Assignment2 Report

What is the purpose of self-attention, and how does it facilitate capturing dependencies in sequences?

Self-attention is a mechanism that allows a model to weigh the importance of each word (or token) in a sequence relative to all other words in the same sequence. This helps capture dependencies between distant words, as each token can attend to any other token regardless of their position. By computing a weighted sum of all token embeddings, the model can understand contextual relationships more effectively. This is especially useful for tasks like translation or text generation, where understanding long-range dependencies is crucial. Self-attention thus enables parallel processing and reduces the limitations of fixed-length context windows seen in traditional models.

Why do transformers use positional encodings in addition to word embeddings? Explain how positional encodings are incorporated into the transformer architecture.

Transformers use positional encodings in addition to word embeddings because they lack a built-in mechanism to capture the order of tokens, as self-attention treats all tokens as unordered. Positional encodings provide information about the relative or absolute positions of tokens in a sequence, ensuring that the model can distinguish between differently ordered sequences.

Recent advances in positional encodings include:

1. **Learnable positional embeddings**: These embeddings are trained directly, allowing the model to adapt positional representations based on the data.
2. **Relative positional encodings**: Instead of using absolute positions, these encodings model the relative distances between tokens, improving performance on tasks where relative token positions are more important (e.g., long sequences).
3. **Rotary positional encodings (RoPE)**: These embeddings incorporate rotational transformations that can generalize across sequence lengths more effectively.

Translation Model Performance Report

The purpose of this report is to evaluate the performance of a transformer model for a translation task by varying four key hyperparameters:

- Number of encoder/decoder layers
- Number of attention heads
- Embedding dimension
- Dropout rate

Three different configurations were tested, with the results summarized below, including final training and validation losses as well as the BLEU score for translation quality. The BLEU score measures the accuracy of the model's translations compared to a reference.

Hyperparameter Configurations:

Configuration	**Layers**	**Attention Heads**	**Embedding Dimension**	**Dropout**	**Final Training Loss**	**Final Validation Loss**	**Validation BLEU Score**
Config 1	4	4	256	0.1	4.3725	4.5337	9.42
Config 2	6	8	512	0.2	3.9336	4.3549	11.66
Config 3	6	8	512	0.3	4.3545	4.5196	8.29
Config 4	8	8	512	0.4	4.7827	4.8179	4.61

Training and Validation Loss Comparison:

- **Config 1** has relatively higher training and validation losses (4.3725 and 4.5337, respectively).
- **Config 2** achieves the lowest training and validation losses (3.9336 and 4.3549), indicating better convergence and generalization.
- **Config 3** shows an increase in both training (4.3545) and validation loss (4.5196), though it remains lower than Config 1.
- **Config 4** has the highest losses in both categories (4.7827 for training and 4.8179 for validation), suggesting it may be overfitting or less optimized.

BLEU Score Comparison:

- **Config 2** also achieves the highest BLEU score of **11.66**, suggesting that its translation performance is the best among the four configurations.
- **Config 1** has a reasonable BLEU score of **9.42**, but it's still lower than Config 2.
- **Config 3** has a reduced BLEU score of **8.29**, reflecting the increasing losses and possible overfitting issues.
- **Config 4** has the worst BLEU score of **4.61**, corresponding to its high loss values, indicating poor translation quality.

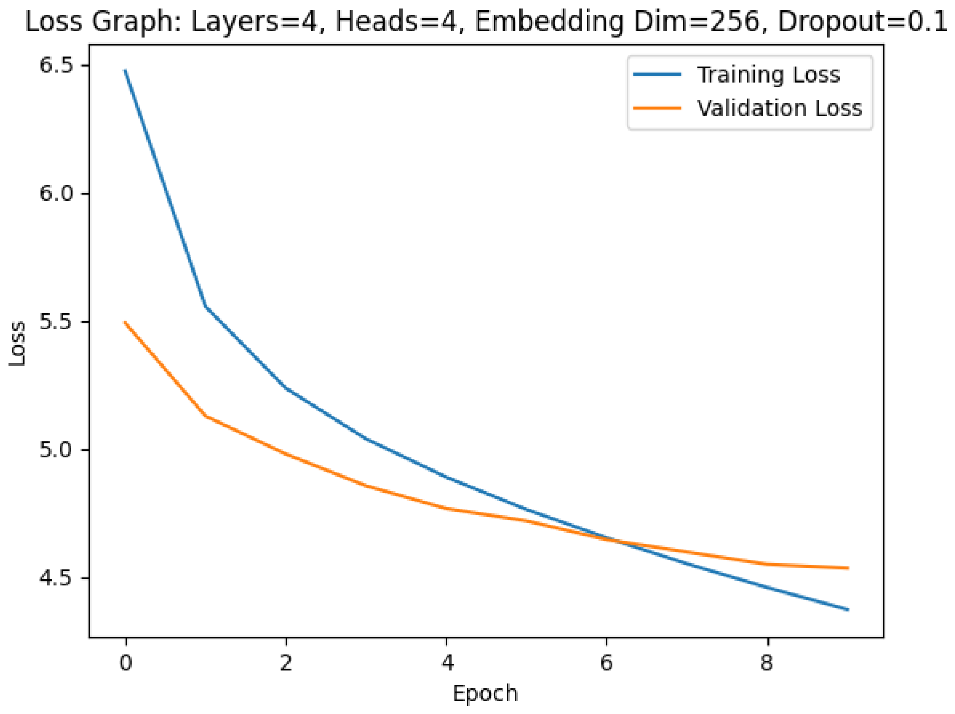
Conclusions:

- **Config 2** (6 layers, 8 attention heads, 512 embedding dimensions, 0.2 dropout) provides the best balance between training loss, validation loss, and BLEU score. It achieves the best translation quality and generalizes well across the dataset.
- Increasing the number of layers (Config 3 and Config 4) beyond 6 and using a higher dropout (0.3 and 0.4) seems to degrade performance, increasing loss and reducing BLEU scores. These configurations may introduce more noise or make the model more complex than necessary for the task.

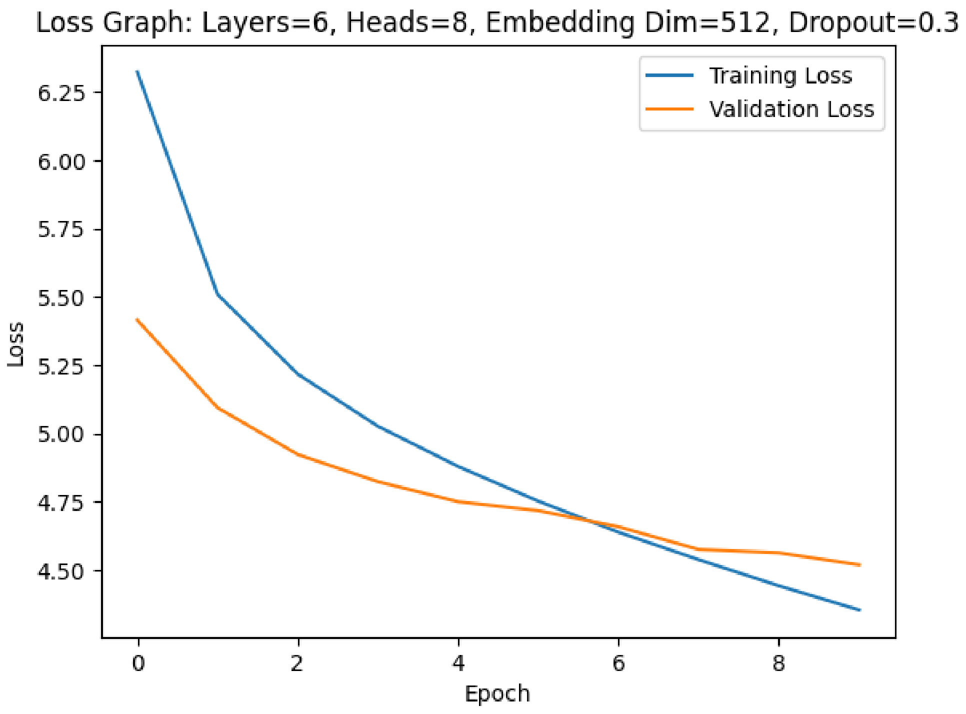
Recommendations:

- **Config 2** is the optimal configuration. It has the best training and validation performance, coupled with the highest BLEU score, indicating superior translation quality.
- Configurations with more layers and higher dropout values (Configs 3 and 4) should be avoided as they seem to hinder performance. Sticking to moderate complexity (6 layers and 0.2 dropout) appears to strike a good balance.
- Further tuning can focus on slight adjustments around Config 2, such as exploring learning rates or experimenting with other regularization techniques (e.g., different dropout strategies).

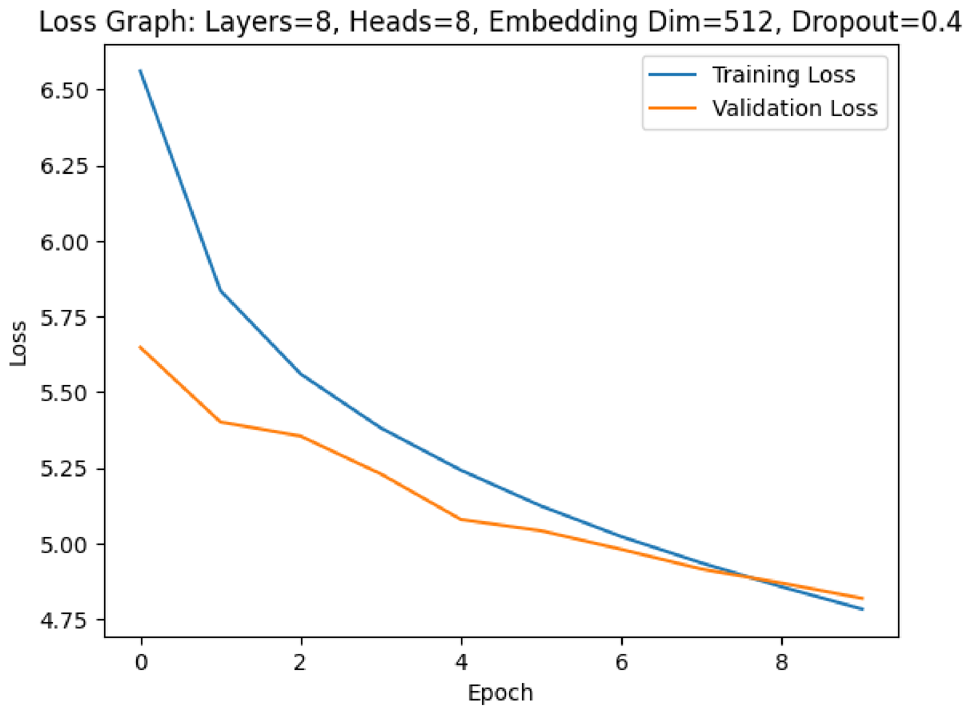
Loss Graphs



Loss Graph: Layers=4, Heads=4, Embedding Dim=256, Dropout=0.1
Training Loss
Validation Loss
6.5
6.0
5.5
5.0
4.5
Loss
0
2
4
6
8
Epoch



Loss Graph: Layers=6, Heads=8, Embedding Dim=512, Dropout=0.3
Training Loss
Validation Loss
6.25
6.00
5.75
5.50
5.25
5.00
4.75
4.50
Loss
0
2
4
6
8
Epoch


Loss Graph: Layers=8, Heads=8, Embedding Dim=512, Dropout=0.4
Training Loss
Validation Loss
Loss
Epoch
6.50
6.25
6.00
5.75
5.50
5.25
5.00
4.75
0
2
4
6
8