Report (Assignment 2)

Theory Questions

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

   The main purpose of self-attention is to allow the model to focus on different parts of the input sequence when generating representations for each token. Unlike traditional models like RNNs or LSTMs, where dependencies between tokens can diminish as the distance between them increases, self-attention allows every token in the sequence to directly attend to every other token. In a self-attention mechanism, for each token in the input sequence, three vectors are generated:

   - The **query** vector represents the token being evaluated.
   - The **key** vector is used to compare against other tokens.
   - The **value** vector contains information about the token itself.

   The query vector of each token is compared against the key vectors of all other tokens to compute attention scores. Once the attention scores are computed, a softmax function normalizes them into probabilities. These probabilities are used to take a weighted sum of the value vectors of all tokens. The resulting vector becomes the updated representation of the token, which now encodes information from the entire sequence based on the learned dependencies. Thus, self-attention enables the model to capture both short- and long-range dependencies between tokens more effectively.

2. **Why do transformers use positional encodings in addition to word embeddings? Explain how positional encodings are incorporated into the transformer architecture. Briefly describe recent advances in various types of positional encodings used for transformers and how they differ from traditional sinusoidal positional encodings.**

   Unlike recurrent models like RNNs or LSTMs, where the sequence order is inherently processed due to their sequential nature, the Transformer processes input tokens in parallel. Positional encodings provide a way for the Transformer model to learn or understand where each token is located in the

sequence, enabling it to capture sequential information and make more accurate predictions based on token positions.

Positional encodings are typically added to the word embeddings before they are fed into the Transformer. For each token, the positional encoding for its position in the sequence is computed and then added element-wise to the corresponding word embedding vector. The combined embedding is then used as the input to the Transformer layers, allowing the model to consider both the token content and its position when computing self-attention and subsequent transformations.

The recent advances in positional encodings are:

Learnable Positional Encodings:
Instead of using fixed sine and cosine functions, positional encodings can be made learnable, meaning the model learns the optimal positional representations during training. This approach is more flexible, allowing the model to adapt the encodings based on the data.

Relative Positional Encodings:
Relative positional encodings focus on capturing the relative position between tokens rather than their absolute positions. This method is particularly useful in tasks where relative positions matter more (e.g., syntactic dependencies in language modeling).

Sinusoidal encoding**s** are fixed, non-learnable, and can generalize to unseen sequence lengths but may not be as adaptable to different tasks.
Learnable encodings are trainable, allowing the model to learn task-specific positional relationships, but may struggle with generalization to longer sequences. Relative positional encodings consider the relative positions between tokens and are more flexible in capturing task-specific positional relationships.

Hyperparameters used to train the model(s)

embed_dim, num_layers, n_heads, dropout, hidden_dim

Hyperparameter tuning results

a) Hyperparameter Set 1: {'num_layers': 2, 'n_heads': 8, 'embed_dim': 512, 'dropout': 0.1}

```
Training with Hyperparameter Set 1: {'num_layers': 2, 'n_heads': 8, 'embed_dim': 512, 'dropout': 0.1}
100%|██████████| 1875/1875 [01:01<00:00, 30.44it/s]
Epoch 1, Loss: 2.0269434969584146, Train_Accuracy: 70.5384
100%|██████████| 1875/1875 [01:02<00:00, 30.22it/s]
Epoch 2, Loss: 1.4712704992453258, Train_Accuracy: 75.5638
100%|██████████| 1875/1875 [01:01<00:00, 30.44it/s]
Epoch 3, Loss: 1.1498138798713684, Train_Accuracy: 78.5534
100%|██████████| 1875/1875 [01:01<00:00, 30.59it/s]
Epoch 4, Loss: 0.9261430006980896, Train_Accuracy: 81.0226
100%|██████████| 1875/1875 [01:01<00:00, 30.70it/s]
Epoch 5, Loss: 0.7556609937349955, Train_Accuracy: 83.3573
100%|██████████| 1875/1875 [01:01<00:00, 30.58it/s]
Epoch 6, Loss: 0.627104169956843, Train_Accuracy: 85.4298
100%|██████████| 1875/1875 [01:00<00:00, 30.78it/s]
Epoch 7, Loss: 0.5400757935365041, Train_Accuracy: 86.9372
100%|██████████| 1875/1875 [01:00<00:00, 30.82it/s]
Epoch 8, Loss: 0.4705745172103246, Train_Accuracy: 88.2674
100%|██████████| 1875/1875 [01:00<00:00, 31.00it/s]
Epoch 9, Loss: 0.4139305272579193, Train_Accuracy: 89.3910
100%|██████████| 1875/1875 [01:00<00:00, 31.05it/s]
Epoch 10, Loss: 0.3701043224891027, Train_Accuracy: 90.2758
100%|██████████| 56/56 [00:00<00:00, 90.23it/s]
Loss: 1.943093525511878, Dev_Accuracy: 74.8011
100%|██████████| 1305/1305 [01:48<00:00, 12.06it/s]
BLEU Score: 0.3745
```

b) Hyperparameter Set 2: {'num_layers': 4, 'n_heads': 8, 'embed_dim': 512, 'dropout': 0.2}

```
Training with Hyperparameter Set 2: {'num_layers': 4, 'n_heads': 8, 'embed_dim': 512, 'dropout': 0.2}
100%|██████████| 1875/1875 [01:33<00:00, 20.12it/s]
Epoch 1, Loss: 2.1478432439168293, Train_Accuracy: 68.7168
100%|██████████| 1875/1875 [01:32<00:00, 20.32it/s]
Epoch 2, Loss: 1.6328859683990478, Train_Accuracy: 73.5027
100%|██████████| 1875/1875 [01:32<00:00, 20.22it/s]
Epoch 3, Loss: 1.3114960537751517, Train_Accuracy: 76.8150
100%|██████████| 1875/1875 [01:33<00:00, 20.15it/s]
Epoch 4, Loss: 1.0487812038421631, Train_Accuracy: 79.4529
100%|██████████| 1875/1875 [01:33<00:00, 20.13it/s]
Epoch 5, Loss: 0.8403986966292063, Train_Accuracy: 82.1210
100%|██████████| 1875/1875 [01:32<00:00, 20.19it/s]
Epoch 6, Loss: 0.6919381571292877, Train_Accuracy: 84.3413
100%|██████████| 1875/1875 [01:33<00:00, 20.08it/s]
Epoch 7, Loss: 0.5786842375914256, Train_Accuracy: 86.2736
100%|██████████| 1875/1875 [01:33<00:00, 20.09it/s]
Epoch 8, Loss: 0.49918310618400574, Train_Accuracy: 87.6710
100%|██████████| 1875/1875 [01:33<00:00, 20.11it/s]
Epoch 9, Loss: 0.4376993365923564, Train_Accuracy: 88.8716
100%|██████████| 1875/1875 [01:33<00:00, 20.04it/s]
Epoch 10, Loss: 0.39103757701714836, Train_Accuracy: 89.7930
100%|██████████| 56/56 [00:00<00:00, 61.00it/s]
Loss: 1.9404557645320892, Dev_Accuracy: 74.9781
100%|██████████| 1305/1305 [03:22<00:00,  6.43it/s]
BLEU Score: 0.3796
```

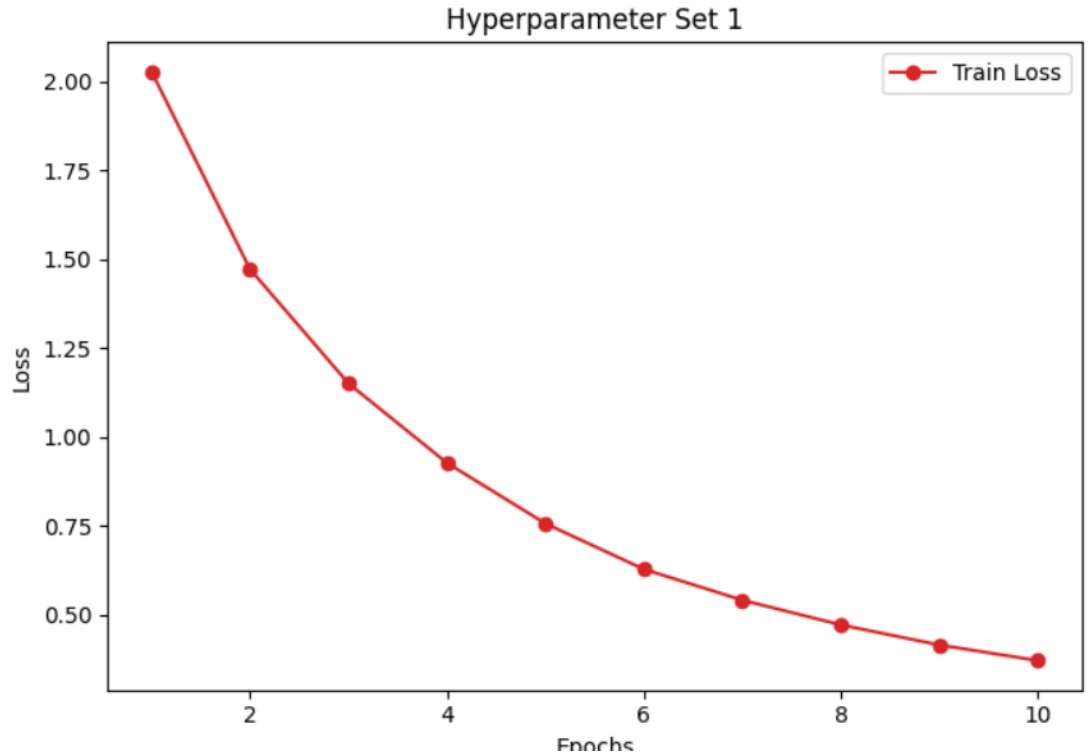
c) Hyperparameter Set 3: {'num_layers': 4, 'n_heads': 12, 'embed_dim': 768, 'dropout': 0.2}

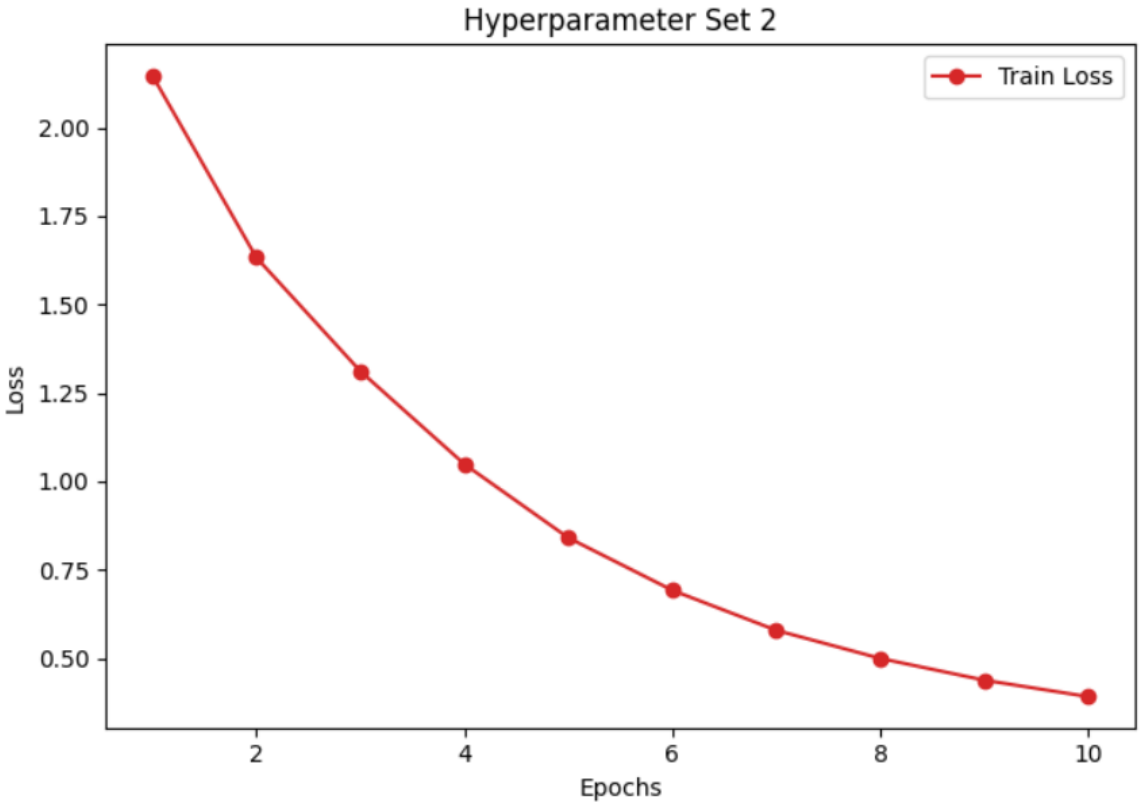
```
Training with Hyperparameter Set 3: {'num_layers': 4, 'n_heads': 12, 'embed_dim': 768, 'dropout': 0.2}
100%|██████████| 1875/1875 [02:05<00:00, 14.90it/s]
Epoch 1, Loss: 2.1405533683776854, Train_Accuracy: 68.5525
100%|██████████| 1875/1875 [02:04<00:00, 15.00it/s]
Epoch 2, Loss: 1.8197551169395447, Train_Accuracy: 70.1800
100%|██████████| 1875/1875 [02:05<00:00, 14.94it/s]
Epoch 3, Loss: 1.5858149734020233, Train_Accuracy: 72.1616
100%|██████████| 1875/1875 [02:06<00:00, 14.85it/s]
Epoch 4, Loss: 1.2593177656332653, Train_Accuracy: 75.8859
100%|██████████| 1875/1875 [02:05<00:00, 14.92it/s]
Epoch 5, Loss: 0.9791069890022278, Train_Accuracy: 79.5067
100%|██████████| 1875/1875 [02:05<00:00, 14.89it/s]
Epoch 6, Loss: 0.7501829787572225, Train_Accuracy: 83.0103
100%|██████████| 1875/1875 [02:05<00:00, 14.93it/s]
Epoch 7, Loss: 0.5963982950210571, Train_Accuracy: 85.7605
100%|██████████| 1875/1875 [02:05<00:00, 14.98it/s]
Epoch 8, Loss: 0.49445183174610136, Train_Accuracy: 87.6673
100%|██████████| 1875/1875 [02:05<00:00, 14.96it/s]
Epoch 9, Loss: 0.42611503226757047, Train_Accuracy: 89.0195
100%|██████████| 1875/1875 [02:04<00:00, 15.03it/s]
Epoch 10, Loss: 0.3744210694233576, Train_Accuracy: 90.1105
100%|██████████| 56/56 [00:01<00:00, 41.87it/s]
Loss: 2.0752521944897517, Dev_Accuracy: 74.0262
100%|██████████| 1305/1305 [03:18<00:00,  6.59it/s]
BLEU_Score: 0.3854
```

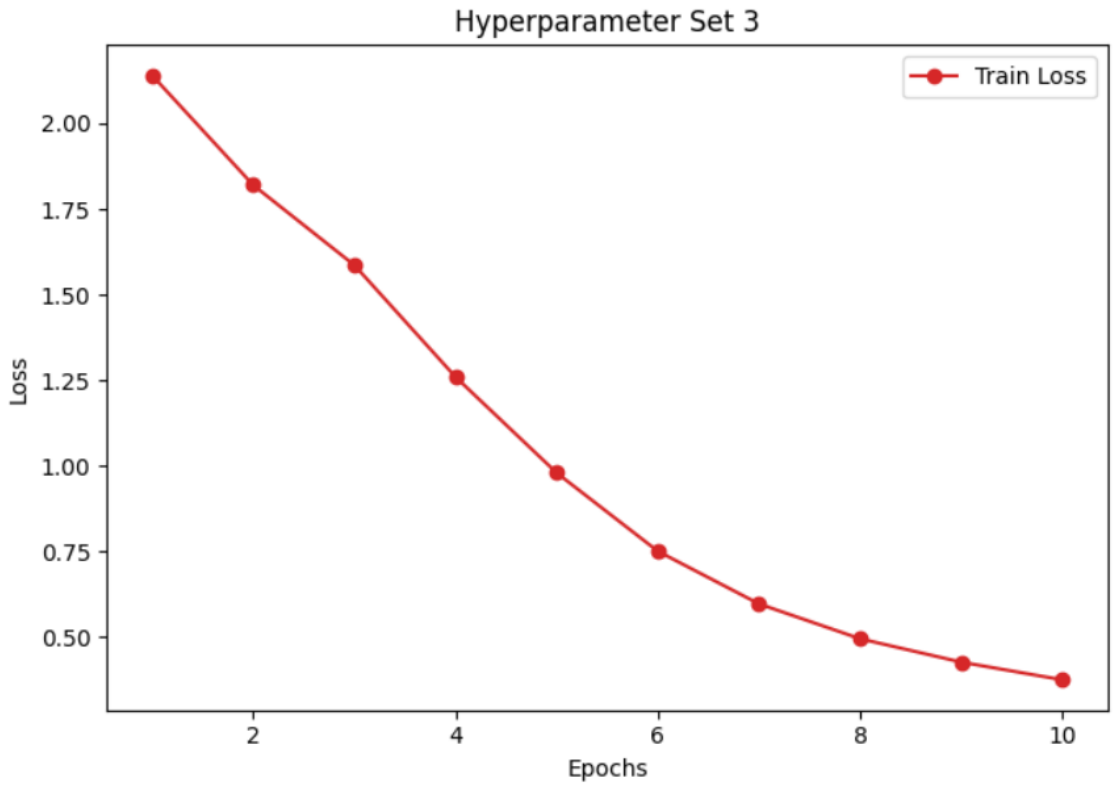
d) Hyperparameter Set 4: {'num_layers': 6, 'n_heads': 12, 'embed_dim': 768, 'dropout': 0.1}

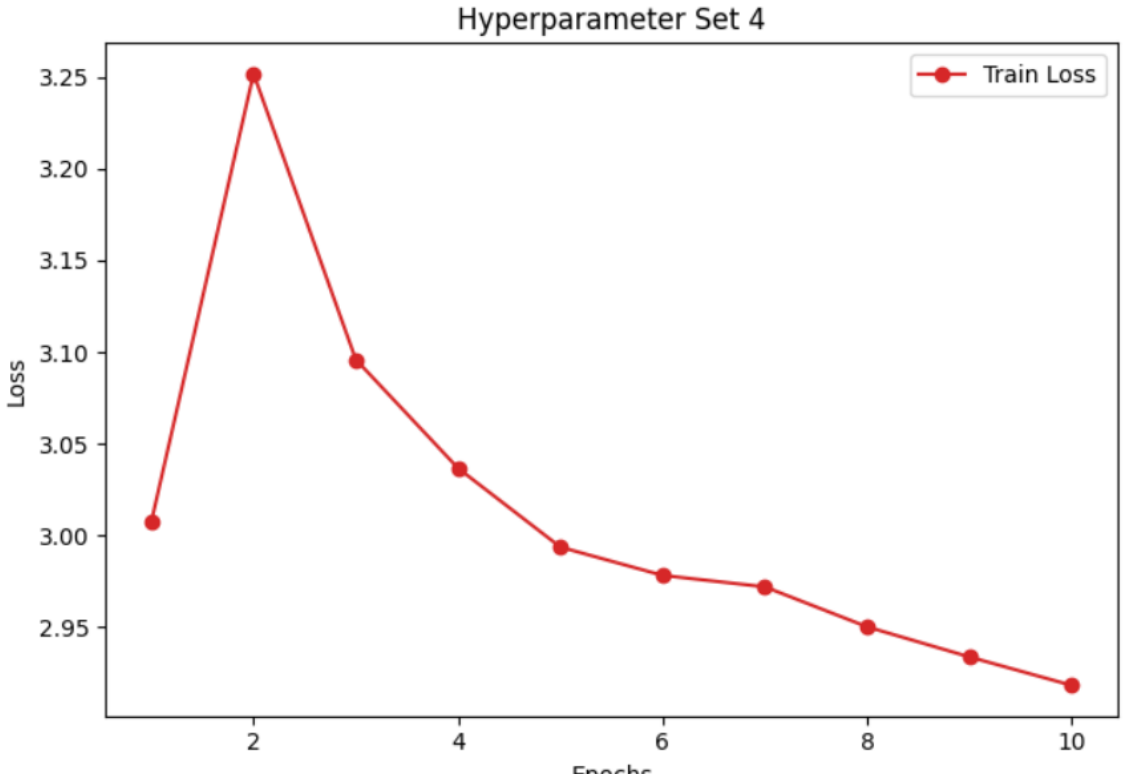
```
Training with Hyperparameter Set 4: {'num_layers': 6, 'n_heads': 12, 'embed_dim': 768, 'dropout': 0.1}
100%|██████████| 1875/1875 [02:46<00:00, 11.28it/s]
Epoch 1, Loss: 3.007438509432475, Train_Accuracy: 63.9824
100%|██████████| 1875/1875 [02:47<00:00, 11.22it/s]
Epoch 2, Loss: 3.2517508827845254, Train_Accuracy: 63.2568
100%|██████████| 1875/1875 [02:47<00:00, 11.22it/s]
Epoch 3, Loss: 3.0959647183418273, Train_Accuracy: 63.3688
100%|██████████| 1875/1875 [02:47<00:00, 11.22it/s]
Epoch 4, Loss: 3.036747418053945, Train_Accuracy: 63.4239
100%|██████████| 1875/1875 [02:47<00:00, 11.18it/s]
Epoch 5, Loss: 2.9937667939186094, Train_Accuracy: 63.5649
100%|██████████| 1875/1875 [02:47<00:00, 11.20it/s]
Epoch 6, Loss: 2.9783344897905986, Train_Accuracy: 63.5102
100%|██████████| 1875/1875 [02:46<00:00, 11.23it/s]
Epoch 7, Loss: 2.9720951636950175, Train_Accuracy: 63.5033
100%|██████████| 1875/1875 [02:46<00:00, 11.25it/s]
Epoch 8, Loss: 2.95029370136261, Train_Accuracy: 63.5704
100%|██████████| 1875/1875 [02:47<00:00, 11.21it/s]
Epoch 9, Loss: 2.933898165035248, Train_Accuracy: 63.6517
100%|██████████| 1875/1875 [02:47<00:00, 11.20it/s]
Epoch 10, Loss: 2.918397142187754, Train_Accuracy: 63.6604
100%|██████████| 56/56 [00:01<00:00, 33.95it/s]
Loss: 4.774970544236047, Dev_Accuracy: 63.2874
100%|██████████| 1305/1305 [27:06<00:00,  1.25s/it]
BLEU_Score: 0.3154
```

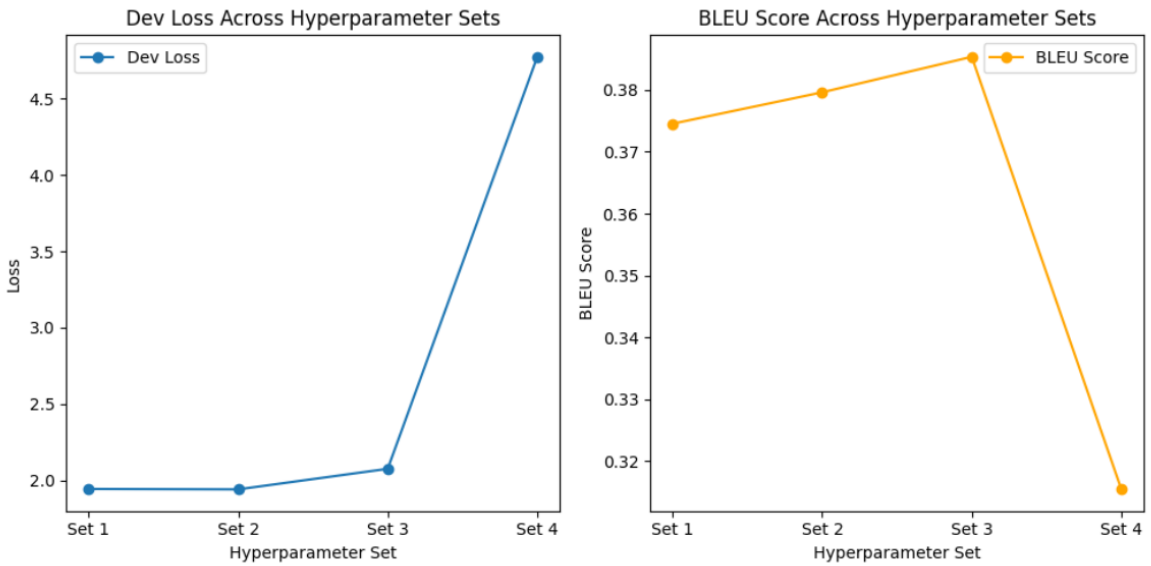
Loss and BLEU Score Graphs






Hyperparameter Set 3
Train Loss
Loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
2
4
6
8
10
Epochs


Hyperparameter Set 4
Train Loss
Loss
3.25
3.20
3.15
3.10
3.05
3.00
2.95
2
4
6
8
10



Analysis

The model trained using hyperparameter set 3 gives the highest BLEU score as its larger embedding size allows the model to capture more nuanced semantic relationships between words and the use of more attention heads allows the model attend to different parts of the input and handle more complex dependencies. Also the model depth is optimal as having more layers than set 1 (2 layers) allows the model to better capture hierarchical representations without being too shallow. At the same time, it's not as deep as set 4 (6 layers), which overfits. While set 2 has lower dev loss and comparable dev accuracy, set 3 likely handles longer, more complex sentence structures better, translating into higher BLEU despite similar dev accuracy.