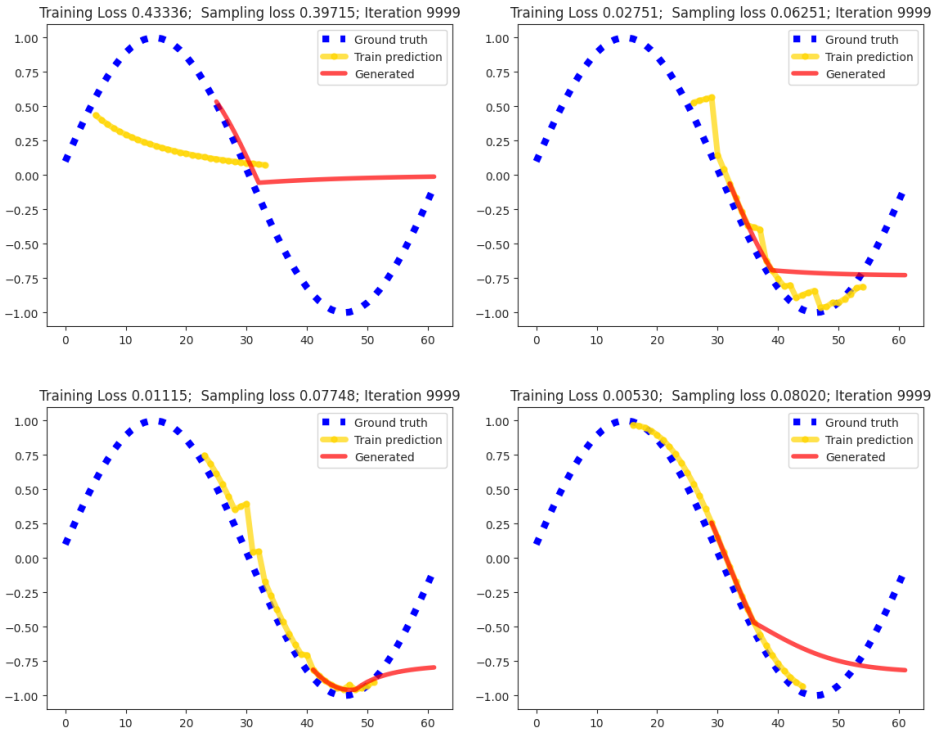
Daniel-Cristian-Marian Țăpuși

Neural Networks - HW2 report

Task 1:

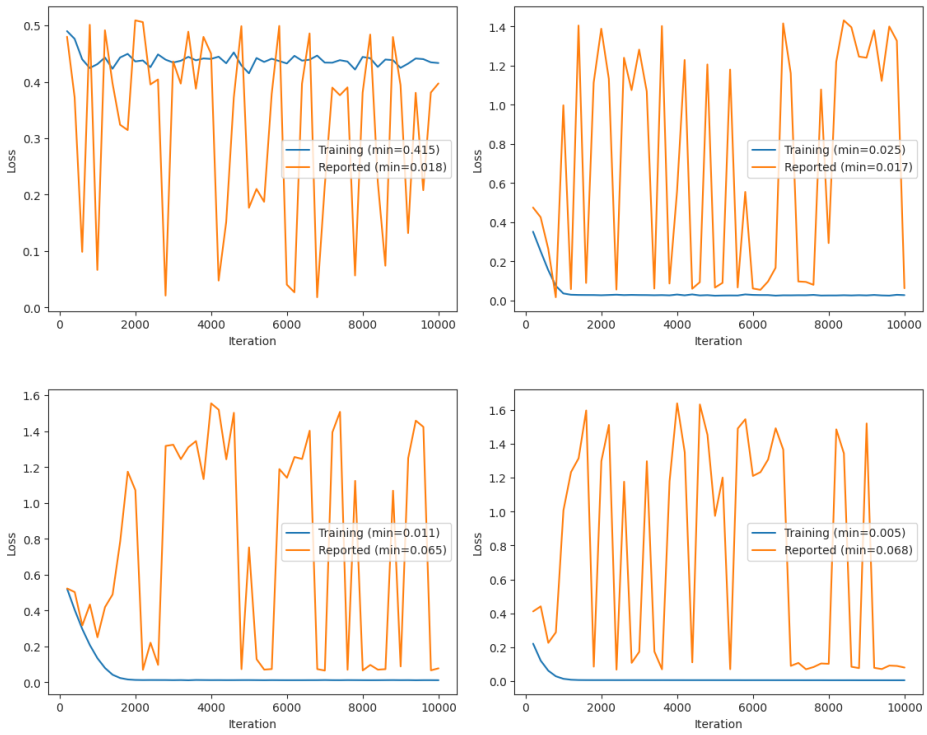
Experiment 1 (teacher forcing = 0, 0.5, 0.75, 1):

The goal of the first experiments was to determine how teacher forcing probability affected the RNN's capacity to recognize and anticipate sine wave patterns. The other hyperparameters were left at their default settings, and four distinct probabilities (p=0, 0.5, 0.75, and 1.0) were tested. By using a methodical approach, we were able to see how varying degrees of ground truth injection during training impact the model's capacity for generalization as well as the learning process.



With no teacher forcing (p=0), the model struggles significantly, quickly deviating from the ground truth and converging to an almost constant value. At p=0.5, we observe

improved performance with better initial predictions, though the model still loses accuracy in later timesteps. Higher probabilities (p=0.75 and p=1.0) show excellent training predictions and better maintenance of the sinusoidal pattern initially but continue to show variations in the sequence that is generated after the first phase, which are especially noticeable in the full teacher forcing case.

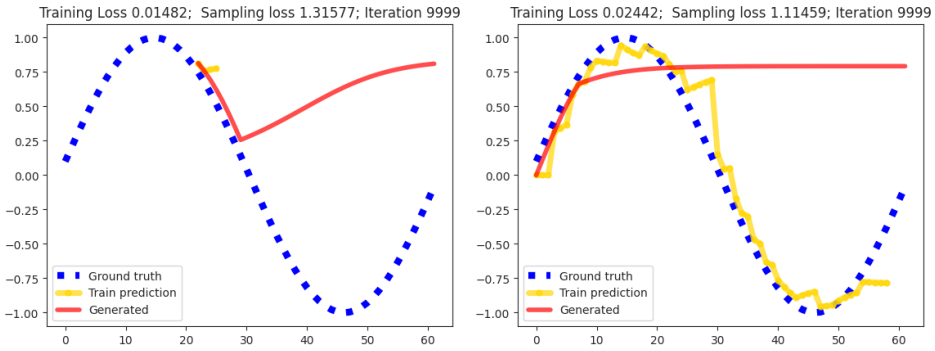


Training loss consistently decreases as teacher forcing probability increases, with p=1.0 achieving the lowest training loss (0.005) and p=0 the highest (0.415). However, the sampling loss tells a different story, showing significant fluctuations across all configurations. The performance between training and reported loss is most balanced at p=0.5, whereas overfitting is obvious at higher probabilities, as indicated by the widening gap between training and reported loss. According to this, p=0.5 provides the best balance between learning and generalization ability, indicating that although higher teacher forcing probabilities improve training performance, they do not always result in better generalization.

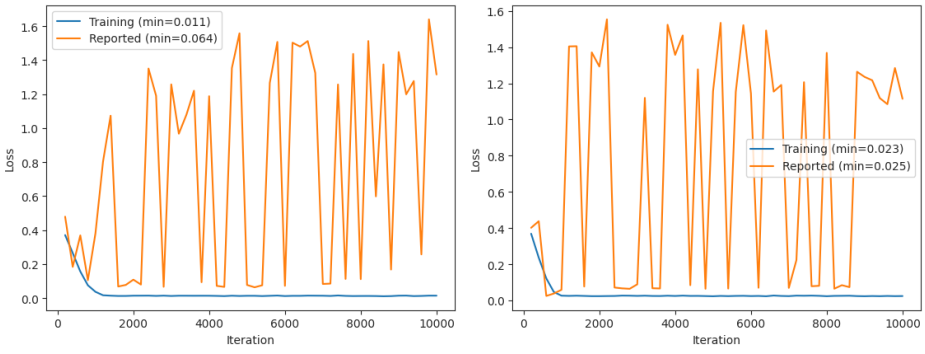
Experiment 2 (teacher forcing = 0.5 and very small vs very large UNROLL_LENGTH):

The model performs better during training (yellow line closely matching the blue dotted line of the ground truth) when UNROLL_LENGTH is larger, but it still struggles with

long-term generation (red line). After the initial phase, the generated sequence tends to converge to a constant value, this indicates challenges in sustaining the periodic pattern. The smaller UNROLL_LENGTH, on the other hand, performs worse overall, deviating from the ground truth pattern more quickly and making less accurate predictions during the training and generation stages. This suggests that when training with only short sequences, the model finds it difficult to learn the underlying sinusoidal structure.



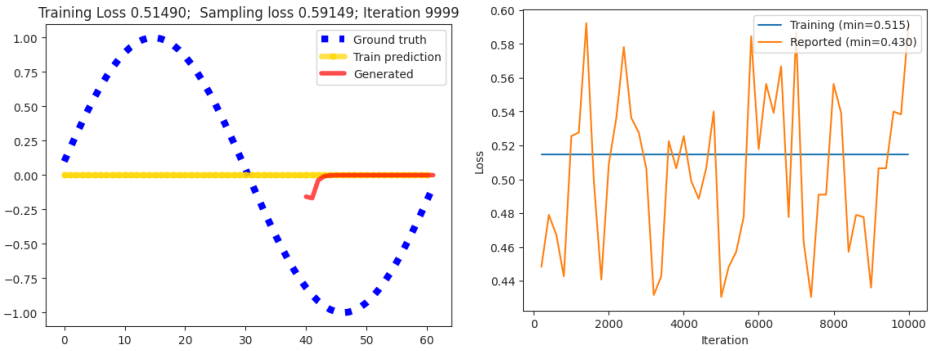
The training loss (blue line) for both scenarios converges to relatively low values (0.011 and 0.023), demonstrating the model's ability to optimize during training. The reported loss (orange line), on the other hand, displays high volatility and notably larger values, especially in the frequent spikes that reach up to 1.4-1.6. The model can learn from the training sequences, but it has trouble generalizing, particularly when it comes to prediction outside of the training window, as indicated by the significant difference between training and reported loss and the high variance in reported loss. This instability is more noticeable in the smaller UNROLL_LENGTH case, suggesting that the model's capacity to recognize and preserve the underlying pattern is greatly impacted by the limited sequence exposure during training.



Experiment 3 (teacher forcing = 0, UNROLL_LENGTH = 62 and WARM_START = 2):

The generated sequence (red line) and training predictions (yellow line) both perform poorly, converge to a nearly constant value close to zero, and totally miss the sinusoidal pattern of the ground truth (blue dotted line). This behavior is especially noticeable since it shows that the model has not even learned the fundamental oscillatory character of the data, as both the generated sequence and the training predictions stay virtually flat over the course of the sequence. Without the advantage of teacher forcing during training, the model cannot establish any meaningful pattern recognition due to the limited warm start of two timesteps.

Although this tiny difference between training and reported loss could appear to be favorable at first, in this instance, it shows that the model has converged to a bad local minimum and is reliably predicting values that are close to constant independent of the input. The model has essentially failed to learn the underlying pattern, as evidenced by the stability of both losses at such high levels and their close closeness to one another. This is probably because of the difficult combination of limited warm start despite the big unroll duration and no teacher forcing.

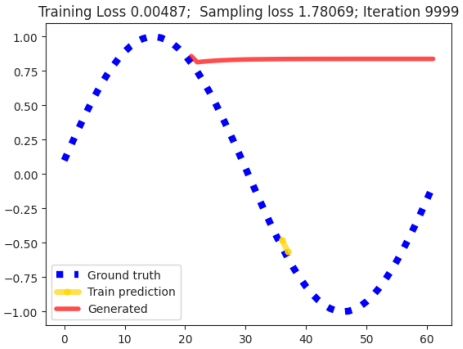


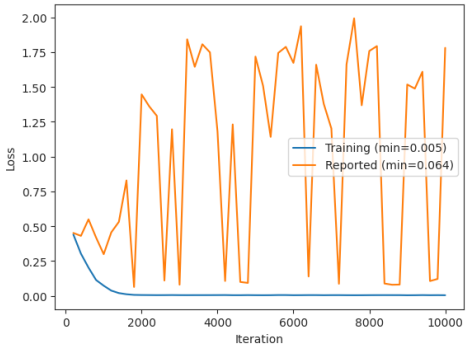
Experiment 4 (teacher forcing = 1, UNROLL_LENGTH = 3 and WARM_START = 2):

There is an obvious discrepancy between training and generation capabilities in the output forecasts. Throughout its short sequence length, the model produces almost flawless predictions when trained with instructor forcing. The generated sequence, however, rapidly departs from the ground truth and settles at about 0.8, demonstrating the model's incapacity to represent the sine wave's periodic nature. This illustrates how, even if it

permits flawless training prediction, the combination of extremely short UNROLL_LENGTH and full teacher forcing prevents the model from learning the underlying pattern required for autonomous generation.

The reported loss shows extreme volatility with spikes up to 2.0 (minimum 0.064), however the training loss quickly converges to an exceptional 0.005. Despite the ideal teacher forcing conditions, the model is obviously overfitting to short sequences during training without gaining any meaningful understanding of the longer-term sinusoidal pattern, as evidenced by the significant difference between training and reported loss and the high volatility in reported loss.





Question 1: Difference between teacher forcing and learning on own samples: What are the pros and cons of teacher forcing?

A better training performance results from teacher forcing's ability to provide a faster convergence and a more stable initial learning. It may, however, result in overfitting and excessive dependence on the true values of the sinusoidal function that we are trying to learn. The model's ability to generalize is reduced at high teacher forcing probabilities due to a significant difference between how it learns (by using correct values) and how it has to function in reality (by using its own predictions).

Question 2: In which setup (combination of unroll_length and teacher forcing probability) is the model struggling to learn?

The model struggles most with extreme unroll lengths and shows its worst learning performance when no teacher force is used (p=0) with any unroll length. Learning outcomes are particularly poor when combining extremely short unroll length (3) and full teacher

forcing, or long unroll length (62) and no teacher forcing, especially with a minimal warm start of 2.

Question 3: How does warm starting affect test time prediction? Why?

By providing the model with accurate initial values, warm starting improves the stability of its forecasts and helps in test time prediction. Before it begins to make its own predictions, the model benefits from this early phase of receiving actual values. The model struggles to start a successful sequence of predictions if there aren't enough warm initial steps.

Question 4: What happens if the structure of interest is much longer than the unroll length?

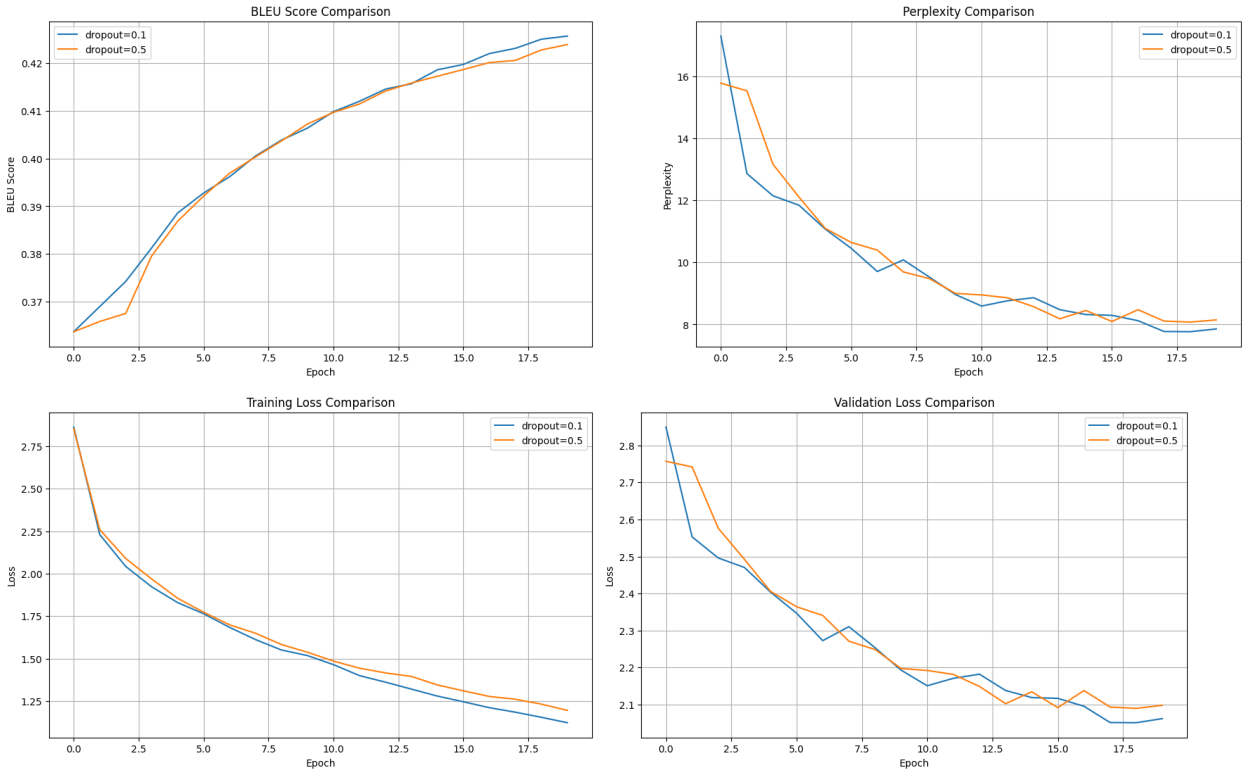
The model is unable to comprehend and accurately predict long-term patterns when the pattern it must learn is longer than the unroll length. After this point, the model's predictions start to lose accuracy because it can only identify patterns for as long as its unroll length. This often causes the model to either lose the repeating pattern it should follow or the predicting of constant values.

Task 2:

Objective 1:

Experiment 1 – dropout comparison (emb_dim: 128, hid_dim: 256, batch_size: 128, tf_ratio: 0.5 and dropout 0.1 vs 0.5):

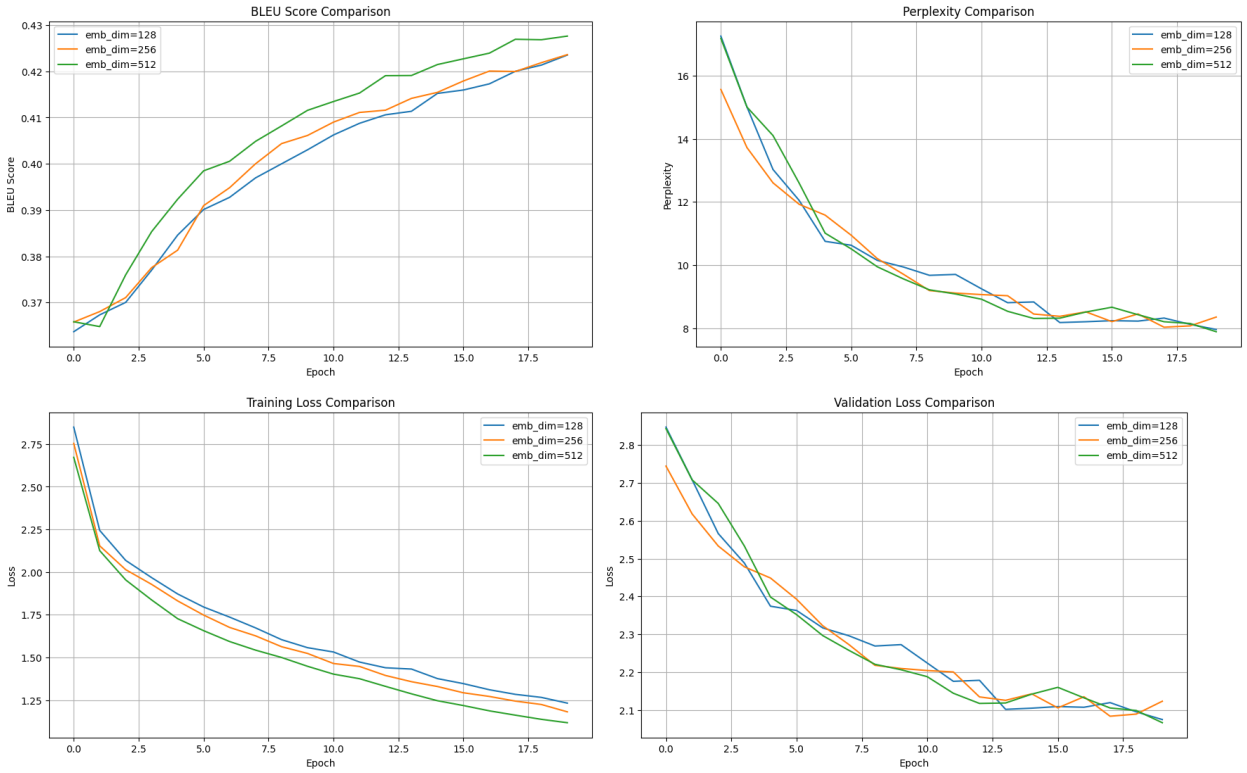
Initially, the model with 0.1 dropout shows faster improvement in BLEU score and lower perplexity, but after epoch 10, both configurations converge to similar performance. Looking at the training and validation losses, we see that 0.1 dropout achieves slightly lower training loss but similar validation loss to 0.5 dropout, suggesting that the higher dropout rate helps prevent overfitting without significantly impacting model performance. The final BLEU scores are comparable, with 0.1 dropout performing marginally better, but 0.5 dropout might offer better generalization for different test scenarios due to its stronger regularization effect.



This behavior can be explained by the nature of the translation task: 0.1 dropout is sufficient for a good performance while permitting slightly faster training convergence. A higher dropout (0.5) provides stronger regularization by randomly deactivating more neurons during training, but the Multi30k dataset's relatively simple sentence structures may not require such aggressive regularization.

Experiment 2 – embedding dimension comparison (hid_dim: 256, batch_size: 128, dropout: 0.5, tf_ratio: 0.5 and emb_dim: 128 vs 256 vs 512):

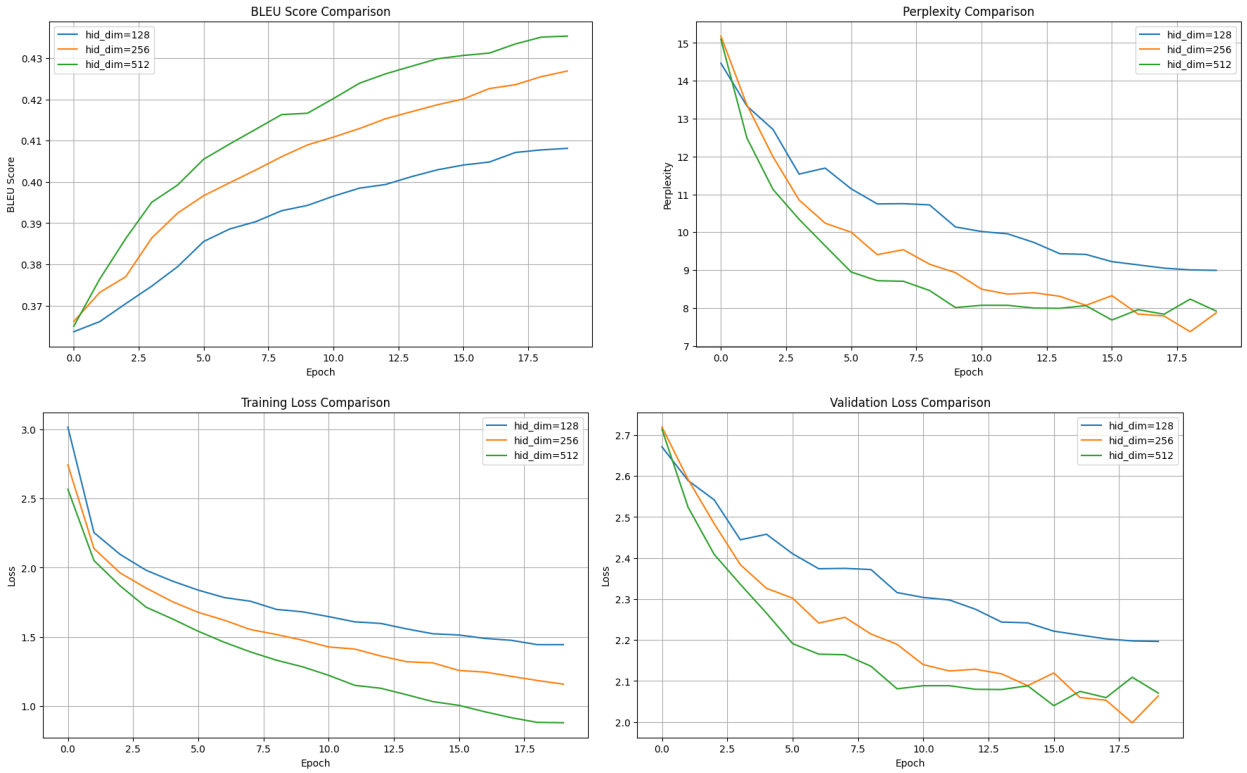
The model with embedding dimension 512 demonstrated superior performance, achieving the highest BLEU score and maintaining consistently lower training and validation losses throughout the training process. All three configurations show steady improvement in BLEU scores, but the 512-dimension embedding maintains a clear advantage after epoch 5. The perplexity plots show similar convergence for all three dimensions by the end of training, though 512 dimensions shows more stable behavior in the later epochs.



The model's improved capacity to capture more complex word associations and semantic meanings in the translation job is responsible for its superior performance of bigger embedding dimensions. The decreasing returns between 256 and 512 dimensions, however, imply that while increasing the embedding dimension may result in higher processing costs, it may not produce appreciable gains.

Experiment 3 – hidden dimension comparison (emb_dim: 256, batch_size: 128, dropout: 0.5, tf_ratio: 0.5 and hid_dim 128 vs 256 vs 512):

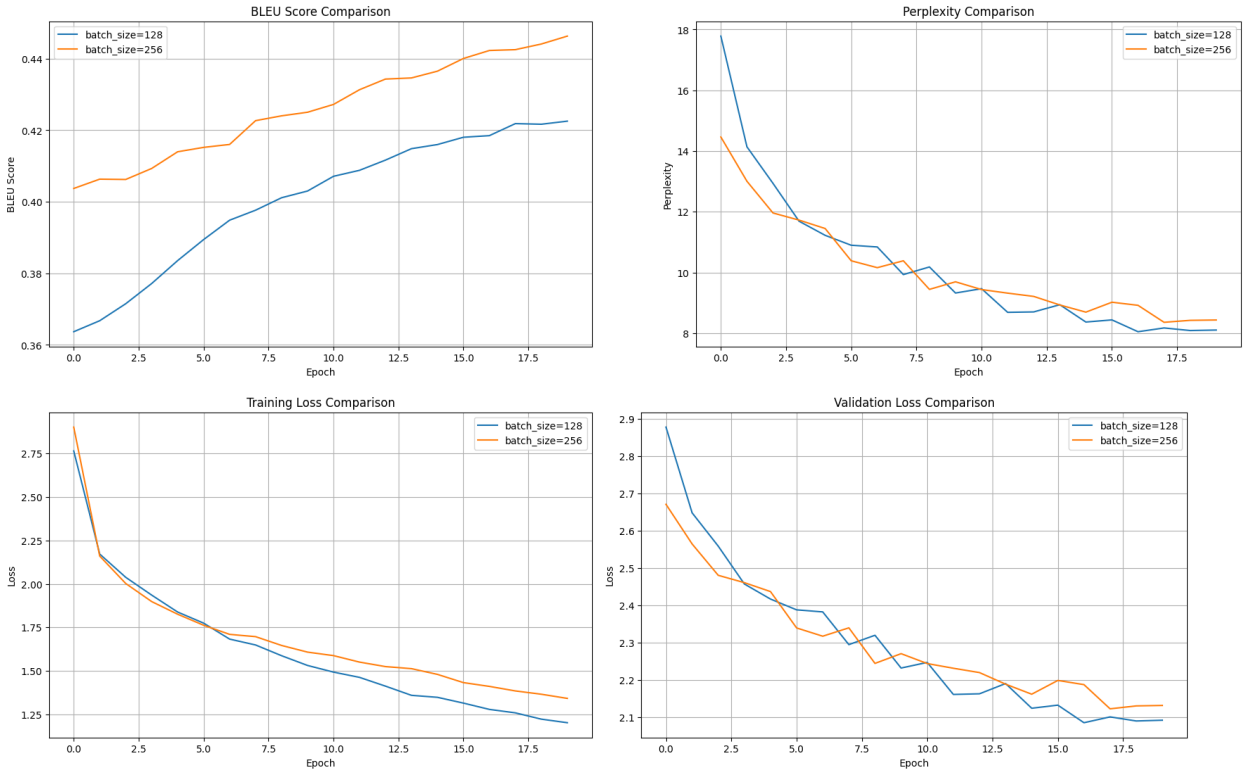
The results clearly show that larger hidden dimensions lead to better performance. The model with hidden dimension 512 consistently outperformed the others, achieving the highest BLEU score (0.435) and lowest perplexity throughout training. Looking at the loss curves, both training and validation losses show faster convergence and lower final values for larger hidden dimensions, with the 512-dimension model showing particularly strong improvement in the early epochs.



The larger hidden state's capacity to preserve more details about the input sequence and capture more complex word associations during translation is what accounts for this performance gain. The difference between 256 and 512 dimensions suggests that the extra capacity of 512 hidden units offers significant advantages for this translation task, whereas the 128-dimension model performs noticeably worse, indicating that it lacks the capacity to capture the required sequence information.

Experiment 4 – batch size comparison (emb_dim: 256, hid_dim: 256, dropout: 0.5, tf_ratio: 0.5 and batch_size 128 vs 256):

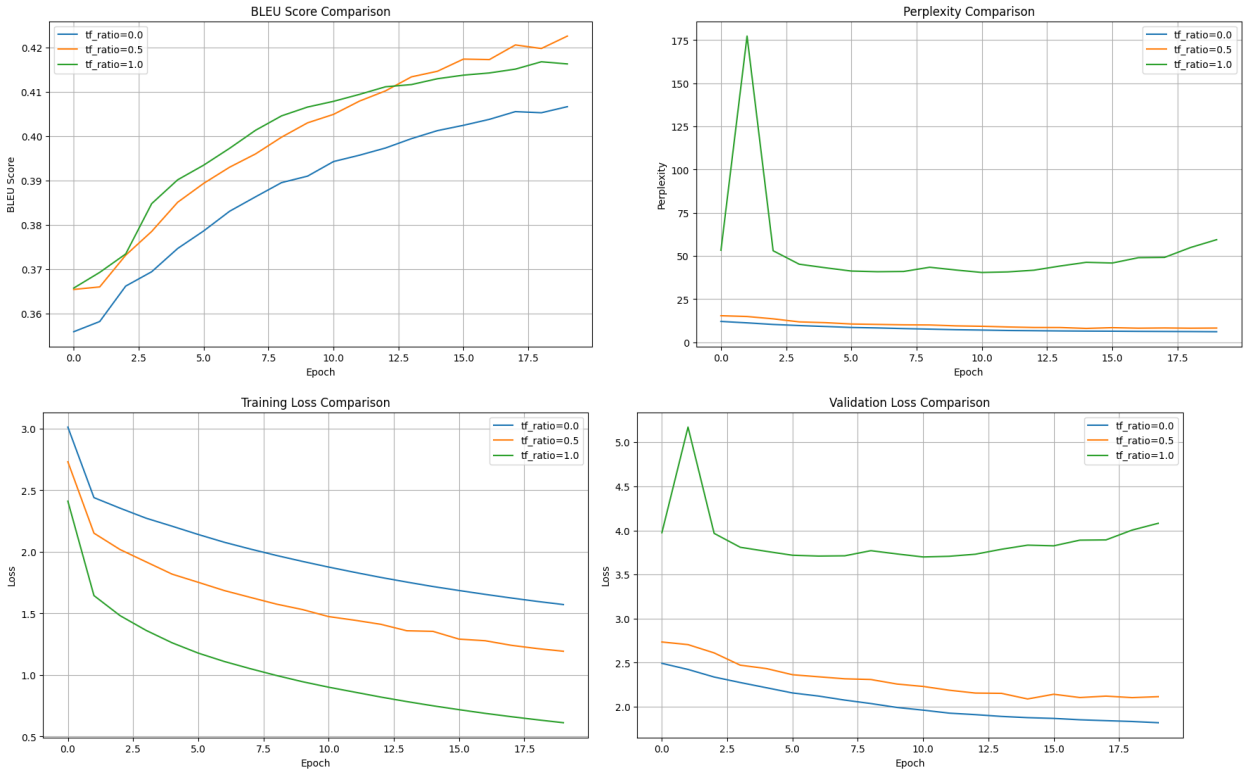
The larger batch size of 256 showed consistently better performance, achieving a final BLEU score of 0.445 compared to 0.422 for batch size 128. Looking at the perplexity curves, both configurations show similar convergence patterns, though batch size 256 demonstrates more stable behavior early in training. The training loss curve shows that batch size 128 achieves slightly lower training loss, while the validation loss curves are comparable, suggesting that the larger batch size provides better generalization.



Since larger batches offer a more accurate approximation of the true gradient direction, the higher performance of batch size 256 can be assigned to more consistent gradient updates during training. The model gains from seeing more samples simultaneously during parameter updates, as shown by the better BLEU scores with batch size 256. This results in more reliable translation capabilities that are free from overfitting.

Experiment 5 – teacher forcing comparison (emb_dim: 256, hid_dim: 256, batch_size: 128, dropout: 0.5 and tf_ratio 0 vs 0.5 vs 1):

The BLEU score comparison shows that a teacher forcing ratio of 0.5 achieves the best performance by the end of training, reaching approximately 0.425, while no teacher forcing (0.0) performs worst at 0.405. Looking at the perplexity plots, tf_ratio=1.0 shows significant instability early in training with a large spike, while 0.5 maintains consistently lower and more stable perplexity throughout. The training loss curves show that full teacher forcing (1.0) achieves the lowest training loss, but the validation loss reveals this leads to overfitting, with tf_ratio=1.0 showing the highest validation loss.



The exposure bias problem explains this behavior: when full teacher forcing is used, the model performs poorly during inference when it has to utilize its own predictions and becomes overly reliant on correct prior tokens during training. The optimal compromise is offered by the balanced strategy of tf_ratio=0.5, which enables the model to learn from both ground truth and its own predictions, improving generalization and stabilizing training dynamics.

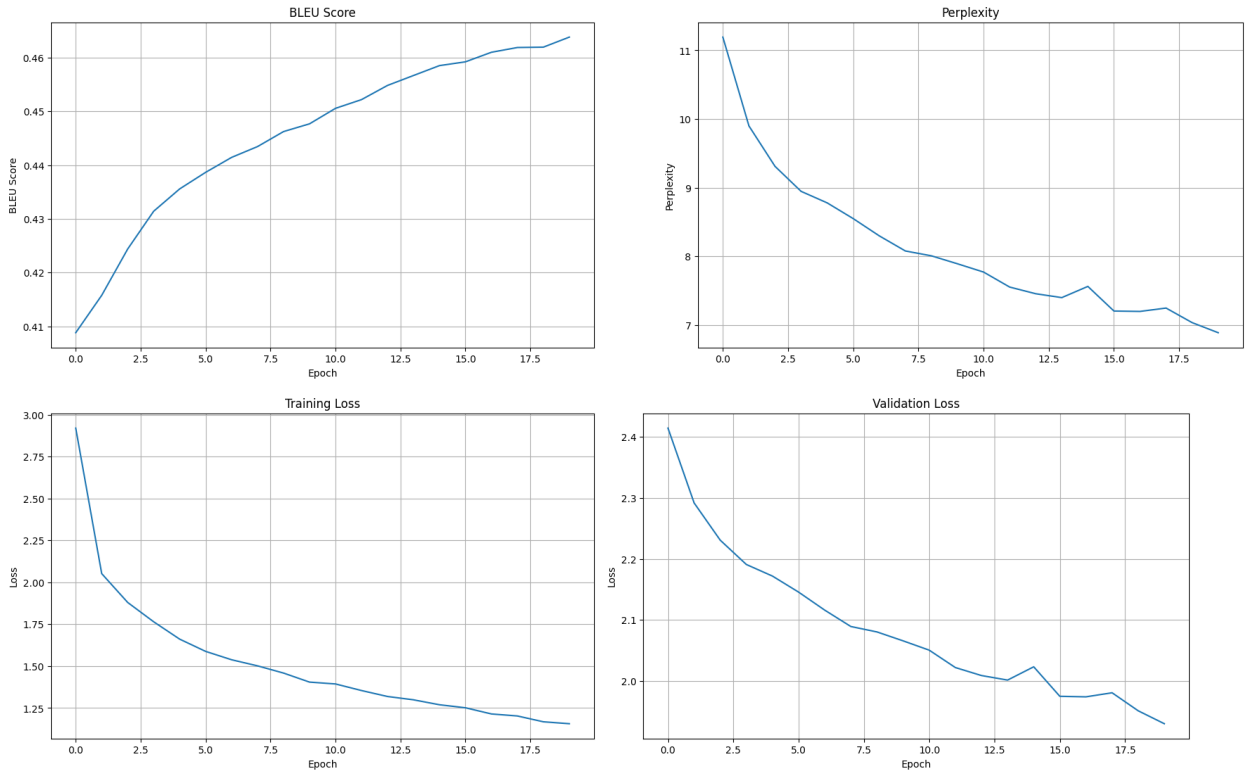
The best model:

After evaluating all configurations, the best performing model used: embedding dimension 256, hidden dimension 256, batch size 256, dropout 0.5, and teacher forcing ratio 0.5. This configuration achieved the highest BLEU score of 0.4463 and maintained a reasonable perplexity of 8.4337, demonstrating good translation quality and model confidence.

Objective 2:

For objective 2, I have modified the best-performing configuration from objective 1 by implementing a bidirectional LSTM encoder while maintaining a unidirectional decoder. The results show significant improvements in model performance, with the bidirectional LSTM

model achieving a higher BLEU score of 0.4638 (compared to 0.4463 in the unidirectional version) and lower perplexity of 6.8858 (improved from 8.4337).



Looking at the training curves, it can be observed the steady BLEU score improvement throughout training, with faster initial learning and reaching a higher final value. The perplexity demonstrates a more stable decrease compared to the unidirectional model, indicating better model confidence. The training loss converges smoothly from around 3.0 to 1.2, showing consistent learning, while the validation loss follows a similar pattern without diverging, suggesting good generalization.

The bidirectional encoder's capacity to process input sentences both forward and backward, which enables it to collect greater contextual information, is responsible for this improved performance. The model can produce more accurate translations by better understanding the relationships between words and their meanings in the source sentence by taking into account both past and future context when encoding each word. While taking advantage of the improved encoded representations, the maintained unidirectional decoder guarantees that the model produces translations in the proper sequential order.