# Assignment 2 - Programming
# Recurrent Neural Networks, Attention, and Transformers

IFT6135-H2024, Section B
Prof : Aaron Courville

Student: Mohammadhossein Malekpour

**Table of Content:**

# Question 2

## Question 2.5

The number of learnable parameters in the *MultiHeadedAttention* module is determined by the dimensions of these linear layers and their associated bias terms as follows:

1. For each linear layer ($W\_Q$, $W\_K$, $W\_V$, and $W\_O$), the number of parameters is given by the product of the input and output dimensions, which is *(head_size * num_heads) * (head_size * num_heads)*. Since there are four such layers, the combined number of parameters for these layers is *4 * (head_size * num_heads) * (head_size * num_heads)*.

2. Each linear layer is accompanied by a bias vector, which adds an additional *(head_size * num_heads)* parameters per layer. With four layers, the total number of bias parameters is *4 * (head_size * num_heads)*.

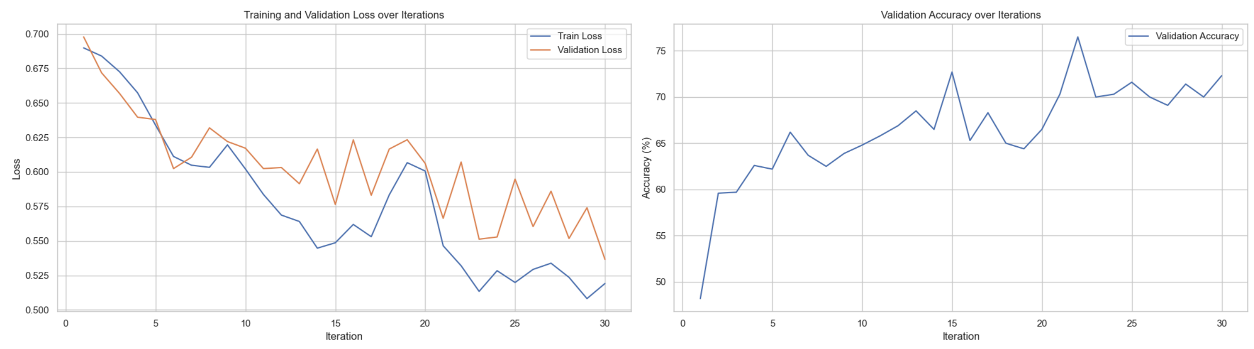
=> The overall number of learnable parameters in the *MultiHeadedAttention* module:

$$Total\,Parameters = 4 * (head\_size * num\_heads) * (head\_size * num\_heads) + 4 * (head\_size * num\_heads)$$

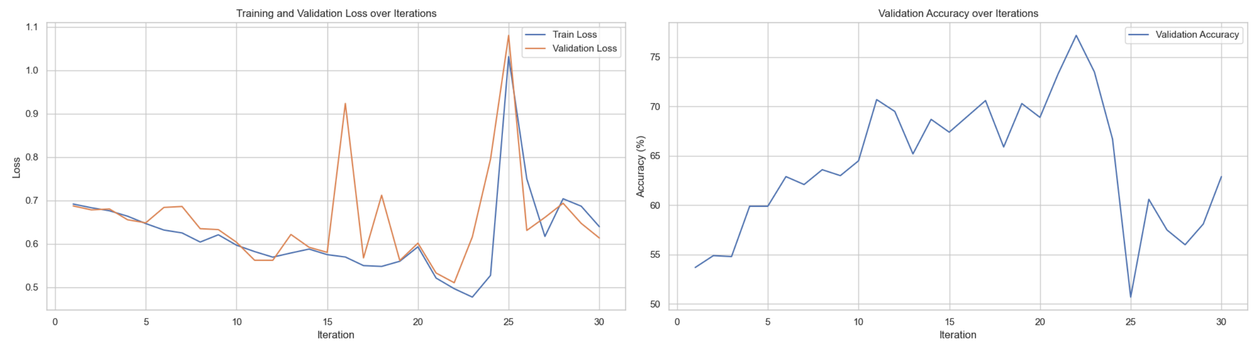
# Question 3

## Question 3.1

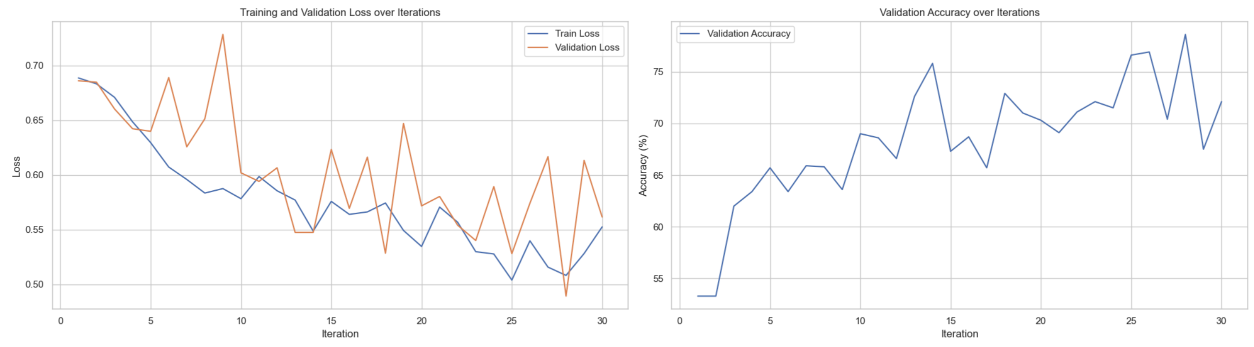
Setting 1



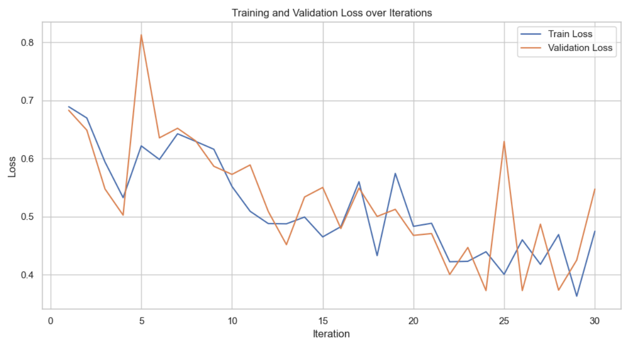
Setting 2

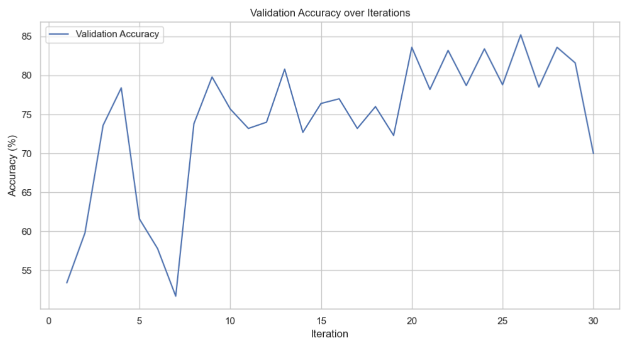


Setting 3

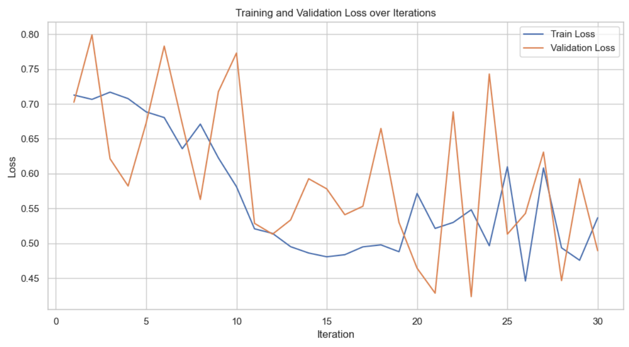


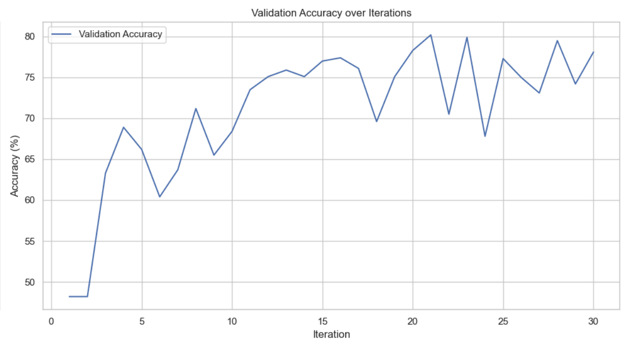
Setting 4



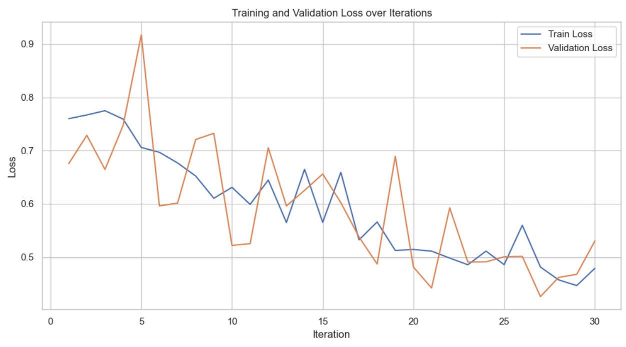


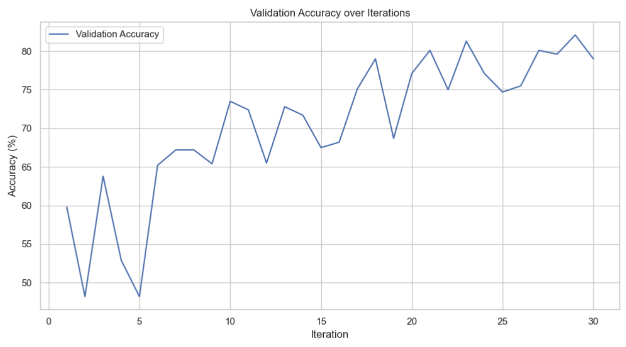
Setting 5



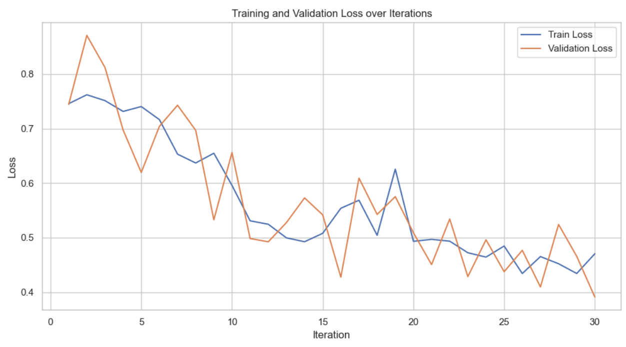


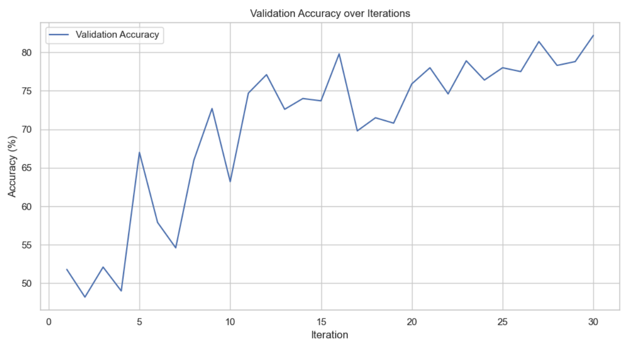
Setting 6

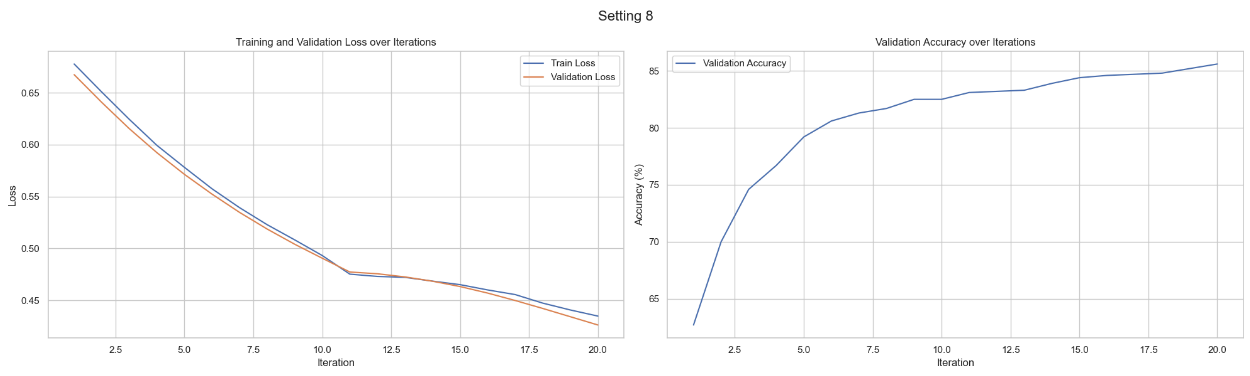




Setting 7







## Question 3.2

| Experiment | Train Time (s) | Eval Time (s) | Best Eval Acc (%) | Last Eval Acc (%) | Last Eval F1 |
|---|---|---|---|---|---|
| **Setting 1**: LSTM, no dropout, encoder only | 721.10 | 0.89 | 73.2 | 73.2 | 0.67 |
| **Setting 2**: LSTM, dropout, encoder only | 721.65 | 0.91 | 71.9 | 66.4 | 0.63 |
| **Setting 3**: LSTM, dropout, encoder-decoder, no attention | 947.89 | 1.00 | 76.4 | 76.4 | 0.75 |
| **Setting 4**: LSTM, dropout, encoder-decoder, attention | 1253.16 | 1.20 | 77.77 | 77.7 | 0.81 |
| **Setting 5**: Transformer, 2 layers, pre-normalization | 874.57 | 0.95 | 78.3 | 78.3 | 0.74 |
| **Setting 6**: Transformer, 4 layers, pre-normalization | 1361.86 | 1.25 | 82.7 | 82.7 | 0.83 |
| **Setting 7**:Transformer, 2 layers, post-normalization | 882.60 | 0.94 | 76.0 | 76.0 | 0.79 |
| **Setting 8**: Fine-tuning BERT | 4877.26 | 8.29 | **86.4** | **86.4** | **0.85** |

**Train Time:** The total time taken for training the model across all epochs.
**Eval Time:** The total time taken for evaluating the model across all epochs.
**Best Eval Acc:** The highest accuracy achieved by the model on the evaluation dataset across all epochs.
**Last Eval Acc:** The accuracy of the model on the evaluation dataset at the final epoch.
**Last Eval F1:** The F1 score of the model on the evaluation dataset at the final epoch.

* For the following analysis, I used Last Eval Acc

## Question 3.3

If I were most concerned with wall clock time, I would choose **Setting 1**. This setting has the shortest total training time (721.10 seconds) and a relatively short evaluation time (0.89 seconds). It uses a simple LSTM architecture without dropout and only an encoder, which contributes to its faster training and evaluation times.

If I were most concerned with generalization performance, I would choose **Setting 8**. This setting has the highest evaluation accuracy (86.4%) and F1 score (0.86) among all the configurations. It involves fine-tuning BERT, which is a powerful pre-trained transformer model. However, it's worth noting that this setting also has the longest training and evaluation times by a significant margin, so there is a trade-off.

## Question 3.4

Both experiments 1 and 2 have similar training times, with Setting 1 at 721.10 seconds and Setting 2 at 721.65 seconds. The addition of dropout does not seem to have a significant impact on the training time. The evaluation times are also very similar, with Setting 1 at 0.89 seconds and Setting 2 at 0.91 seconds. Setting 2 has a lower evaluation accuracy (66.4%) compared to Setting 1 (73.2%). This could indicate that the dropout regularization might be causing some loss of information or over-regularization in this particular case. The F1 score is lower for Setting 2 (0.63) than for Setting 1 (0.67), suggesting that the model's overall performance has decreased with the addition of dropout. From these results, we can infer that the impact of having dropout in this case was negative, leading to a decrease in both evaluation accuracy and F1 score. In this case, it's possible that the model was not overfitting to begin with, so the addition of dropout may have introduced unnecessary noise or complexity that hindered the model's performance.

## Question 3.5

Both experiments 2 and 3 have similar training times, with Setting 2 at 721.65 seconds and Setting 3 at 947.89 seconds. The slight increase in training time for Setting 3 is likely due to the additional complexity introduced by the decoder in the encoder-decoder architecture. Setting 3 has a significantly higher evaluation accuracy (76.4%) compared to Setting 2 (66.4%). Similarly, the F1 score is higher for Setting 3 (0.75) than for Setting 2 (0.63), indicating better overall performance. From these results, we can infer that there are benefits to using an encoder-decoder network instead of just an encoder for this particular task. The encoder-decoder architecture allows the model to better capture the relationship between the input and output sequences, which can be particularly important for tasks that require a mapping between sequences of different lengths or complexities. The decoder can generate output sequences based on the context provided by the encoder, leading to more accurate and coherent results.

## Question 3.6

The training time increases significantly from Setting 3 (947.89 seconds) to Setting 4 (1253.16 seconds). This increase is due to the added complexity of incorporating self-attention into the LSTM encoder-decoder architecture. Similarly, the evaluation time increases from Setting 3 to Setting 4. Setting

4 shows an improvement in evaluation accuracy, increasing from 76.4% in Setting 3 to 77.7% in Setting 4. The F1 score also improves with the addition of self-attention, going from 0.75 in Setting 3 to 0.81 in Setting 4. From these results, we can infer that training the LSTM network with self-attention had a positive impact on the model's performance, as evidenced by the improvements in evaluation accuracy and F1 score. The self-attention mechanism allows the model to weigh different parts of the input sequence differently, enabling it to capture long-range dependencies and contextual information more effectively.

## Question 3.7

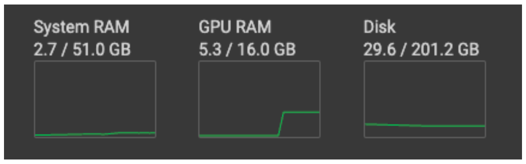
The results from experiments 5, 6, 7, and 8 show a general trend of increasing evaluation accuracy and F1 score with the expectation that more complex Transformer-based models, with more layers or pre-training, tend to perform better on natural language processing tasks.

In Setting 5 the model achieves an evaluation accuracy of 78.3% and an F1 score of 0.74. This is a respectable performance for a relatively simple Transformer model with only 2 layers. The increase in the number of layers (Setting 6) leads to improved performance, with an evaluation accuracy of 82.7% and an F1 score of 0.83. This suggests that the additional layers help the model capture more complex relationships in the data. In Setting 7 the performance is slightly lower than in Setting 6 but still competitive, with an evaluation accuracy of 76.0% and an F1 score of 0.79. The difference between pre-normalization and post-normalization does not seem to have a major impact on the overall performance. As expected, Setting 8 (fine-tuning BERT) results in the best performance among all settings, with an evaluation accuracy of 86.4% and an F1 score of 0.86. BERT's pre-training on a large corpus allows it to capture a wide range of linguistic features, which contributes to its strong performance when fine-tuned on specific tasks.

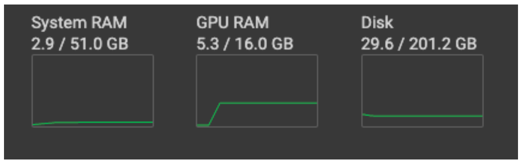
The results are in line with expectations, as Transformer-based models, especially those with pre-training like BERT, have been shown to achieve state-of-the-art performance on a variety of nlp tasks. The success of these models can be attributed to their ability to capture long-range dependencies and contextual information through self-attention mechanisms.

## Question 3.8

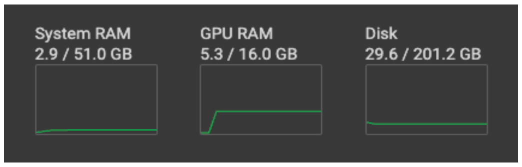
1. Setting 1: The memory usage is 5.3/16 GB, which is relatively low. This is expected as the model is a simple LSTM with no additional components like attention mechanisms.



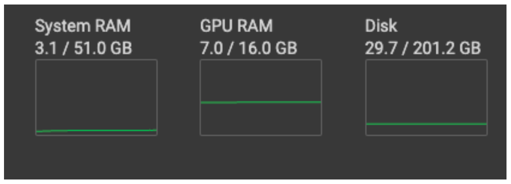
2. Setting 2: The memory usage remains the same as Setting 1 at 5.3/16 GB. This indicates that the addition of dropout does not impact the GPU memory footprint for this model architecture.



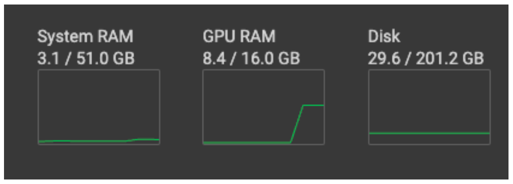
3. Setting 3: The memory usage is still 5.3/16 GB, which is surprising given that the encoder-decoder architecture is generally more complex than an encoder-only model. However, it seems that the LSTM-based encoder-decoder without attention does not substantially increase the memory requirements in this case.



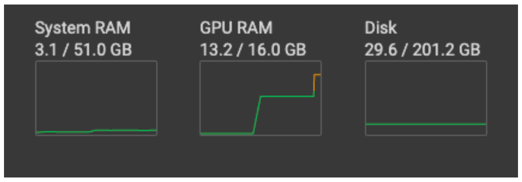
4. Setting 4: The memory usage increases to 7.0/16 GB. The addition of the attention mechanism is likely responsible for this increase in memory consumption. Attention mechanisms require additional memory to store the attention weights and to perform the weighted sum calculations.



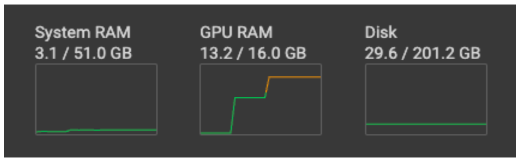
5. Setting 5: The memory usage jumps to 8.4/16 GB. Transformers are known to be more memory-intensive than LSTMs due to their self-attention mechanisms and the multiple heads in each attention layer. Even with only 2 layers, the memory footprint is larger than the LSTM-based models.



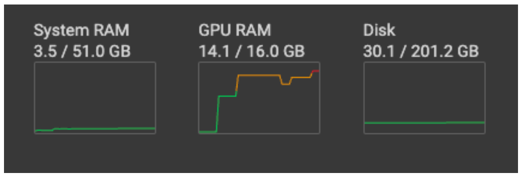
6. Setting 6: The memory usage further increases to 13.2/16 GB. Doubling the number of layers in the Transformer model significantly increases the memory requirements due to the additional parameters and intermediate computations in each layer. (Question 2.5)



7. Setting 7: The memory usage is the same as Setting 6 at 13.2/16 GB. This suggests that the choice of pre-normalization or post-normalization does not have an impact on the memory footprint of the Transformer model.



8. Setting 8: The memory usage is 14/16 GB, which is the highest among all settings. Fine-tuning BERT involves working with a pre-trained model that already has a large number of parameters, leading to a higher memory consumption. Additionally, BERT is a Transformer-based model, which further contributes to its substantial memory requirements.



## Question 3.9

As you can see in Question 3.1's plots, in Settings 1 through 7, the learning curves display a decrease in loss over iterations without a gap between the training and validation loss, suggesting no significant overfitting. However, the presence of fluctuations in Setting 1-7, especially in Setting 2 where a drastic jump in the learning curve and a steep drop in accuracy were noted at iteration 25 indicates instability. This volatility could be attributed to several factors:

- In Setting 2, which uses dropout, the erratic movement could be due to the stochastic nature of dropout itself or an inappropriate dropout rate that disrupts the model's learning intermittently.
- For LSTM-based models (Settings 1 to 4), the instability could also be a result of the inherent sensitivity of LSTMs to learning rate settings and batch variance.
- The Transformer models (Settings 5 to 7), while generally stable due to their self-attention mechanism, may still experience fluctuations if the learning rate isn't tuned properly or if the model's capacity is not aligned with the complexity of the task.

Setting 8 exhibits a stable learning curve, which can be ascribed to BERT's extensive pre-training that aids in generalizing well across different datasets and tasks, providing resilience against the fluctuations seen in the other models.
To handle overfitting or underfitting and to ensure stability across these diverse architectures, one might:

- For LSTM settings (1-4), consider implementing gradient clipping to handle exploding gradients, or tweak the learning rate and use a learning rate scheduler for smoother convergence.
- For Transformer settings (5-7), ensure that the learning rate is warm-up sufficiently and use layer normalization to stabilize training.
- In all settings, shuffling the data before each epoch can help to maintain independence and identical distribution (i.i.d) across batches.
- Regularization techniques like dropout (for LSTMs) and attention dropout or layer dropout (for Transformers) should be fine-tuned to balance model complexity and training data variance.

## Question 3.10

To assess the robustness of different models, I experimented with nine distinct text augmentations on three models: the best-performing LSTM model (Setting 4), the best Transformer model (Setting 7), and BERT (Setting 8). The augmentations applied were:

1. Word Swap Contract: Simulating contractions in the text.
2. Word Swap Extend: Extending contractions to their full forms.
3. Word Swap Homoglyph Swap: Swapping characters with visually similar ones.
4. Word Swap Neighboring Character Swap: Testing the model's handling of typos with adjacent character swaps.
5. Word Swap QWERTY: Mimicking common typing errors based on the QWERTY keyboard layout.
6. Word Swap Random Character Deletion: Removing random characters from words.
7. Word Swap Random Character Insertion: Inserting random characters into words.

8. Word Swap Random Character Substitution: Substituting characters in words with other random characters.
9. Composite Perturbation: Combining multiple augmentations to apply a complex transformation.

Here's the output of the code and after that I'm gonna analyze the result:

```
-------------------- Setting 4 --------------------
Original Review: I loved the food and the service was great!
Actual Sentiment: 1
1. Augmented review by ['word_swap_contract']: I loved the food and the service was great!
Predicted Sentiment: 1
2. Augmented review by ['word_swap_extend']: I loved the food and the service was great!
Predicted Sentiment: 1
3. Augmented review by ['word_swap_homoglyph_swap']: I loved the food ɑnd the service was gгeat!
Predicted Sentiment: 0
4. Augmented review by ['word_swap_neighboring_character_swap']: I loved the food and teh sevrice was great!
Predicted Sentiment: 1
5. Augmented review by ['word_swap_qwerty']: I loved tge fiod amd fhe service was great!
Predicted Sentiment: 1
6. Augmented review by ['word_swap_random_character_deletion']: I lvd the food and te service was geat!
Predicted Sentiment: 0
7. Augmented review by ['word_swap_random_character_insertion']: I lpoved the food aznd the serbvice was grkeat!
Predicted Sentiment: 1
8. Augmented review by ['word_swap_random_character_substitution']: I lozed the food anh the service Zas great!
Predicted Sentiment: 1
9. Augmented review by ['word_swap_contract', 'word_swap_extend', 'word_swap_homoglyph_swap', 'word_swap_neighboring_character_swap',
'word_swap_qwerty', 'word_swap_random_character_deletion', 'word_swap_random_character_insertion', 'word_swap_random_character_substitution']: I loved
the ofod aud the serviye was great!
Predicted Sentiment: 1
-------------------- Setting 6 --------------------
Original Review: I loved the food and the service was great!
Actual Sentiment: 1
1. Augmented review by ['word_swap_contract']: I loved the food and the service was great!
Predicted Sentiment: 1
2. Augmented review by ['word_swap_extend']: I loved the food and the service was great!
Predicted Sentiment: 1
3. Augmented review by ['word_swap_homoglyph_swap']: I loved the food aнd the service was gгeat!
Predicted Sentiment: 0
4. Augmented review by ['word_swap_neighboring_character_swap']: I lovde teh ofod and the service wsa great!
Predicted Sentiment: 1
5. Augmented review by ['word_swap_qwerty']: I loved the food anc rhe servics was grsat!
Predicted Sentiment: 1
6. Augmented review by ['word_swap_random_character_deletion']: I oved th food and the serice wa great!
Predicted Sentiment: 1
7. Augmented review by ['word_swap_random_character_insertion']: I loved the food uand tehe service wiOas great!
Predicted Sentiment: 1
8. Augmented review by ['word_swap_random_character_substitution']: I Koved thm fooh and the service was gceat!
Predicted Sentiment: 0
9. Augmented review by ['word_swap_contract', 'word_swap_extend', 'word_swap_homoglyph_swap', 'word_swap_neighboring_character_swap',
'word_swap_qwerty', 'word_swap_random_character_deletion', 'word_swap_random_character_insertion', 'word_swap_random_character_substitution']: I loved
the foodd and teh service was great!
Predicted Sentiment: 1
-------------------- Setting 8 --------------------
Original Review: I loved the food and the service was great!
Actual Sentiment: 1
1. Augmented review by ['word_swap_contract']: I loved the food and the service was great!
Predicted Sentiment: 1
2. Augmented review by ['word_swap_extend']: I loved the food and the service was great!
Predicted Sentiment: 1
3. Augmented review by ['word_swap_homoglyph_swap']: I loved the food and the service was gгeat!
Predicted Sentiment: 1
4. Augmented review by ['word_swap_neighboring_character_swap']: I lovde the food nad the servcie was gerat!
Predicted Sentiment: 1
5. Augmented review by ['word_swap_qwerty']: I loved the food and fye sdrvice was grfat!
Predicted Sentiment: 1
```

```
6. Augmented review by ['word_swap_random_character_deletion']: I oved the fod and the service a great!
Predicted Sentiment: 1
7. Augmented review by ['word_swap_random_character_insertion']: I loveId the fooLd Qand the service waes great!
Predicted Sentiment: 1
8. Augmented review by ['word_swap_random_character_substitution']: I loved the food and tho service wUs gwEat!
Predicted Sentiment: 1
9. Augmented review by ['word_swap_contract', 'word_swap_extend', 'word_swap_homoglyph_swap', 'word_swap_neighboring_character_swap',
'word_swap_qwerty', 'word_swap_random_character_deletion', 'word_swap_random_character_insertion', 'word_swap_random_character_substitution']: I loved
the food ɑnd 𝚝he sevrice eas great!
Predicted Sentiment: 1
```

**Setting 4 (LSTM with attention):** The model maintained performance with simple typos (neighboring character swaps, QWERTY errors) and when multiple perturbations were combined. Homoglyph swaps and random character deletion led to incorrect predictions, indicating difficulty in dealing with visually altered characters and missing information. LSTM with attention can focus on relevant parts of a sequence, which may help it tolerate minor typos, but character-level changes, especially deletions or homoglyph swaps, can disrupt its sequence processing.

**Setting 7 (Transformer, 2 layers, post-normalization):** Managed to maintain correct sentiment predictions across most augmentations, including neighboring character swaps and random character insertions. Homoglyph swaps and random character substitution resulted in incorrect predictions. The Transformer's self-attention mechanism helps it understand context better, making it robust against certain perturbations. However, it's not immune to all noise, especially those that significantly alter word appearance.

**Setting 8 (Fine-tuning BERT):** All tested perturbations, maintaining correct predictions throughout. No significant struggles were observed. BERT's pre-training includes diverse linguistic patterns, likely encompassing similar noise to the perturbations tested. Its bidirectional context understanding makes it very robust to the types of perturbations applied.

To improve robustness, future steps could include, training models on data that includes similar perturbations to those tested, employing character-level models or subword tokenization to better handle character swaps and deletions, using preprocessing steps that attempt to correct typos and normalize text before feeding it into the model.