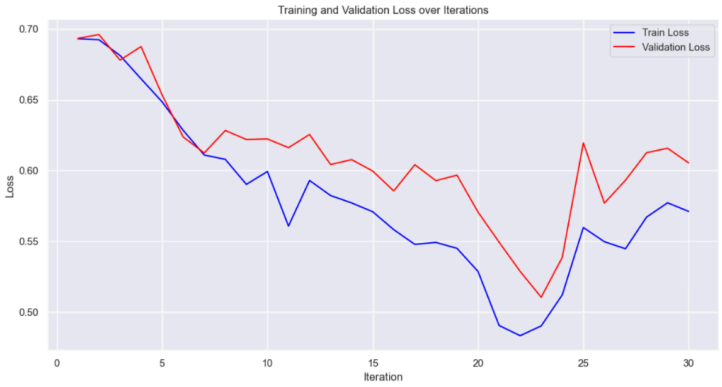
Representation Learning
Assignment 1 - Report

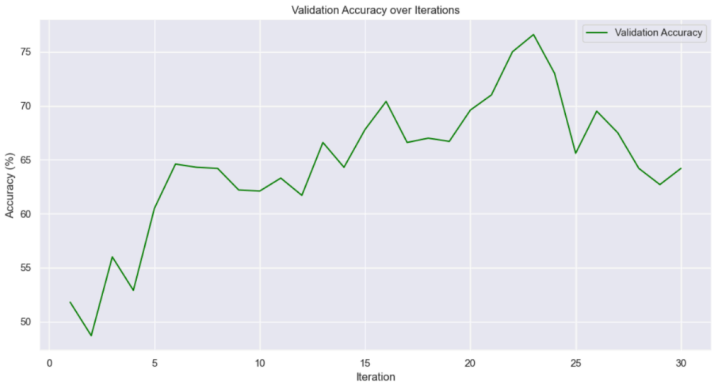
Student Name: **Maryam Alipourhajiagha**

Question 3:

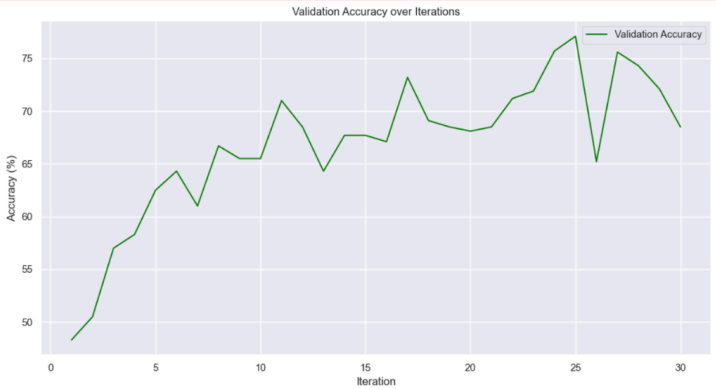
Q.3.1: Learning Curves

Setting 1: LSTM, no dropout, encoder only

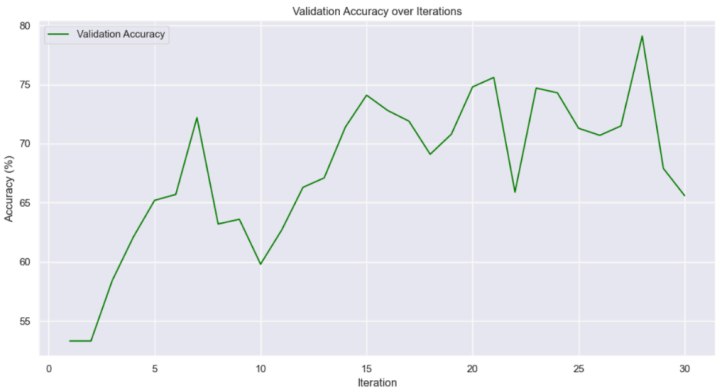




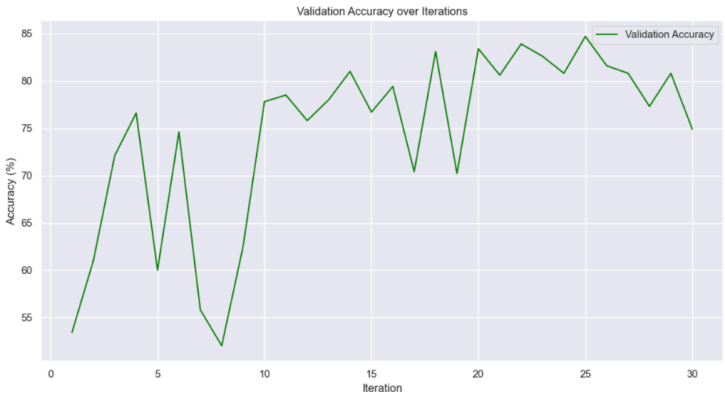
Setting 2: LSTM, dropout, encoder only



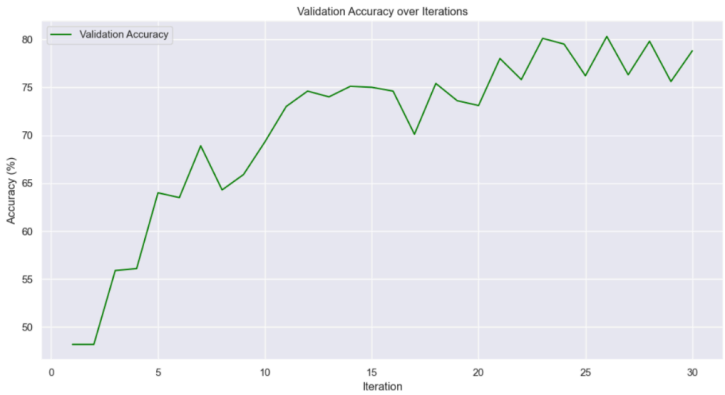
Setting 3: LSTM, dropout, encoder-decoder, no attention



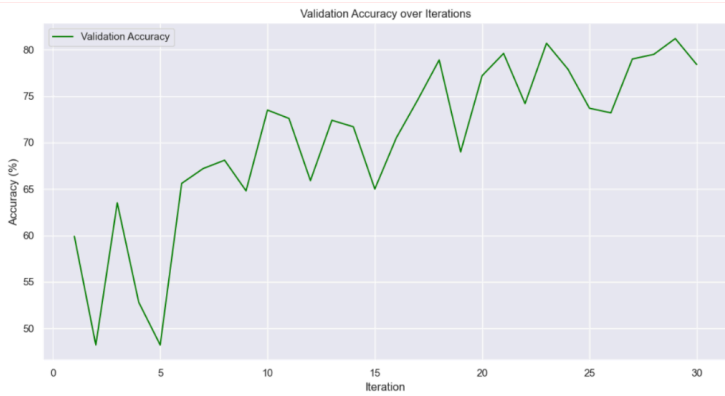
Setting 4: LSTM, dropout, encoder-decoder, with attention



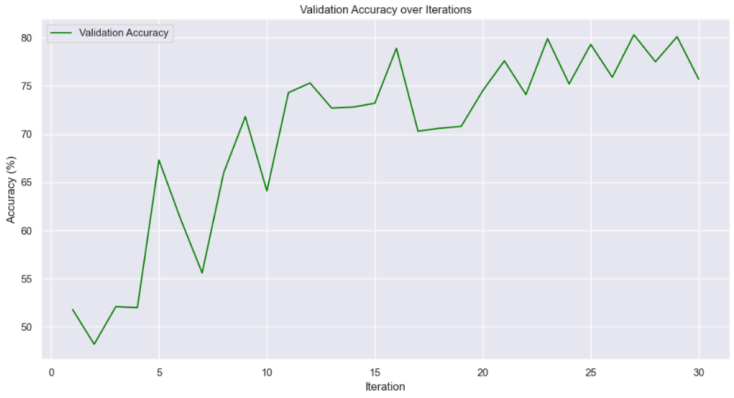
Setting 5: Transformer, 2 layers, pre-normalization



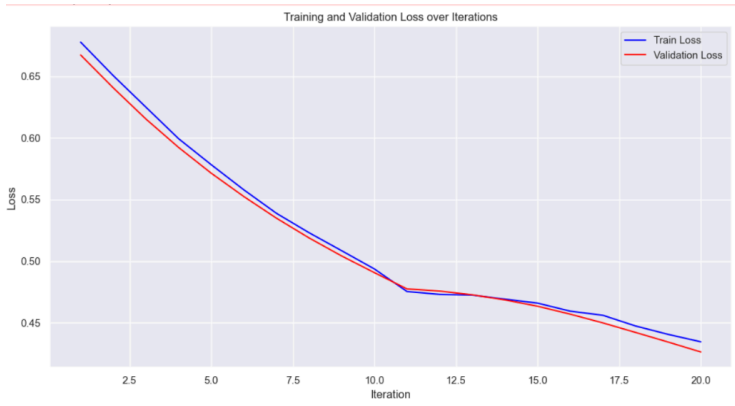
Setting 6: Transformer, 4 layers, pre-normalization

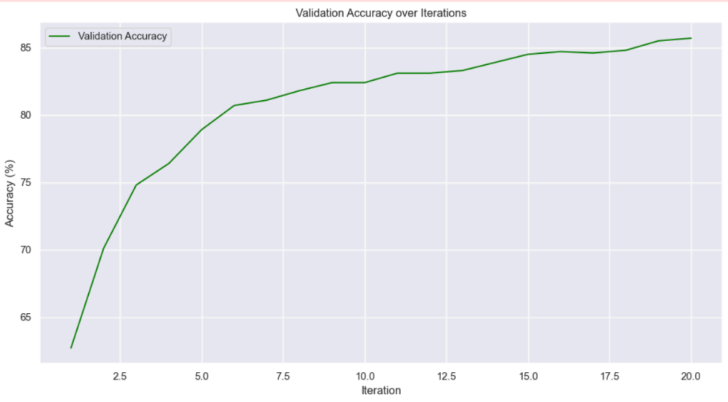


Setting 7: Transformer, 2 layers, post-normalization



Setting 8: Fine-tuning BERT





Q.3.2:

	Model Architecture	Train Time (s)	Eval Time (s)	Last Eval Loss (%)	Last Train Loss(%)	Best Val Acc (%)	Val F1
1	LSTM, no dropout, encoder only	440.83	0.65	0.593365	0.536789	69.8	0.67
2	LSTM, dropout, encoder only	440.52	0.64	0.530894	0.517814	68.1	0.71
3	LSTM, dropout, encoder-decoder, no attention	486.13	0.68	0.667056	0.522535	68.0	0.70
4	LSTM, dropout, encoder-decoder, attention	705.05	0.79	0.367904	0.414917	**86.6**	**0.86**
5	Transformer, 2 layers, pre-normalization	737.60	0.82	0.620144	0.452944	78.2	0.74
6	Transformer, 4 layers, pre-normalization	1123.01	1.05	0.410303	0.487624	82.5	0.82
7	Transformer, 2 layers, post-normalization	731.20	0.80	0.410506	0.476204	81.3	0.82
8	Fine-tuning BERT	3607.38	6.12	0.418001	0.456604	86.4	**0.86**

Table: Summary of training and validation performance for different model architectures. The table includes the **total duration of training (Train Time) and total duration of evaluation (Eval Time) in seconds across epochs**, as well as **best validation accuracy across 3 epochs (Best Val Acc)**, and F1 score. I also included the **last epoch loss in time of**

training and validation in the two middle columns. The best validation accuracy is highlighted in bold.

Q.3.3:

If we care most about **saving time**, we can choose **Setting 2** (LSTM with dropout, encoder only). It's the fastest, with a training time of 440.52 seconds and an evaluation time of 0.64 seconds. Since wall-clock time for each experiment setting is the sum of its training time and evaluation time. If we care most about getting the best results, we can choose **Setting 4** (LSTM with dropout, encoder-decoder, and attention). It has the **highest validation accuracy of 86.6%**. Additionally, Setting 4 has the lowest last evaluation loss (0.367904%), suggesting that by the end of training, this model has achieved the best fit to the validation data without overfitting, as evidenced by the relatively low last training loss (0.414917%) as well.

Q.3.4:

Comparing Setting 1 (LSTM, no dropout) and Setting 2 (LSTM, dropout), we observe:

- **Wall Clock Time:** The times are virtually identical, with a minuscule decrease in Setting 2.
- **Best Evaluation Accuracy:** There is a slight decrease in accuracy with dropout, from 69.8% in Setting 1 to 68.1% in Setting 2.
- **Validation F1 Score:** There is an improvement in the F1 score with dropout, increasing from 0.67 in Setting 1 to 0.71 in Setting 2.
- **Last Evaluation Loss:** The last evaluation loss decreased with dropout, from 0.593365% in Setting 1 to 0.530894% in Setting 2.
- **Last Training Loss:** The last training loss is nearly unchanged, with a marginal increase from 0.536789% in Setting 1 to 0.517814% in Setting 2.

Despite a slight decrease in the best evaluation accuracy, the introduction of dropout in Setting 2 resulted in an improved F1 score, which indicates a better balance of precision and recall. Additionally, a small reduction in the last evaluation loss suggests improved model generalization. The training and evaluation times remained largely unaffected by the introduction of dropout. Therefore, even with a slight drop in accuracy, the benefits of dropout in enhancing the model's generalization and predictive quality, as reflected by the F1 score and evaluation loss, are evident. But on the other hand we should consider that the drop in validation accuracy after introducing dropout in Setting 2 could be due to

several factors. Over-regularization from a high dropout rate can lead to underfitting, reducing the model's ability to capture the data's patterns. Additionally, if the task is simple or the model has the right capacity, adding dropout might not provide benefits and could even harm performance. Training dynamics also change with dropout, potentially requiring adjustments in other hyperparameters. Finally, the randomness introduced by dropout can impact training outcomes.

Q.3.5:

Comparing Setting 2 (LSTM, dropout, encoder only) and Setting 3 (LSTM, dropout, encoder-decoder, no attention), we observe:

- **Training Time:** Moving from an encoder only to an encoder-decoder setup, the training time increases from 440.52 to 486.13 seconds, which can be attributed to the additional parameters and the complexity that the decoder introduces.
- **Best Evaluation Accuracy:** Accuracy slightly decreases from 68.1% to 68.0%. For semantic analysis, where the task is often to predict the sentiment or meaning of a single piece of text rather than generating new text, the encoder-only model might capture the necessary features without the need for a decoder.
- **Validation F1 Score:** A slight decrease from 0.71 to 0.70, suggesting that the encoder-decoder model does not offer a substantial advantage in terms of precision and recall over the encoder-only model in this context.
- **Last Evaluation Loss:** The encoder-decoder setup shows increased last evaluation loss (0.667056%) compared to the encoder only (0.580894%). This indicates that by the end of training, the encoder-only model was more efficient in generalizing to the validation data.
- **Last Training Loss:** There is a decrease in the last training loss when the decoder is added (from 0.537814% to 0.522535%), suggesting that the encoder-decoder model might be capturing the training data's nuances slightly better.

In semantic analysis tasks, the goal is usually to understand the meaning or sentiment of the text rather than to generate new text sequences. An encoder-only architecture might be sufficient for capturing the semantic representation of the input data. The addition of a decoder is often more beneficial in tasks that require the generation of new sequences based on the input, such as translation or summarization.

In this case, the slight improvement in training loss with the encoder-decoder model does not translate into improved validation performance, as seen in the best evaluation accuracy

and validation F1 score. This could mean that for semantic analysis, the additional complexity of an encoder-decoder structure may not be necessary and does not confer a significant benefit over a simpler encoder-only model. The encoder-only model appears to be more efficient and nearly as effective, if not more so, in generalizing from the training to the validation data.

Q.3.6:

Comparing Setting 3 (LSTM, dropout, encoder-decoder, no attention) and Setting 4 (LSTM, dropout, encoder-decoder, attention), we observe:

- **Training Time:** The training time increased significantly from 486.13 seconds to 705.05 seconds. The self-attention mechanism adds complexity and more parameters to be learned to the model, which results in longer training times.
- **Best Evaluation Accuracy:** There's a marked improvement in accuracy, with Setting 4 achieving a best evaluation accuracy of 86.6% compared to Setting 3's 68.0%.
- **Validation F1 Score:** The F1 score improved greatly from 0.70 to 0.86, suggesting that the attention mechanism contributed to a much better balance between precision and recall, which is crucial in a task like semantic analysis.

The attention mechanism likely enhanced the model's performance by focusing on the most relevant words and phrases that contribute to the overall meaning or sentiment of a sentence, even if they are separated by several other words. This is crucial in semantic analysis, where the relationship between words can significantly alter the interpretation. Furthermore, the ability of the attention mechanism to capture long-range dependencies helps in recognizing subtle cues that can be critical for accurately determining sentiment or semantic meaning of the text.

Q.3.7:

Comparing experiments 5, 6, 7 and 8 we observe following results:

- **Settings 5 & 7** (Transformer with 2 layers, pre- and post-normalization respectively) have moderate training times and evaluation times, with best evaluation accuracies of 78.2% and 81.3%, and validation F1 scores of 0.74 and

0.82. The performance here is good, though perhaps not as high as one might expect from the Transformer's reputation. This could be due to the limited number of layers used, which might restrict the model's ability to capture more complex representations.

- **Setting 6** (Transformer with 4 layers, pre-normalization) shows improved performance over the 2-layer configurations with a best evaluation accuracy of 82.5% and an F1 score of 0.82, but at the cost of a much longer training time. The additional layers may allow for better learning of data complexities.
- **Setting 8** (Fine-tuning BERT, which is a Transformer-based model) results in the highest evaluation accuracy and F1 score tied with Setting 4, indicating very strong performance. However, it also has by far the longest training time and the longest evaluation time, which is a common trade-off with larger pre-trained models like BERT that contain significantly more parameters.

Given the strong performance typically associated with Transformers, the results for Settings 5, 6, and 7 might initially seem a bit underwhelming. However, they are still competitive.

In my opinion, in our task of classifying Amazon reviews as 1 (good reviews) and 0 (bad reviews), we can leverage the potential of LSTMs with an attention mechanism to the extent that this setting can outperform transformer-based models in other settings. This is because we do not need to capture long-range dependencies to the same extent as we would in tasks that involve generating a target sequence that needs to closely match the input sequence.

Q.3.8:

The GPU usage reported in the table below reflects a steady-state estimate, as it was consistently measured from the point where usage stabilized until the end of training.

Setting	System RAM	GPU RAM	Disk
1	2.9 / 83.5 GB	4.1 / 40.0 GB	29.7 / 201.2 GB
2	3.1 / 83.5 GB	4.1 / 40.0 GB	29.7 / 201.2 GB
3	3.1 / 83.5 GB	4.1 / 40.0 GB	29.7 / 201.2 GB
4	3.3 / 83.5 GB	5.4 / 40.0 GB	29.7 / 201.2 GB
5	3.3 / 83.5 GB	6.8 / 40.0 GB	29.7 / 201.2 GB
6	3.4 / 83.5 GB	11.6 / 40.0 GB	29.7 / 201.2 GB
7	3.4 / 83.5 GB	11.6 / 40.0 GB	29.7 / 201.2 GB
8	3.6 / 83.5 GB	17.6 / 40.0 GB	30.2 / 201.2 GB

I've used Google Colab Pro to train my models and here is GPU memory usage analysis:

- **Setting 1** (LSTM, no dropout, encoder only) consumes GPU memory at 4.1 GB, as expected due to its simple architecture without any additional complexity like dropout or attention layers.
- **Setting 2** (LSTM, dropout, encoder only) also uses 4.1 GB of GPU memory, indicating that introducing dropout doesn't impact the memory requirement. Dropout typically doesn't increase memory demand since it merely deactivates a subset of neurons during training.
- **Setting 3** (LSTM, dropout, encoder-decoder, no attention) maintains GPU memory usage at the same level as the first two settings. The encoder-decoder architecture in this case doesn't appear to demand additional GPU memory, possibly due to effective memory optimization.
- **Setting 4** (LSTM, dropout, encoder-decoder, attention) we see a jump in memory usage to 5.4 GB, which can be attributed to the attention mechanism that needs extra memory for calculating and storing the attention scores across the input sequence.

- **Setting 5** (Transformer, 2 layers, pre-normalization) shows increased memory usage of 6.8 GB. The self-attention within Transformers requires more memory as it processes all sequence elements simultaneously at each layer.
- **Setting 6** (Transformer, 4 layers, pre-normalization) adding two more layers of encoder increases GPU memory usage to 11.6 GB. Additional layers mean more parameters and intermediate values, thus a higher memory footprint.
- **Setting 7** (Transformer, 2 layers, post-normalization) surprisingly matches Setting 6 in GPU memory consumption, despite having fewer layers. The post-normalization technique shouldn't cause this on its own, so other training variables like batch size or sequence length might play a role.
- **Setting 8** (Fine-tuning BERT) has the highest GPU usage at 17.6 GB due to BERT's extensive architecture with millions of parameters requiring significant memory for both the pre-trained model and gradient computations during fine-tuning.

Q.3.9:

- **Setting 1:** Both curves are trending downwards, indicating that the model is learning and improving thus there is **no sign of underfitting**. The overall trend does not indicate a clear divergence between training and validation loss thus there is **no sign of overfitting.** The validation accuracy shows **some degree of instability**, as indicated by the fluctuations in accuracy. I think the instability in the learning curves could be a result of a combination of the large batch size (512) and the small learning rate (1e-5). The large batch size may cause the model to make fewer updates per iteration, each with a very small learning rate, potentially causing the model to underfit or to converge very slowly, resulting in the observed fluctuations in validation accuracy. **Addressing the instability:** Setting a high learning rate and decreasing it gradually by implementing a learning rate schedule.
- **Setting 2:** Again exactly like the previous setting there is **no clear sign of overfitting and underfitting.** However, there are spikes in the validation loss that do not correspond to similar spikes in the training loss which is **a sign of instability**. This can be due to dropout that introduces variability during training because a random subset of neurons is "dropped" during each forward pass. This can cause fluctuations in the loss and accuracy, as different sets of neurons are trained at each iteration. Perhaps by lowering the dropout rate, the fluctuations might smooth out slightly.

- **Setting 3:** Same as the previous settings.
- **Setting 4:** Same as the previous settings. However we can observe that by introduction of attention mechanism the convergence happened faster.
- **Setting 5-8:** Transformer-based models display an overarching trend of rising validation accuracy with less fluctuation than the LSTM-based models, signaling steadier training dynamics. However, the Transformer-based models, particularly in Experiment 6, do show occasional spikes in training loss, which points to sporadic training instability. The BERT model from Experiment 8 demonstrates a steady enhancement in validation accuracy and a converging training loss, suggesting a stable training process and effective learning. T**he LSTM models that lack an attention mechanism (Experiments 1-3) showed greater variability and potential instability compared to those incorporating attention mechanisms and the Transformer models.** The LSTM model enhanced with attention (Experiment 4) and the Transformer models (Experiments 5-8) seem to exhibit greater stability and a reduced tendency for overfitting. Especially, the fine-tuned BERT model benefits from a learning process that is more uniform and steady, likely due to its architectural features that efficiently capture dependencies and mitigate overfitting with self-attention strategies.

  To address overfitting, underfitting, or instability in such scenarios, one could employ suitable regularization methods, adjust and manage the learning rate with schedulers or adaptive techniques, and use cross-validation for a more accurate evaluation of model performance. Enhancing the diversity of the training dataset with data augmentation tactics can also be beneficial. When fine-tuning models like BERT, it is crucial to fine-tune the learning rate and consider initially freezing some layers to aid the training process.

Q.3.10:

I performed 9 combination of perturbations and experimented them with 3 models (1,4,7). For the last perturbation I set all the word level and character level perturbations True except word_swap_neighboring_character_swap and word_swap_random_character_substitution. I ran each perturbation for 100 times for each model to get the classification accuracy. I also included an example of the perturbed review to show what happens to the original review by adding that perturbation. The Augmented Review Accs column shows the classification accuracy of each perturbation.

The below table is the results of augmentation for Setting 1:

Perturbation	Augmented Review (Example)	True Sentiment	Augmented Review Accs (%)
1- word_swap_contract	I loved the food and the service was great!	1	100
2- word_swap_extend	I loved the food and the service was great!	1	100
3- word_swap_homoglyph_swap	I loved tհe food and the service wɑs gгeat!	1	95
4- word_swap_neighboring_character_swap	I lovde teh ofod and the serviec was great!	1	100
5- word_swap_qwerty	I ooved tge food and the servide was great!	1	**86**
6- word_swap_random_character_deletion	I loved the ood ad he service as great!	1	98
7- word_swap_random_character_insertion	I loved the lfokod and the service jwxas great!	1	92
8- word_swap_random_character_substitution	I loved the fHod ard the serdice wrs great!	1	93
Combination of (1,2,3,5,6,7)	I loved the food and the service wɑs great	1	93

This model shows a profound level of robustness in most of the perturbation except in the QWERTY-based perturbation (5th perturbation). without dropout, this model might have become too sensitive to the exact sequence of characters in the training data, reducing its ability to generalize to perturbed inputs. The QWERTY perturbation changes the keys to their neighbors on a QWERTY keyboard, which can significantly alter the word shapes and consequently the model's ability to recognize them.

The below table is the results of augmentation for Setting 4:

Perturbation	Augmented Review	True Sentiment	Augmented Review Accs (%)
1- word_swap_contract	I loved the food and the service was great!	1	100
2- word_swap_extend	I loved the food and the service was great!	1	100
3- word_swap_homoglyph_swap	I loved the food and the service was great!	1	98
4- word_swap_neighboring_character_swap	I loved the ofod and the service asw gerat!	1	97
5- word_swap_qwerty	I loved the foor and the service wad rrwat!	1	96
6- word_swap_random_character_deletion	I oved the foo an the sevice was great!	1	100
7- word_swap_random_character_insertion	I lonved the food aCnd the sertvice Uwas great!	1	95
8- word_swap_random_character_substitution	I loved thS food aTd Ahe service was yreat!	1	97
Combination of (1,2,3,5,6,7)	I love the food znd the service was great!	1	98

This setting has a high percentage of classification accuracy for all 9 perturbations. The incorporation of dropout introduces randomness during training, which can help the model generalize better by preventing it from relying too heavily on specific features of the training data. The attention mechanism allows the model to focus on different parts of the input sequence when making predictions, which can make it more resilient to perturbations that do not change the overall meaning of the text. The encoder-decoder structure can also help in reconstructing the intended meaning from a perturbed input, as it is designed to map sequences to sequences, potentially correcting or ignoring errors in the input sequence.

The below table is the results of augmentation for Setting 7:

Perturbation	Augmented Review	True Sentiment	Augmented Review Accs (%)
1- word_swap_contract	I loved the food and the service was great!	1	100
2- word_swap_extend	I loved the food and the service was great!	1	100
3- word_swap_homoglyph_swap	I I oved the food and the service was great	1	89
4- word_swap_neighboring_character_swap	I lovde the food and the serivce wsa gerat!	1	86
5- word_swap_qwerty	I lived yhe tood and the segvice was great!	1	84
6- word_swap_random_character_deletion	I loved te food and the sevice was gat!	1	86
7- word_swap_random_character_insertion	I loved the foSod and the service was grUeardt!	1	**71**
8- word_swap_random_character_substitution	I lovez the food aud hhe service fas great!	1	81
Combination of (1,2,3,5,6,7)	I loved he foor and the service was greaqt!	1	86

Even though transformers are designed to handle long-range dependencies and can be very robust to variations, however, the model did perform poorly with random character insertion. The use of post-normalization in the transformer might play a role. If the inserted characters significantly change the token distribution or introduce rare tokens, the post-normalization step might not be able to effectively stabilize the learning representations, leading to higher rate of misclassification with adding word_swap_random_character_insertion perturbation.

Q.2.5:

The total number of parameters in the MultiHeadedAttention module can be calculated as follows:

Each of the four linear layers (W_Q ,W_K , W_V , and W_O) has parameters equal to the square of the product of head_size and num_heads. Therefore, the total parameters for these layers combined is 4 times this value. Additionally, each linear layer has a bias vector, contributing $(head\_size \times num\_heads)$ parameters per layer. With four layers, the total bias parameters amount to 4 times this value. the total number of parameters in the MultiHeadedAttention module is given by:

Total Parameters = $4 \times ((head\_size \times num\_heads))^2 + (head\_size \times num\_heads))$