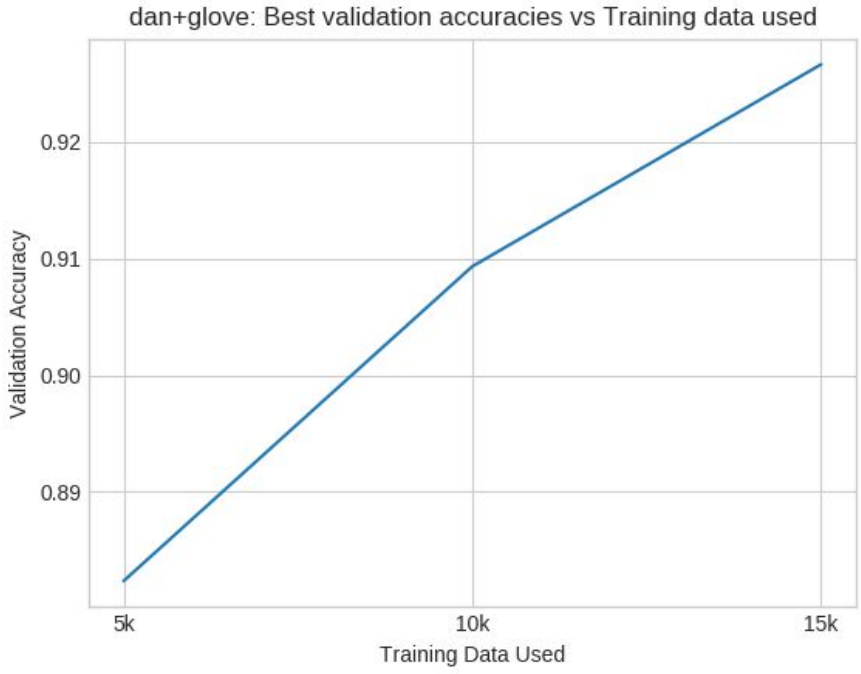
Niranjan Agnihotri (#112686660) **NLP Homework 2**

Implementation Details -

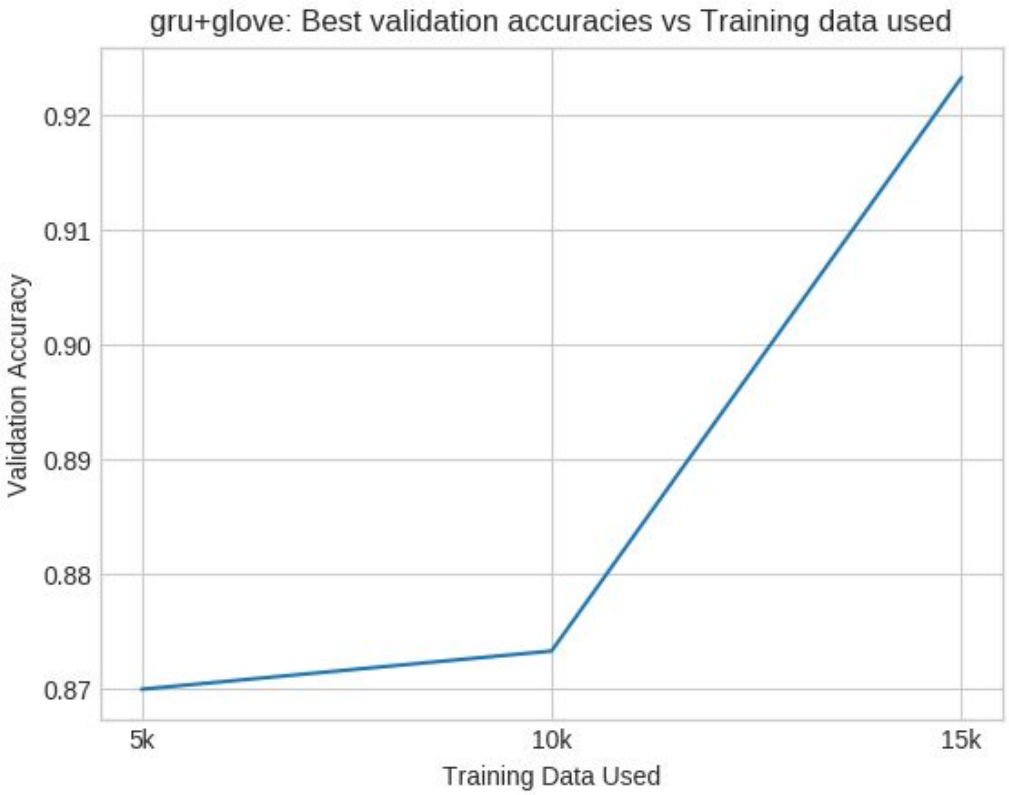
1. DAN -
    a. In the init-method, an array of Tensorflow Dense layers were initialized and stored. The layers activated with the 'relu' function.
    b. The call method was implemented by processing the inputs with the sequence mask. Then random numbers were generated from interval (0, 1) to implement word the drop-outs. Then the input vector of dimension (64, 209, 50) was reduced to (64, 5) [by taking mean] and fed to the Dense layers. The intermediate layer outputs were stored in an n-darray and was returned along with the output of the last layer.
2. GRU -
    a. GRU units / layers equivalent to *num_layers* were created and stored in an array.
    b. In the call method, the sequence mask was applied, to filter out padded words. And the input was fed to the first GRU layer. It's output was then fed to the remaining GRU layers and all the intermediate layer states were recorded.
    c. The recorded intermediate layers were returned along with the representation / output of the final layer
3. Linear Classifier -
    a. The classifier was defined as a simple Dense layer with *classes_num.* The pretrained model's weights were kept frozen.
    b. In the call method, first, the inputs were passed to the pretrained model. This returned the layer states for all the layers. Then I extracted the layer state for the layer no that was specified. It was then passed to the linear model, whose outputs were then recorded and returned as logits.

3.1 Learning Curves

- **The effects of increasing training data -**



Observation - Increasing training data linearly increases accuracy in the case of DAN. With increasing data, we also achieve a higher accuracy on our task. This is evident on the basis of the above graph.



Observation - Increasing data in the case of GRU doesn't linearly increase the performance in terms of accuracy, but after a certain threshold, it skyrockets. Thus the increase in the accuracy is non-linear in-terms of GRU.

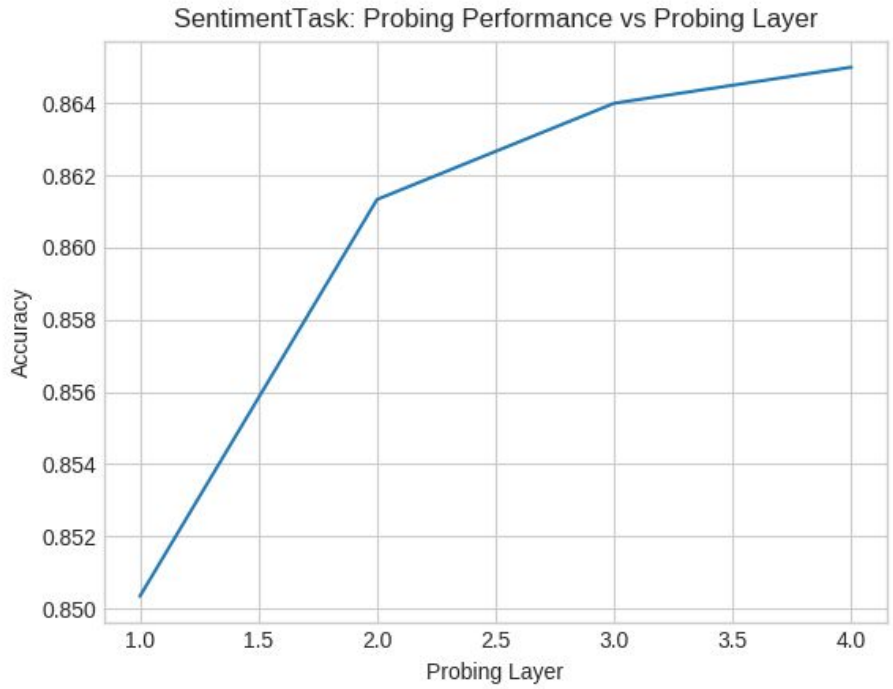
- **Increase training time (number of epochs) -**

Above were the pics obtained by training DAN for 50 Rounds. I can see that that with increased training time, the training loss was minimized with an increased training time but that does not necessarily translate to a decreased validation loss. Therefore, indefinite training for a longer time does not necessarily help. One should always train a model to a time where both the training and validation losses go down. The moment, the validation loss starts going up comes the time for us to stop the training. In the above case, it would be good to train for no more than 10 rounds.

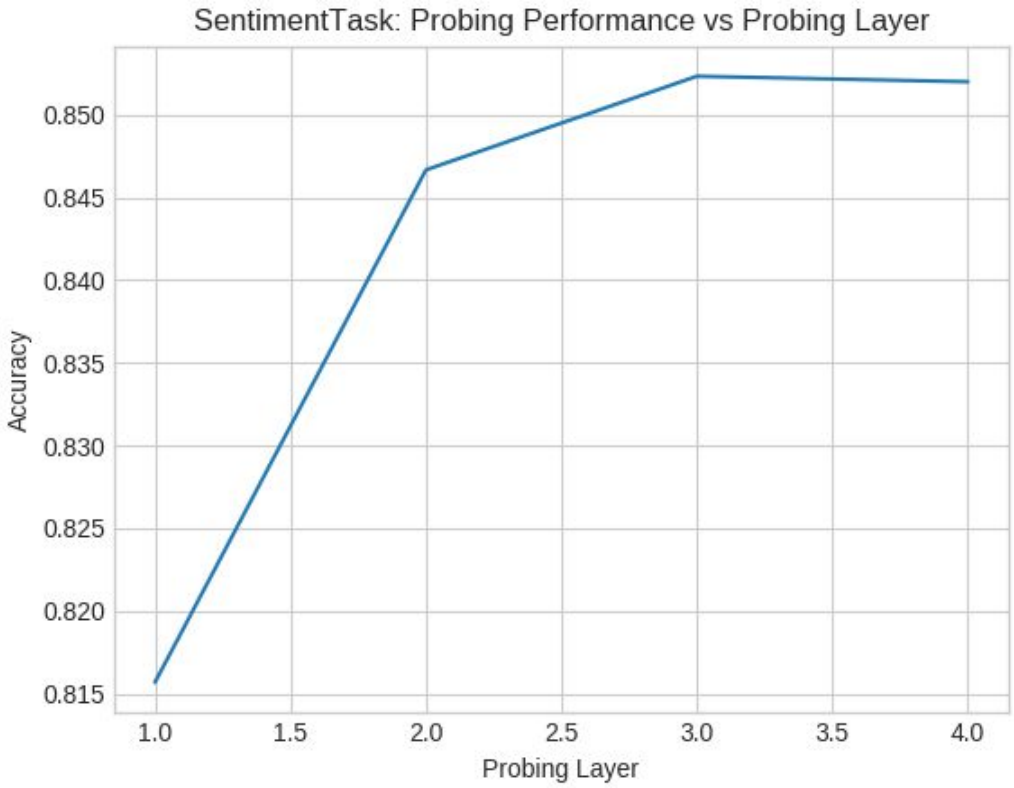
3.2 Error Analysis

- DAN is computationally less intensive to train. It is also an intuitive and simple model to work with.
- GRU is more powerful and complex than DAN. Therefore, it is able to capture, complex sentence representations. It's results are also far more superior than that of the DAN
- Failure case Analysis -
    - GRU over DAN
        - One case is during perturbation response analysis - GRU is able to amplify the distance between sentence representations of 'worst' and the other two words 'okay' and 'cool' well in advance. To be specify, right from the first layer. While for DAN the same sentences are not separated by a large distance in layer one. Due to averaging functions, all the components of a sentence mix up with equal weights in DAN which is undesirable. This does not happen in GRU's case.
    - DAN over GRU
        - For the same perturbation analysis task, the advanced layers of the DAN are able to do a fair job in keeping the sentence representations for sentences containing two different words 'okay' and 'cool' separate. While the advanced layers in the GRU model tries to represent the sentences separated with very low distance. Essentially implying that they mean the same.

4.0 Probing Tasks



Graph for DAN



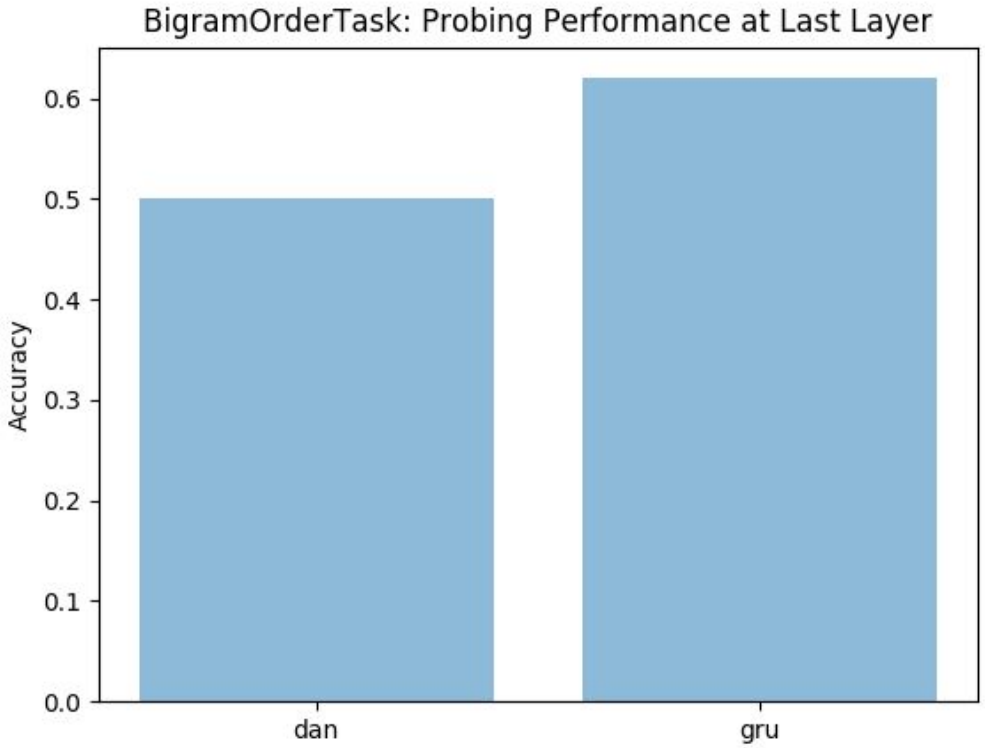
Graph for GRU

Observations -

From the above two plots it is clear that the performance of the layers that are at a greater depth is more than the performance of the layers which come near to the input. The difference in the performance of the 1st layer and the last layer does not vary significantly, this implies that the first layer manages to capture significant information from our input tensors. And the more deep layers, try to capture finer patterns on the top of it, to improve the overall performance.

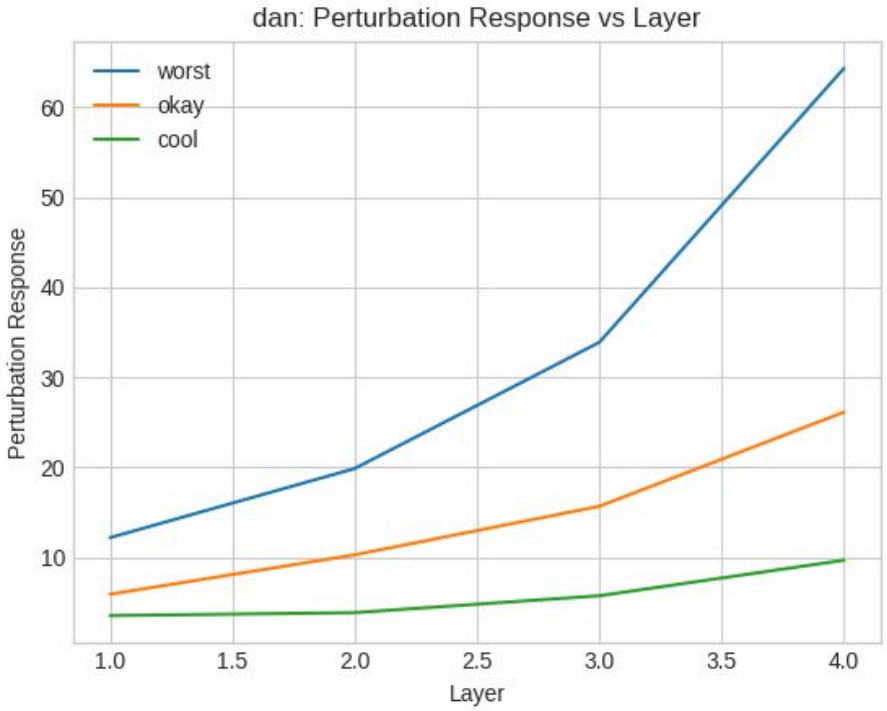
The performance saturated after the 3rd layer during my experiments with GRU while for DAN is seemed to be slightly increasing even after adding the last layer.

Comparison DAN vs GRU



In the bi-gram comparison task however, GRU seems to way outperform DAN, as seen in the previous graph.

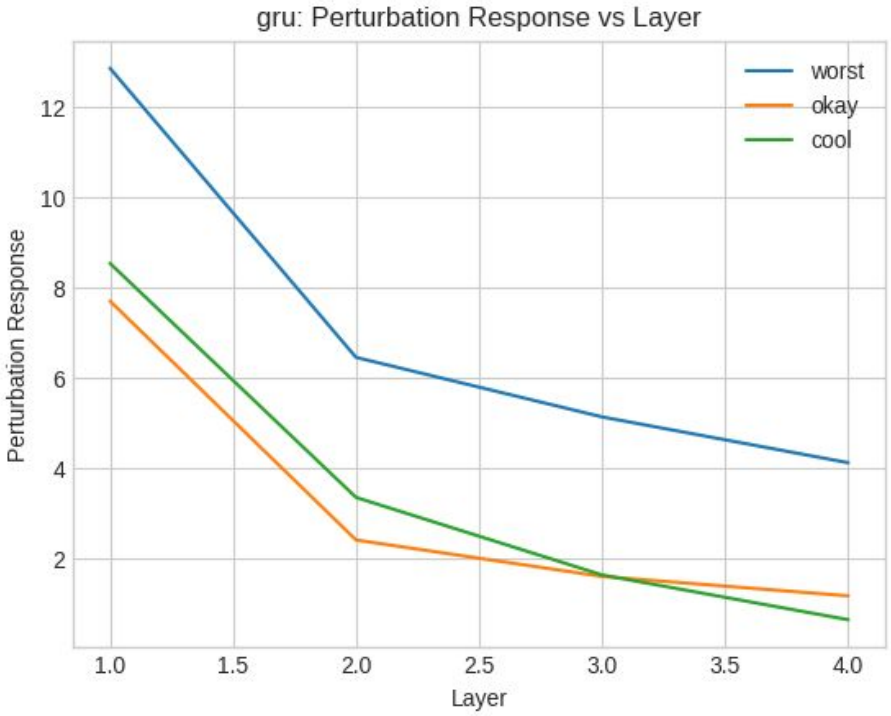
Perturbation Response -



Perturbation Response for DAN

Observations -

When words ‘words’, ‘okay’, and ‘cool’ are interchanged, DAN amplifies the distance between the corresponding sentence vectors to a reasonable difference at it’s advanced layers. So, at layer 1 for DAN the differences are not that much in the sentence representations. The sentence representations with words ‘okay’ and ‘cool’ are reasonably spaced.



Perturbation Response for GRU

Observations -

GRU being more complex than DAN is successfully able in maintaining a reasonable difference between sentence representations the ones containing words ‘okay’ and ‘cool’ and the ones containing ‘words’. This is a desirable property. And unlike the DAN, GRU is able to do this even in a reasonably shallow (with less number of) layers. This is one big advantage of GRU over DAN.