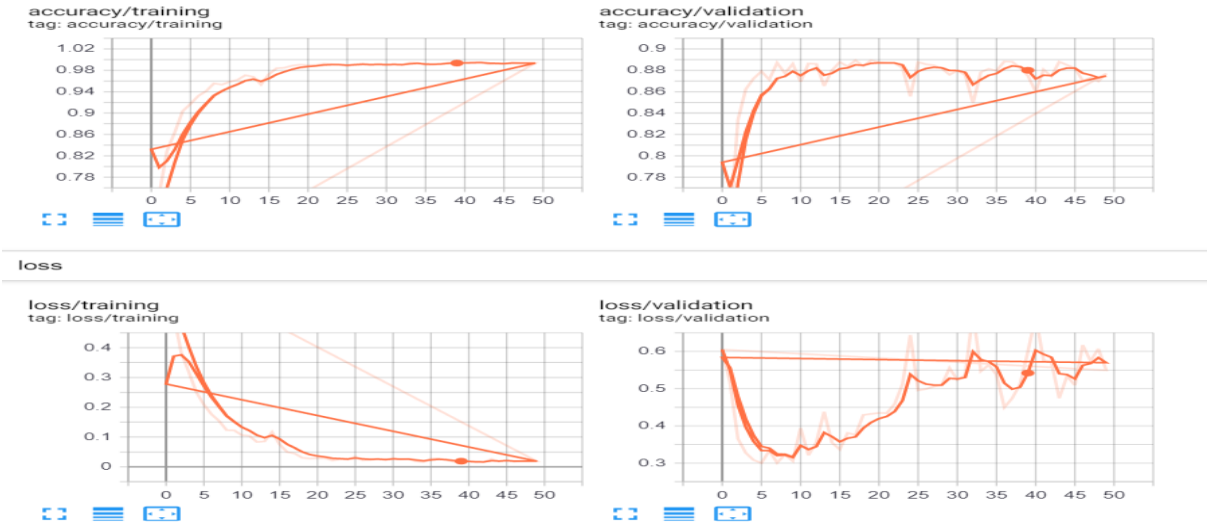

Figure 2: Training DAN for 50 epochs

The above shows the graphs when training DAN for 50 epochs, here as the number of epochs increases so does the accuracy, which makes sense. With each epoch the loss further decreases, however after a certain point the loss starts increasing, this is the point to stop training. It would be better not to train DAN for many epochs as it may lead to some form of overfitting.

Section 2.2: Error Analysis

2.2.1:

GRU can could give us better accuracy than DAN, however GRU takes a longer training time and takes several iterations before it gives us a trained model that could be used for useful predictions, i.e GRU is time consuming.

In contrast to this DAN reaches higher accuracy within shorter iterations but as iterations get increased the accuracy remains nearly the same. Thus, DAN takes shorter time than GRU to train and could theoretically be used in less complex data-sets to obtain a meaningful model quicker.

2.2.2

Since DAN can find a lot of difference in fewer iterations, without regard for word order we can used DAN on very large sentences to be find the sentiment of a sentence quicker, while for large sentences GRU would take a long time to discern the sentiment of a large sentence.

On the other hand in shorter sentences where the word order matters to identify sentences, we can use GRU for our analysis, if we did use DAN for shorter sentences where the word order matters it would likely perform badly.

Section 3.1 Probing sentence representation for sentiment task

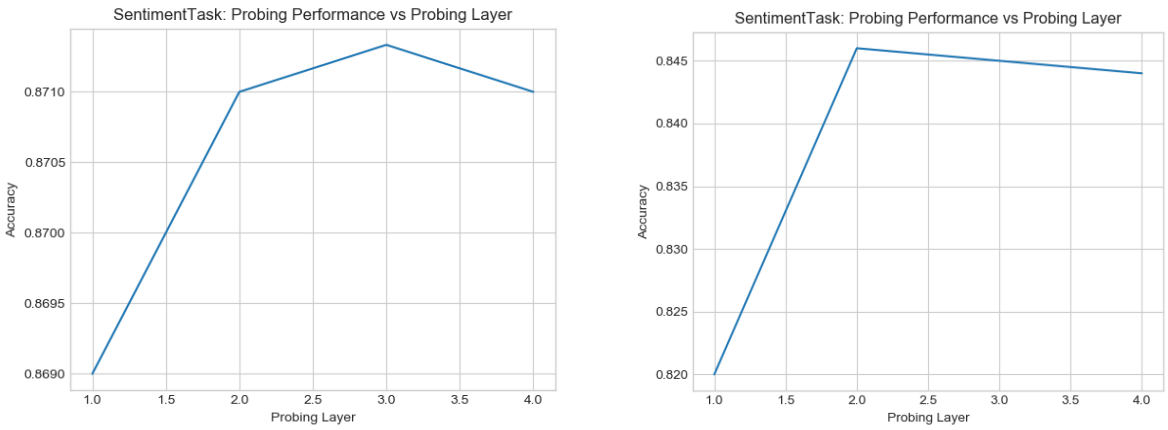

Figure 3: Probing performances on sentiment analysis (DAN vs GRU)

On the left is the graph for DAN which shows the accuracy when probing the each of the layers. What could be discerned from this graph is that as the number of layers increase the accuracy increases, however adding too many layers could indeed lower the accuracy, and that could be due to overfitting our model as is evident in the last layer.

In GRU however as accuracy is reached within a few layers but as the number of layers increases the accuracy decreases. Thus, we would have to make sure that GRU based models get trained for fewer layers to train a model.

Section 3.2 Probing sentence representation for Bigram Order

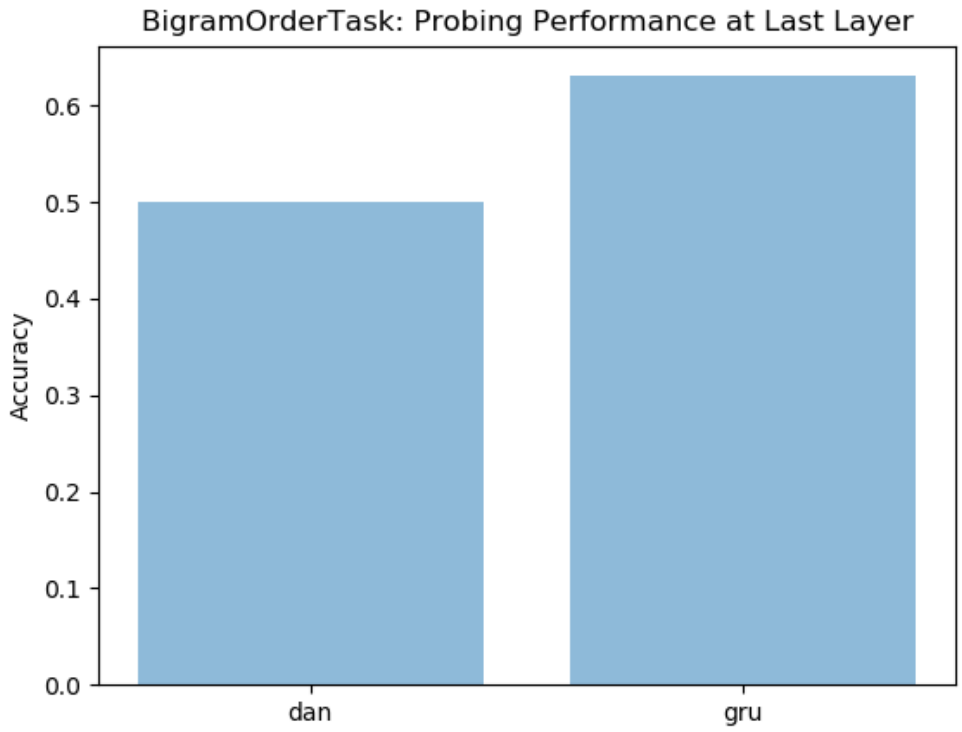

Figure 4: Probing performance on Bigram Order

We can see in the final layer of DAN and GRU in the above plot for Bigram performance analysis for the last layer. Since DAN takes an average of the word vectors as opposed to GRU, it would mean that the word order gets lost, which is not the case with GRU. GRU here can be seen performing better since it preserves the word order when it is passed through each cell in the RNN while also updating cell states in sequence; DAN on the other simply aims to discern information from the average of word vectors without any regard for the word order.

Section 3: Analyzing Perturbation response of representations

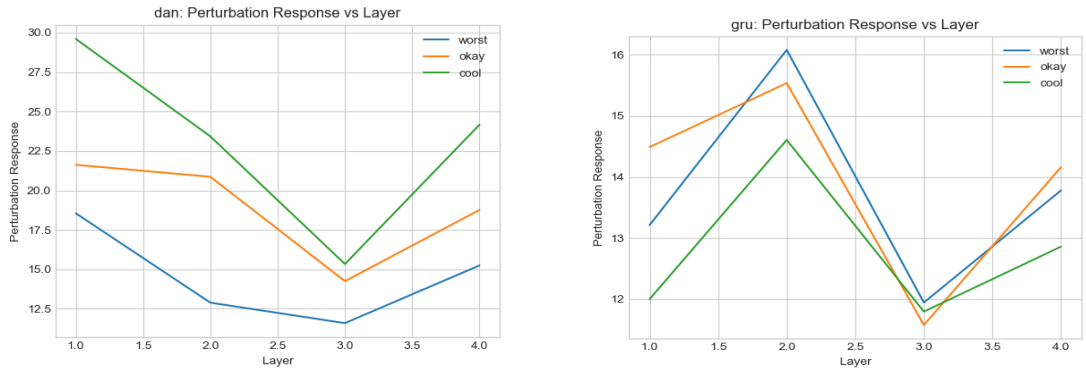

Figure 5: Perturbation response for DAN and GRU

We analyze the perturbation for the sentence “The movie performance was awesome” by changing the word awesome 3 times. We can see than in DAN when the word ‘awesome’ was replaced by ‘okay’ and ‘cool’ the resultant sentence representations were close to each other at every layer. When the word ‘awesome’ was replaced by ‘worst’ the resultant sentence representation for far off from the others.

For GRU however, the sentence representations seemed a bit skewed when the word awesome was replaced with ‘worst’, ‘okay’ and ‘cool’. Only in the fourth layer however we could see some differences being discerned in the sentence representation which means it would take more layers to better identify perturbation response.