Rana Mohamed Barakat (6926) Mohab Mohamed Ali (7199)

Report for assignment 4

Most of what will be described in this report has already been mentioned in the attached paper, but the report provides more detailed explanations and comments on the graphs shown in the notebook.

Vocab selection:

Our vocab selection strategy ended up being like the one used in the original paper where the most common answer is selected, and tie breaking is done via levenshtein distance to find the answer that most describes all answers. However, this was not our first choice. In our first attempt, we used clip to encode each image and all 10 answers per image, then we used the example code on clip's github repo to find the probabilities of each answer being the caption to its respective image. While this approach might make sense, it is not human and that is why it produced too much variety and ultimately poor results (~12000 classes, ~%20-25 accuracy). The next attempt was to choose the most common answer and tie break using the previous captioning strategy. That seemed to improve accuracy and loss but not by much (~10000 classes, ~%29 accuracy). This gave us an indication that we need to completely get rid of the clip captioning strategy. Next, we used the most common answer with tie breaking by choosing the answer that appears most in the entire train set. This improved results a lot compared to the previous two attempts (~6000 classes, ~%45 accuracy). Finally, we implemented levenshtein distance for tie breaking and it produced the best results although they were not far from the previous attempt (~4000 classes, ~%48-50 accuracy). We also pre-emptively chose 'unanswerable' as the answer for questions where the answerable attribute was 0. Since some answers were 'unanswerable', 'unsuitable', 'unsuitable-image' combining them all into one class would give us higher accuracy and would help the model learn the similarities better. Our final vocabulary had 4960 classes with 4 answer types.

Encoding:

As per the paper, we used rotation for image augmentation during encoding. We would rotate the image 90 degrees and -90 degrees. Then we would encode each of the 3 images separately. Then we would find the mean of the embeddings of the images. We also encode the tokenized questions and append them to the images (images go first). We used different clip models for encoding which produced varying results, but all the results were in the same range %48-50 accuracy. We encoded the final answers from string to one hot encoded vector to represent the labels. We used label binarizer as opposed to one hot encoder for readability and as a good practice as one hot encoder is usually used for features. Although there is no major difference between the two encoders. We did not use label encoder as that may produce bias in the model towards some answer but the answers are not ordinal so it wouldn't be fair.

After fitting the encoder to the training data and transforming the validation set, we found that out of 3173 answers, only 2595 answers from the validation set were found in the vocabulary we had built. This meant that the model was always going to misclassify at least 578 image/question pairs.

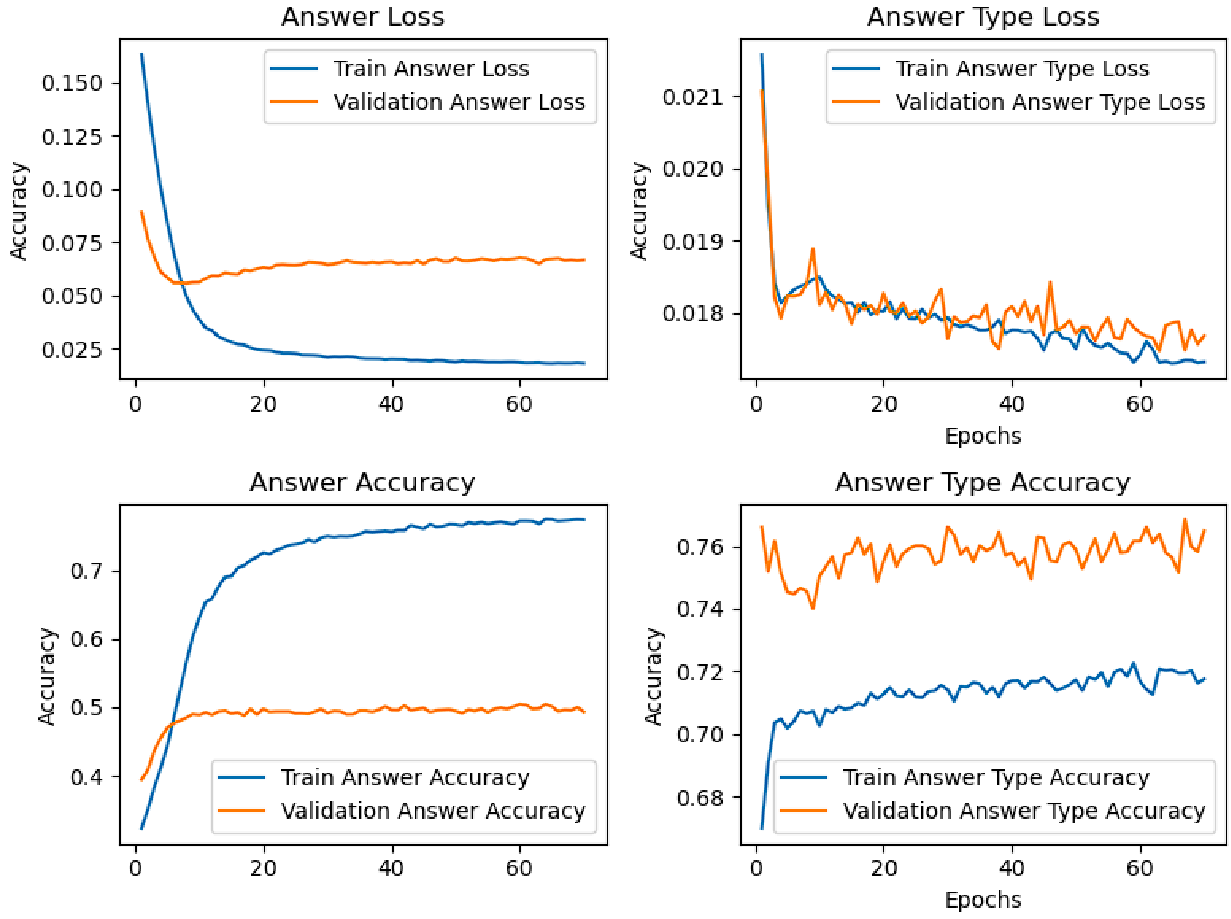
The model:

We followed the implementation of the paper where the concatenated image and text were normalized then passed to a fully connected layer whose output split into two branches. The main branch was another fully connected layer that projected to the vocab size. The auxiliary branch first projected to the number of possible answer types (4) then projected back to the vocab size with a sigmoid activation function. Both branches were then elementwise multiplied to produce the classification of the answer. We used dropout and layer normalization between layers to avoid overfitting. We used Adam optimizer with a learning rate of 1e-4 and weight decay of 1e-5. We used cross entropy loss for both the answers and the auxiliary answer type predictions. Initially, we trained each model for 200 epochs. However, we soon realized that after around 50 epochs the validation accuracy starts to decline, and the train accuracy continues to increase so we decided to train for 70 epochs instead.

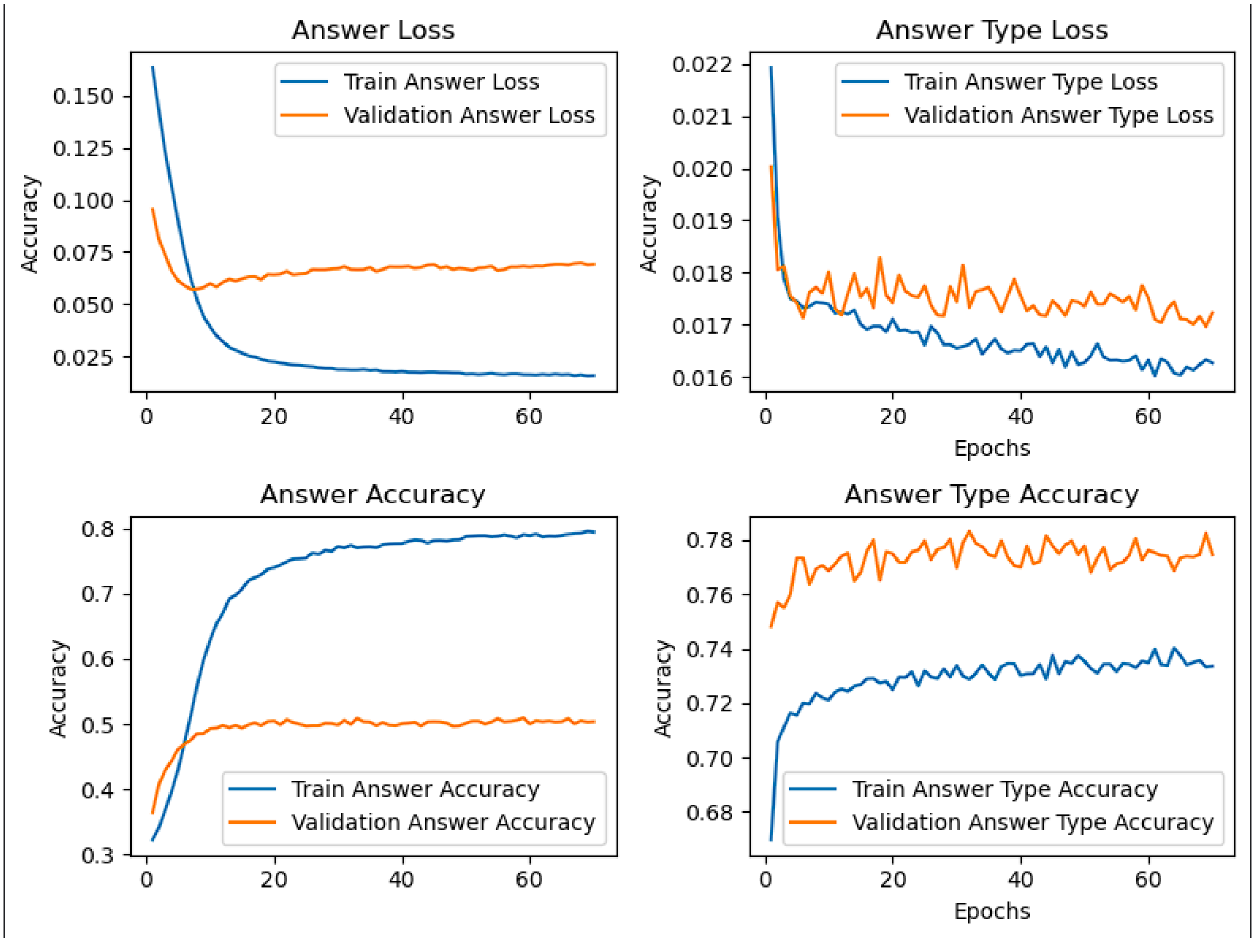
Results and plots:

We trained the same train set on 4 different models. Each model used the same architecture we built but had input from different clip model encodings. Here are the results:

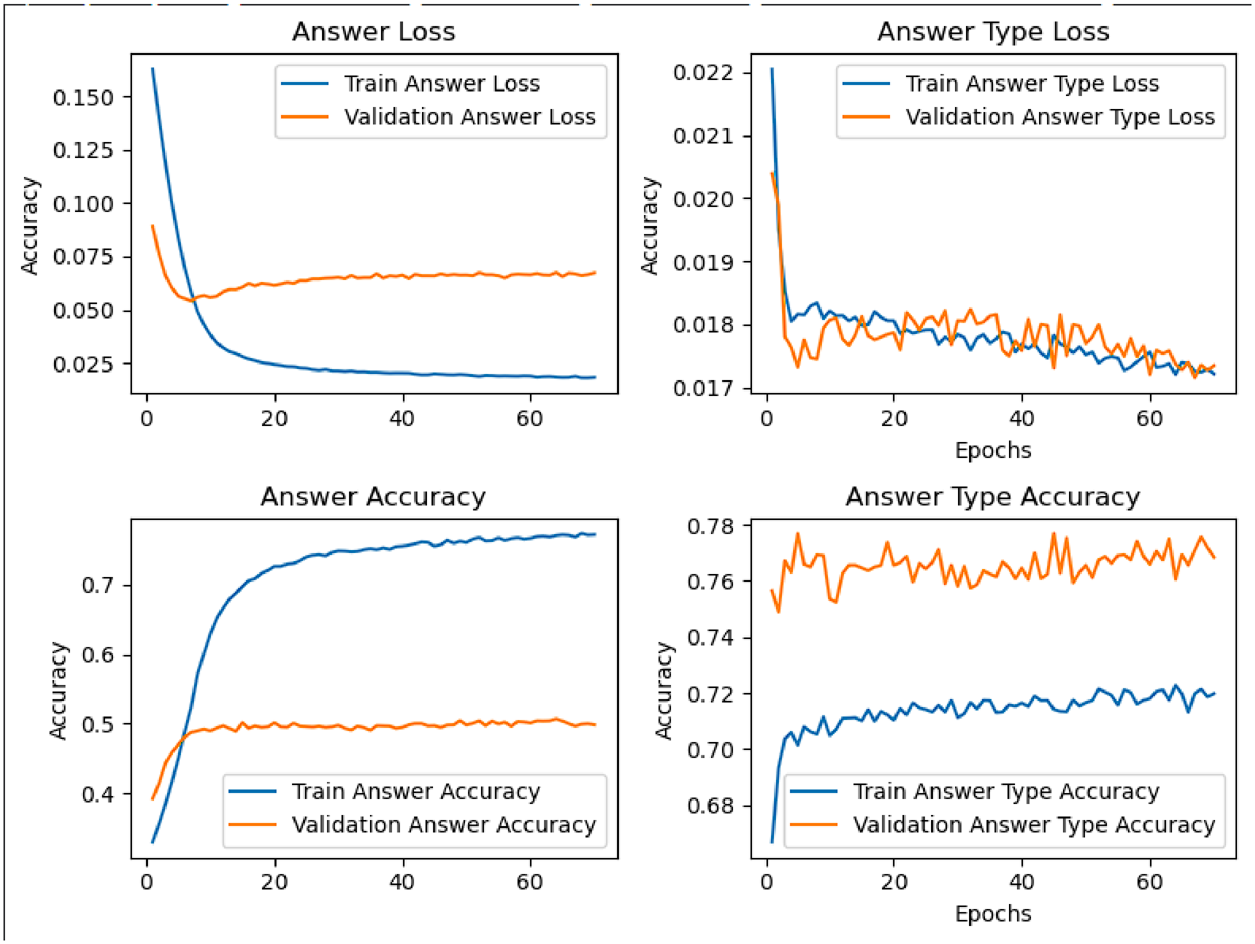
ViT-L/14:


Answer Loss
Train Answer Loss
Validation Answer Loss
Accuracy
0.150
0.125
0.100
0.075
0.050
0.025
0
20
40
60
Answer Type Loss
Train Answer Type Loss
Validation Answer Type Loss
Accuracy
0.021
0.020
0.019
0.018
0
20
40
60
Epochs
Answer Accuracy
Accuracy
0.7
0.6
0.5
0.4
Train Answer Accuracy
Validation Answer Accuracy
0
20
40
60
Answer Type Accuracy
Accuracy
0.76
0.74
0.72
0.70
0.68
Train Answer Type Accuracy
Validation Answer Type Accuracy
0
20
40
60
Epochs

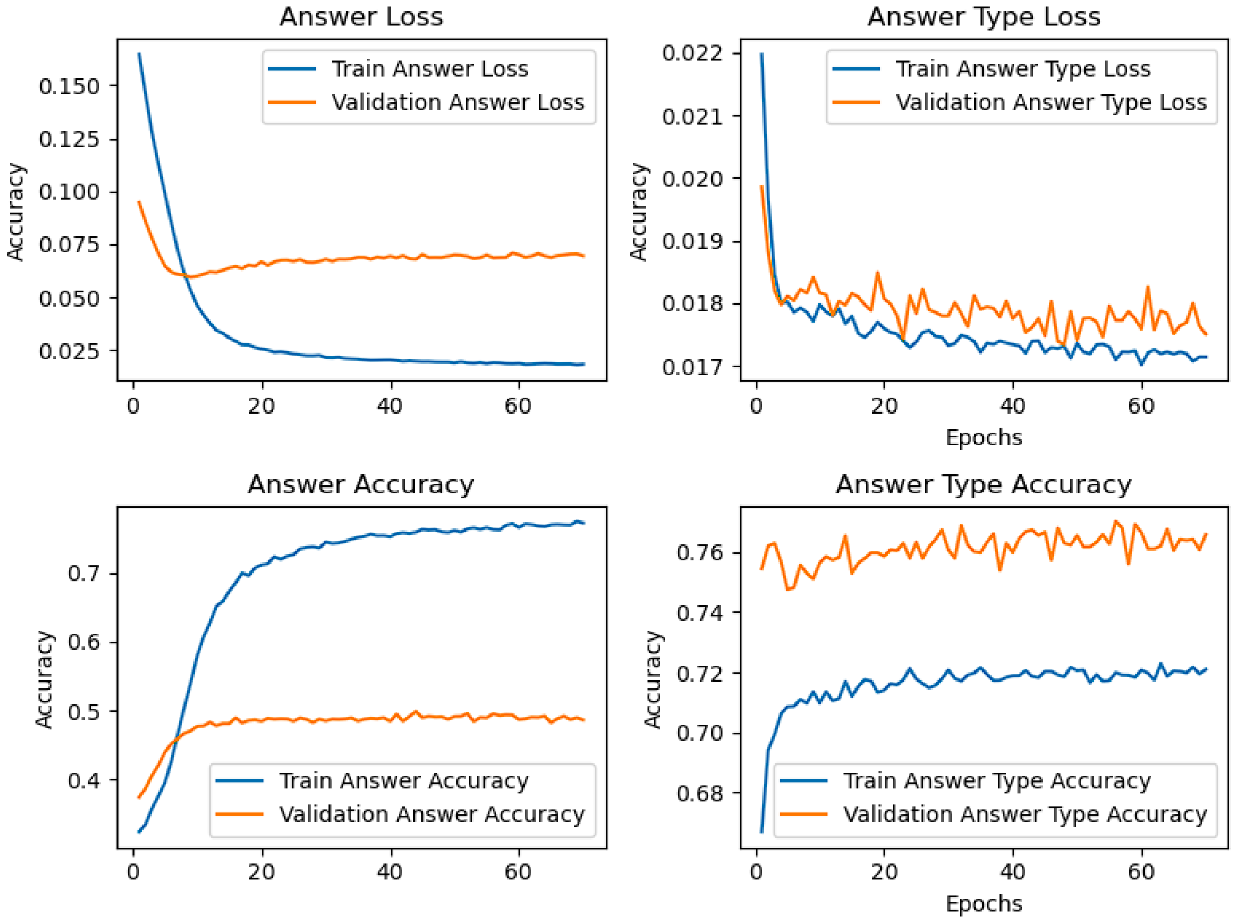
RN50x64:


Answer Loss
Train Answer Loss
Validation Answer Loss
Accuracy
0.150
0.125
0.100
0.075
0.050
0.025
0
20
40
60
Answer Type Loss
Train Answer Type Loss
Validation Answer Type Loss
Accuracy
0.022
0.021
0.020
0.019
0.018
0.017
0.016
0
20
40
60
Epochs
Answer Accuracy
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
Train Answer Accuracy
Validation Answer Accuracy
0
20
40
60
Answer Type Accuracy
Accuracy
0.78
0.76
0.74
0.72
0.70
0.68
Train Answer Type Accuracy
Validation Answer Type Accuracy
0
20
40
60
Epochs

Vit-L/14@336px:



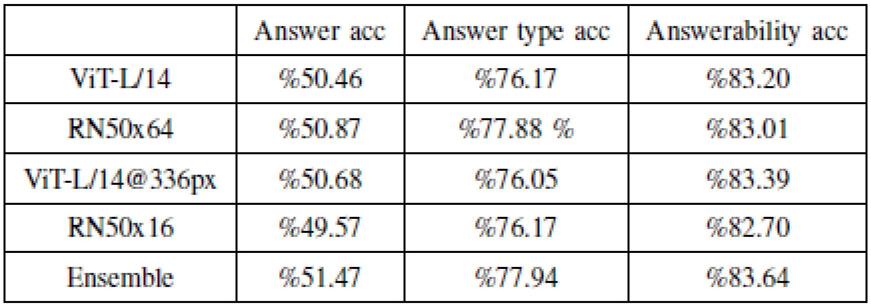
RN50x16:



The graphs look similar with minor differences in the answer type loss. It seems from these results that answer type validation results were much closer to the training results compared to the answers. Also surprisingly, the answer type validation accuracy was higher than the train accuracy. However, this can be explained by the dropout that is not used during validation. On the other hand, the answer seems to be overfitting to the training data. This was mitigated by dropout but it's still very significant. This can be explained through the nature of the problem. Classification of text strings is not straightforward, and it would make a lot of sense that the model would highly fit the training data but wouldn't do so well when presented with answers that aren't in its vocabulary set. This can be solved by increasing the train set and using more diverse examples that would improve the vocabulary of the model.

Ensemble:

Finally, we created an ensemble out of all 4 models to produce our final output. We would train each model individually, save the epoch with the highest validation accuracy, then load the 4 best models later for evaluation. We then fed each model its respective encoding of the same image/question pair then took the mean of the 4 outputs. That produced the following results:

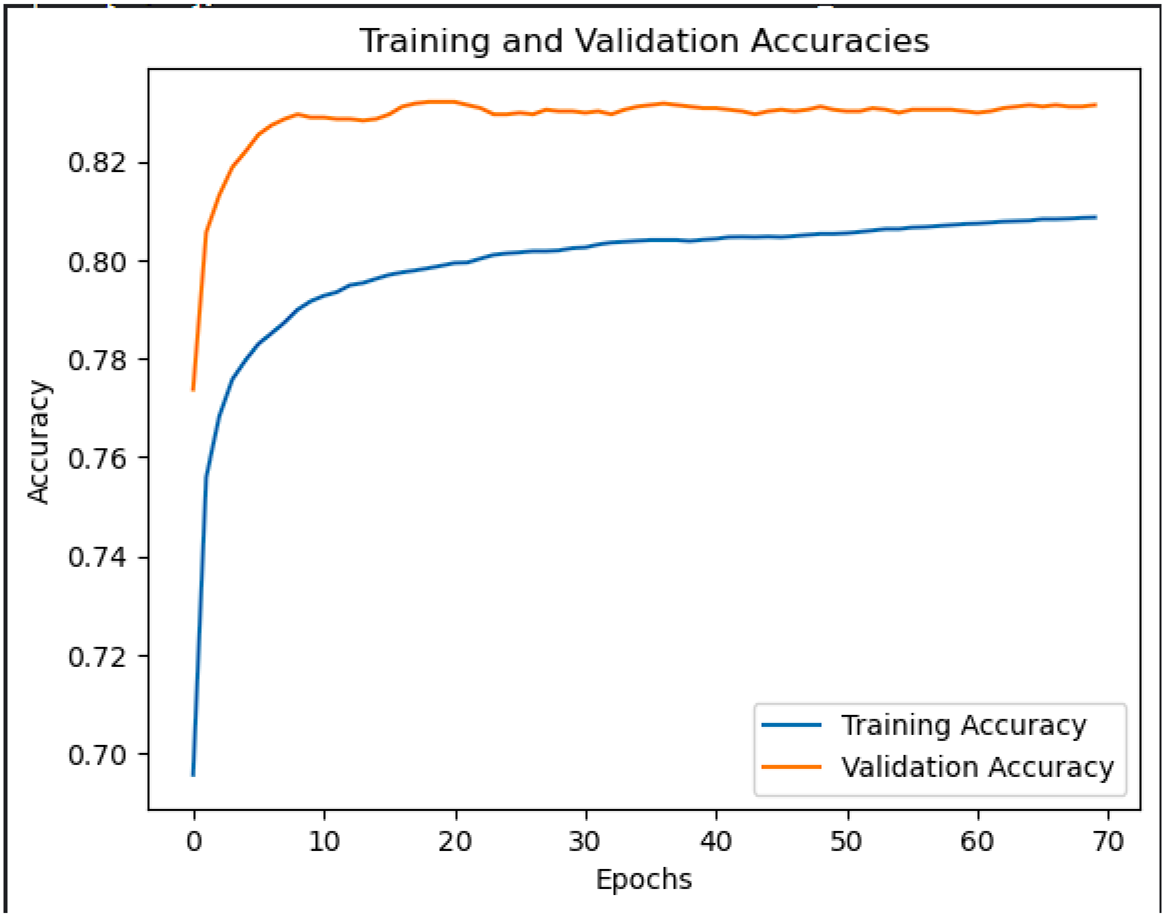


	Answer acc	Answer type acc	Answerability acc
ViT-L/14	%50.46	%76.17	%83.20
RN50x64	%50.87	%77.88 %	%83.01
ViT-L/14@336px	%50.68	%76.05	%83.39
RN50x16	%49.57	%76.17	%82.70
Ensemble	%51.47	%77.94	%83.64

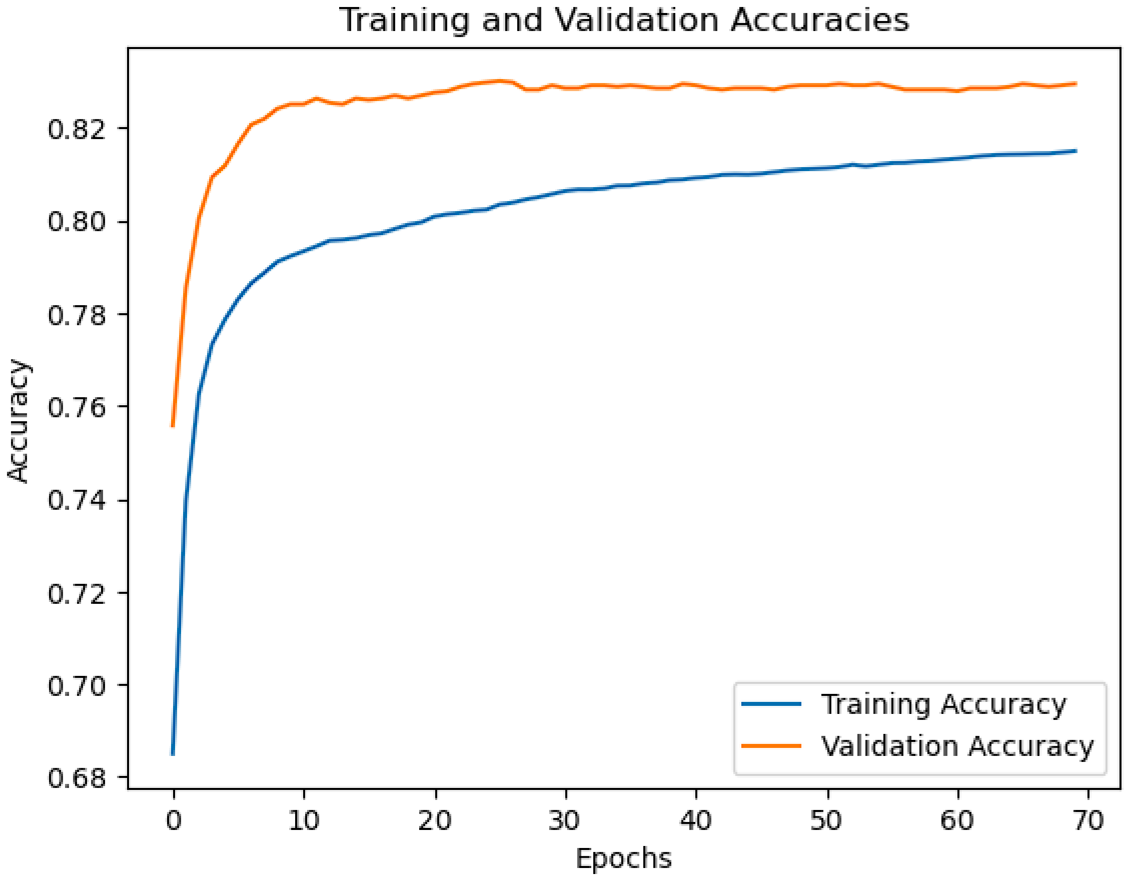
Answerability:

This task was very simple. We created a simple classifier that mapped the embedding dimension to 1 and used a sigmoid activation function. We then repeated all the steps in the above model to produce the following graphs: (results of answerability are shown in the table above)

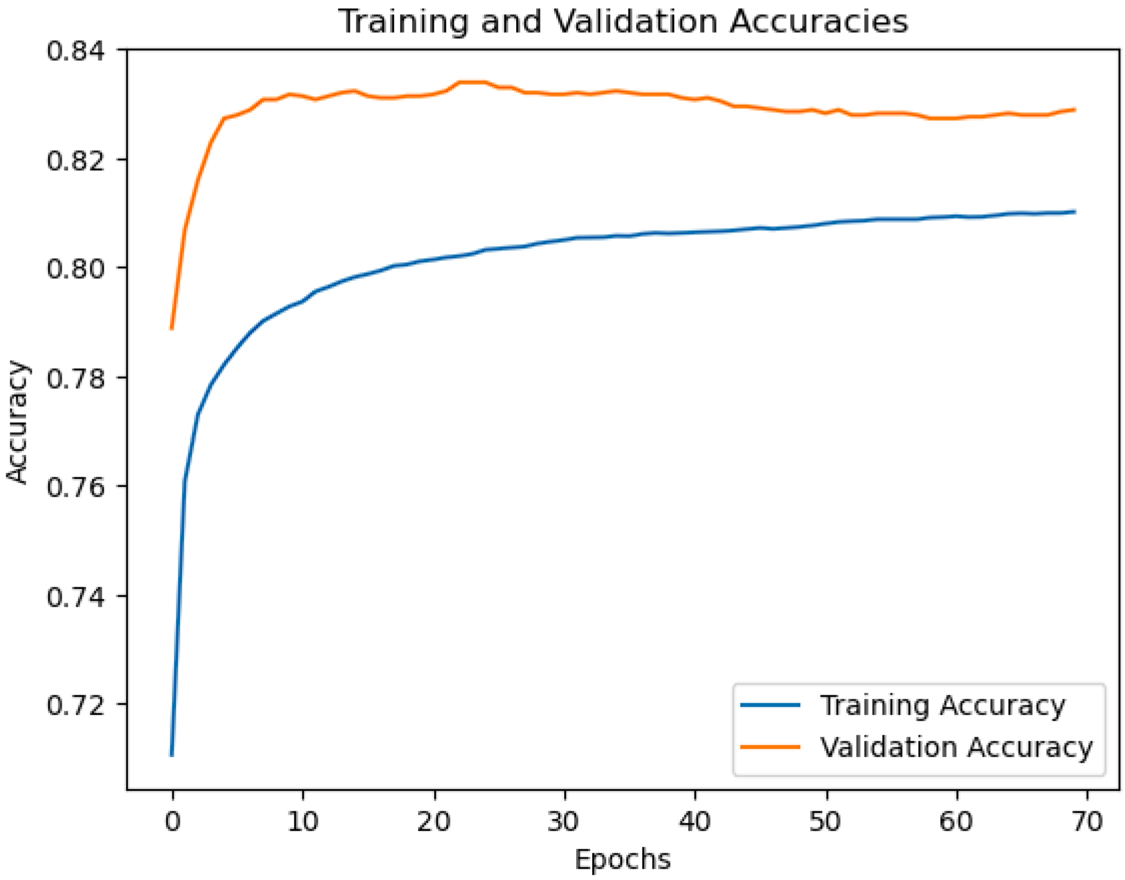
ViT-L/14:



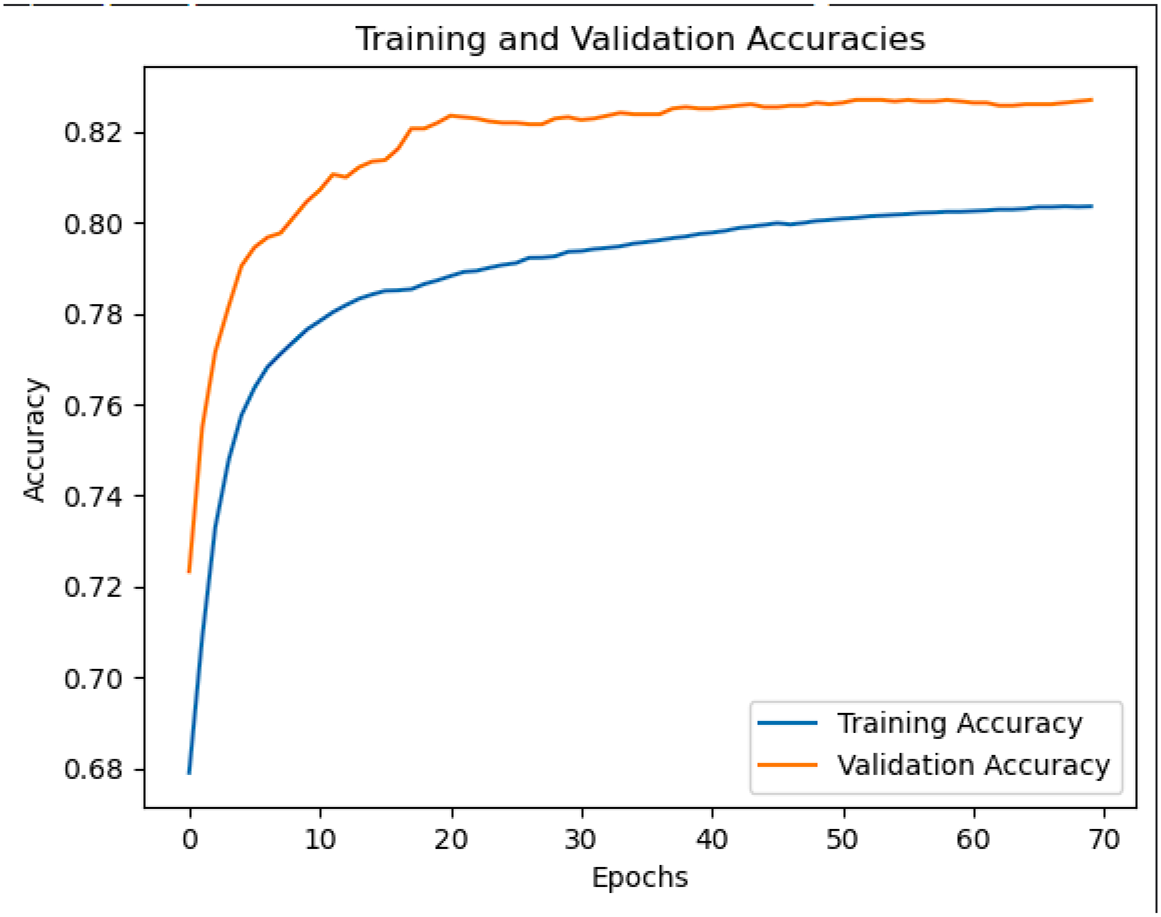
RN50x64:


Training and Validation Accuracies
Accuracy
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.68
0
10
20
30
40
50
60
70
Epochs
Training Accuracy
Validation Accuracy

Vit-L/14@336px:



RN50x16:



We decidede to use ensemble as our final model as it produced the best validation accuracies and thus our evaluation on the test set were: %51.6 in answers, %76.2 in answer types, and %82.60 in answerability.