ANLP ASSIGNMENT 3

Theory Questions

How does the introduction of ”soft prompts” address the limitations of discrete text prompts in large language models? Why might soft prompts be considered a more flexible and efficient approach for task-specific conditioning?

Soft prompts differ from traditional text prompts in that they use continuous embeddings instead of discrete words, allowing for a more refined level of control over a language model's responses. This shift enables the model to capture complex, task-specific instructions without needing explicit, worded descriptions. By using these soft prompts, models can adapt to new tasks with greater flexibility and efficiency; rather than retraining the entire model, only a small set of parameters associated with the prompt needs adjustment. This approach results in quicker fine-tuning that is targeted and effective, making the model highly adaptable for various applications. Consequently, soft prompts provide a more efficient method for conditioning models, offering a streamlined way to tailor outputs across different tasks without the resource-heavy demands of traditional methods.

How does the efficiency of prompt tuning relate to the scale of the language model Discuss the implications of this relationship for future developments in large-scale language models and their adaptability to specific tasks.

As language models increase in scale, prompt tuning becomes an efficient alternative to full model retraining by adjusting only a small subset of parameters. This selective adjustment makes tuning faster and significantly reduces the computational resources required, which is particularly advantageous for adapting large models to new tasks. Because prompt tuning requires fewer resources, it provides a practical means to fine-tune expansive models for specific applications quickly and cost-effectively. This adaptability enhances the versatility of large-scale models, enabling them to be tailored for various specialized tasks without the need for extensive data or costly retraining. The efficiency and scalability of prompt tuning thus have promising implications for future developments, where large models can be more readily employed across a wider range of specialized applications, allowing quicker and more accessible task-specific adaptations.

What are the key principles behind Low-Rank Adaptation (LoRA) in fine-tuning large language models? How does LoRA improve upon traditional fine-tuning methods regarding efficiency and performance?

Low-Rank Adaptation (LoRA) is a technique used to fine-tune large language models by focusing on efficiently adjusting a low-rank subset of the model's weight matrices. Rather than updating all parameters during fine-tuning, LoRA inserts low-rank matrices into certain layers of the model, enabling effective adaptation with minimal parameter changes. This approach significantly reduces the computational cost compared to traditional fine-tuning, where every parameter in the model may need adjustment. LoRA's low-rank updates allow for task-specific learning without drastically increasing the model’s memory or computational requirements, making it efficient for fine-tuning even very large models. Additionally, because LoRA only modifies a small part of the model, it preserves the original model's knowledge, enhancing stability and generalization performance. This method is particularly useful for adapting large models to a wide range of tasks, as it provides an efficient, scalable, and memory-friendly way to specialize a model without extensive resources.

Discuss the theoretical implications of introducing low-rank adaptations to the parameter space of large language models. How does this affect the expressiveness and generalization capabilities of the model compared to standard fine-tuning?

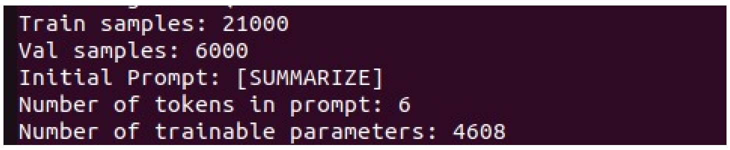
Introducing low-rank adaptations (LoRA) to the parameter space of large language models has significant theoretical implications for both expressiveness and generalization capabilities. By focusing on low-rank updates, LoRA allows models to learn task-specific information without overwhelming the parameter space with excessive modifications. This targeted adaptation preserves the original model's knowledge while enabling it to capture new patterns efficiently. The low-rank structure effectively reduces the complexity of the parameter space, which can enhance the model's expressiveness by allowing it to represent a wide variety of tasks without requiring a full parameter update. Furthermore, this approach mitigates the risk of overfitting, as it restricts the number of parameters being modified, leading to improved generalization on unseen data. In contrast, standard fine-tuning often involves extensive updates across all parameters, which can lead to a loss of the model's prior knowledge and a higher likelihood of overfitting. Therefore, LoRA not only enhances the efficiency of the adaptation process but also strikes a balance between leveraging existing knowledge and accommodating new task-specific information, ultimately resulting in a more robust and versatile model.

PROMPT TUNING

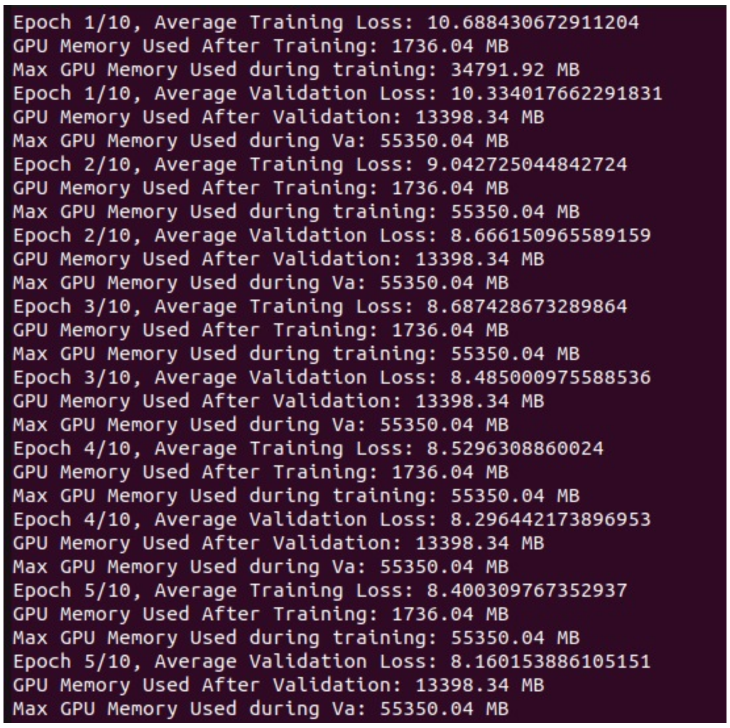
Training:

Hyperparameters

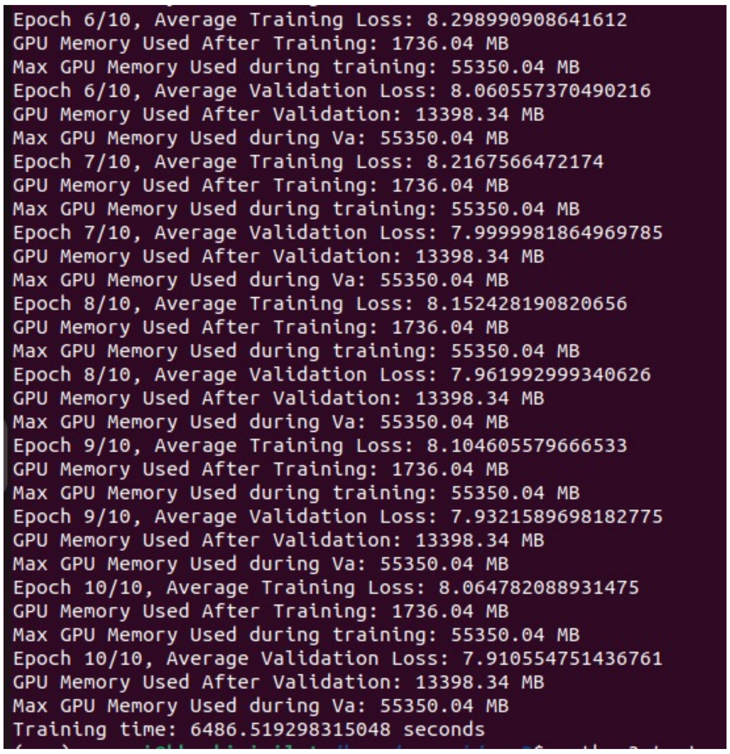
```
initial_prompt = '[SUMMARIZE]'
num_prompts = 6
batch_size = 32
learning_rate = 1e-4
n_epochs = 10
optimizer = 'Adam'
max_length = 512
```



```
Train samples: 21000
Val samples: 6000
Initial Prompt: [SUMMARIZE]
Number of tokens in prompt: 6
Number of trainable parameters: 4608
```

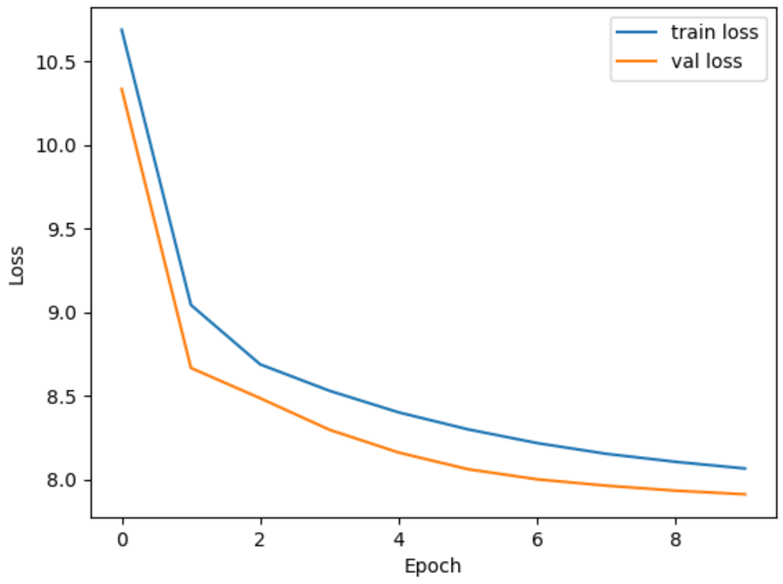


```
Epoch 1/10, Average Training Loss: 10.688430672911204
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 34791.92 MB
Epoch 1/10, Average Validation Loss: 10.334017662291831
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 2/10, Average Training Loss: 9.042725044842724
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 2/10, Average Validation Loss: 8.666150965589159
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 3/10, Average Training Loss: 8.687428673289864
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 3/10, Average Validation Loss: 8.485000975588536
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 4/10, Average Training Loss: 8.5296308860024
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 4/10, Average Validation Loss: 8.296442173896953
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 5/10, Average Training Loss: 8.400309767352937
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 5/10, Average Validation Loss: 8.160153886105151
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
```

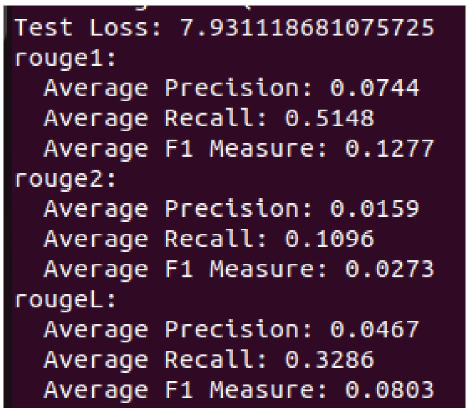


```
Epoch 6/10, Average Training Loss: 8.298990908641612
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 6/10, Average Validation Loss: 8.060557370490216
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 7/10, Average Training Loss: 8.2167566472174
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 7/10, Average Validation Loss: 7.9999981864969785
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 8/10, Average Training Loss: 8.152428190820656
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 8/10, Average Validation Loss: 7.961992999340626
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 9/10, Average Training Loss: 8.104605579666533
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 9/10, Average Validation Loss: 7.9321589698182775
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Epoch 10/10, Average Training Loss: 8.064782088931475
GPU Memory Used After Training: 1736.04 MB
Max GPU Memory Used during training: 55350.04 MB
Epoch 10/10, Average Validation Loss: 7.910554751436761
GPU Memory Used After Validation: 13398.34 MB
Max GPU Memory Used during Va: 55350.04 MB
Training time: 6486.519298315048 seconds
```

Epoch-Loss Curve



Testing:



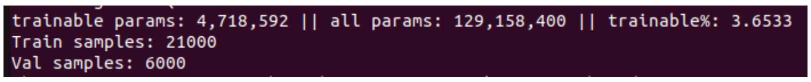
```
Test Loss: 7.931118681075725
rouge1:
  Average Precision: 0.0744
  Average Recall: 0.5148
  Average F1 Measure: 0.1277
rouge2:
  Average Precision: 0.0159
  Average Recall: 0.1096
  Average F1 Measure: 0.0273
rougeL:
  Average Precision: 0.0467
  Average Recall: 0.3286
  Average F1 Measure: 0.0803
```

LORA

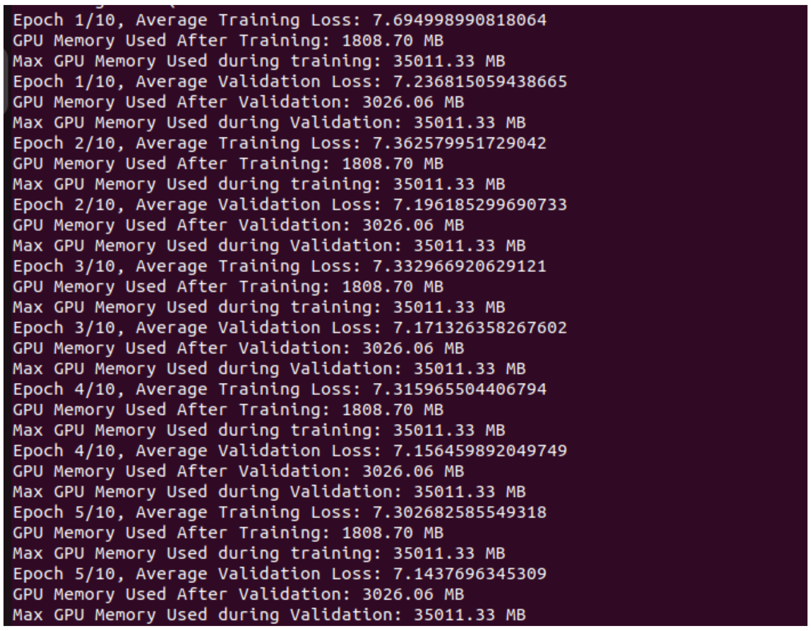
Training:

Hyperparameters

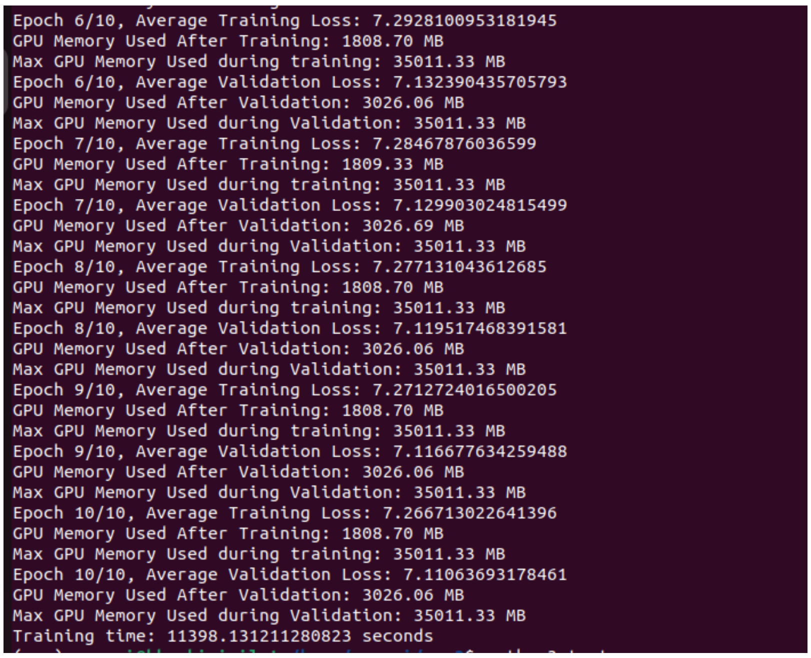
```
lora_r = 128
lora_alpha = 32
lora_dropout = 0.1
batch_size = 32
learning_rate = 1e-4
n_epochs = 10
optimizer = 'Adam'
max_length = 512
```



```
trainable params: 4,718,592 || all params: 129,158,400 || trainable%: 3.6533
Train samples: 21000
Val samples: 6000
```

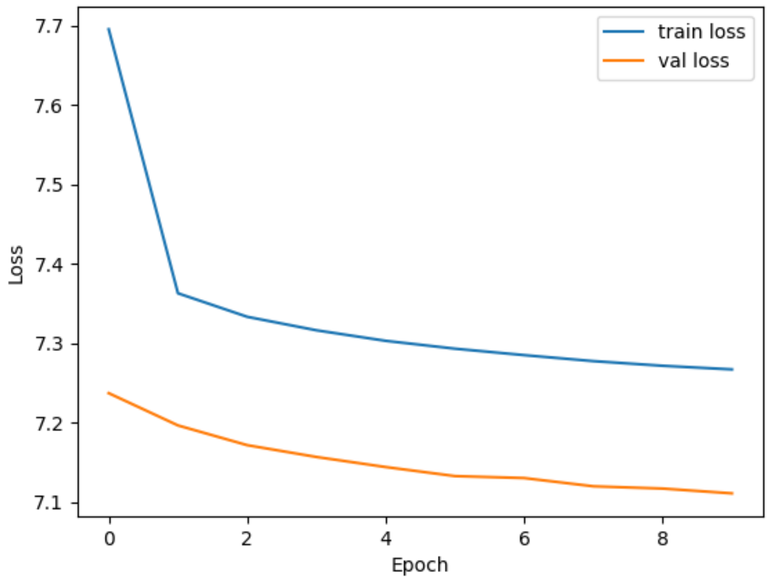


```
Epoch 1/10, Average Training Loss: 7.694998990818064
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 1/10, Average Validation Loss: 7.236815059438665
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 2/10, Average Training Loss: 7.362579951729042
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 2/10, Average Validation Loss: 7.196185299690733
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 3/10, Average Training Loss: 7.332966920629121
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 3/10, Average Validation Loss: 7.171326358267602
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 4/10, Average Training Loss: 7.315965504406794
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 4/10, Average Validation Loss: 7.156459892049749
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 5/10, Average Training Loss: 7.302682585549318
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 5/10, Average Validation Loss: 7.1437696345309
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
```

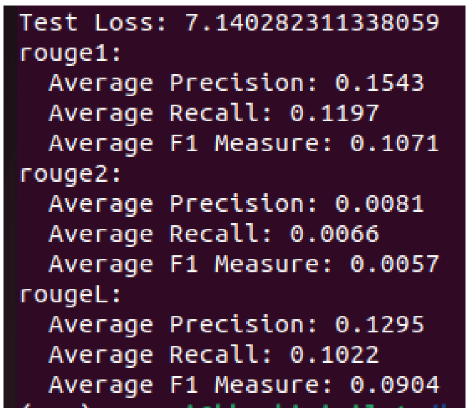


```
Epoch 6/10, Average Training Loss: 7.2928100953181945
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 6/10, Average Validation Loss: 7.132390435705793
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 7/10, Average Training Loss: 7.28467876036599
GPU Memory Used After Training: 1809.33 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 7/10, Average Validation Loss: 7.129903024815499
GPU Memory Used After Validation: 3026.69 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 8/10, Average Training Loss: 7.277131043612685
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 8/10, Average Validation Loss: 7.119517468391581
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 9/10, Average Training Loss: 7.2712724016500205
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 9/10, Average Validation Loss: 7.116677634259488
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Epoch 10/10, Average Training Loss: 7.266713022641396
GPU Memory Used After Training: 1808.70 MB
Max GPU Memory Used during training: 35011.33 MB
Epoch 10/10, Average Validation Loss: 7.11063693178461
GPU Memory Used After Validation: 3026.06 MB
Max GPU Memory Used during Validation: 35011.33 MB
Training time: 11398.131211280823 seconds
```

Epoch-Loss Curve



Testing:

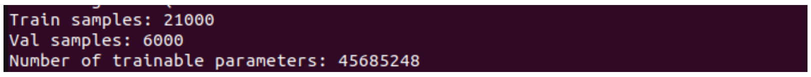

```
Test Loss: 7.140282311338059
rouge1:
  Average Precision: 0.1543
  Average Recall: 0.1197
  Average F1 Measure: 0.1071
rouge2:
  Average Precision: 0.0081
  Average Recall: 0.0066
  Average F1 Measure: 0.0057
rougeL:
  Average Precision: 0.1295
  Average Recall: 0.1022
  Average F1 Measure: 0.0904
```

Traditional Fine-tuning

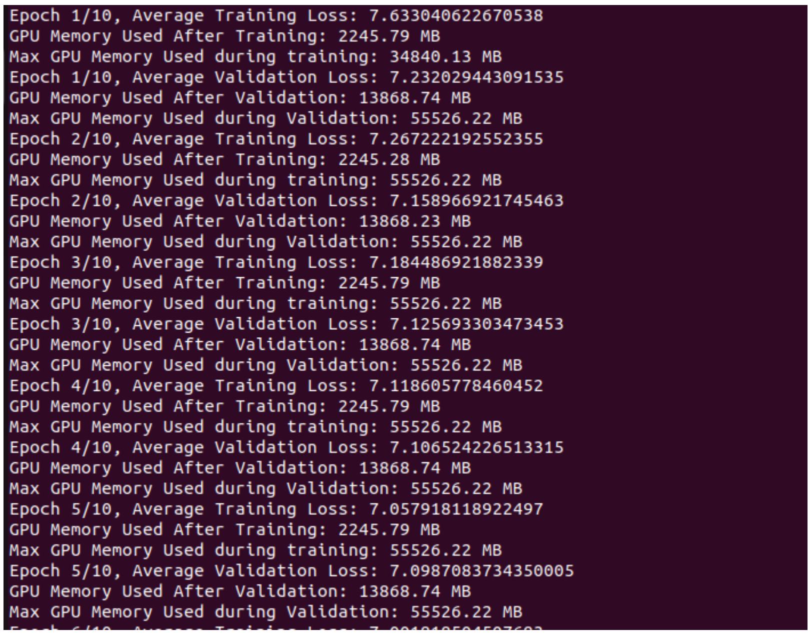
Training:

Hyperparameters

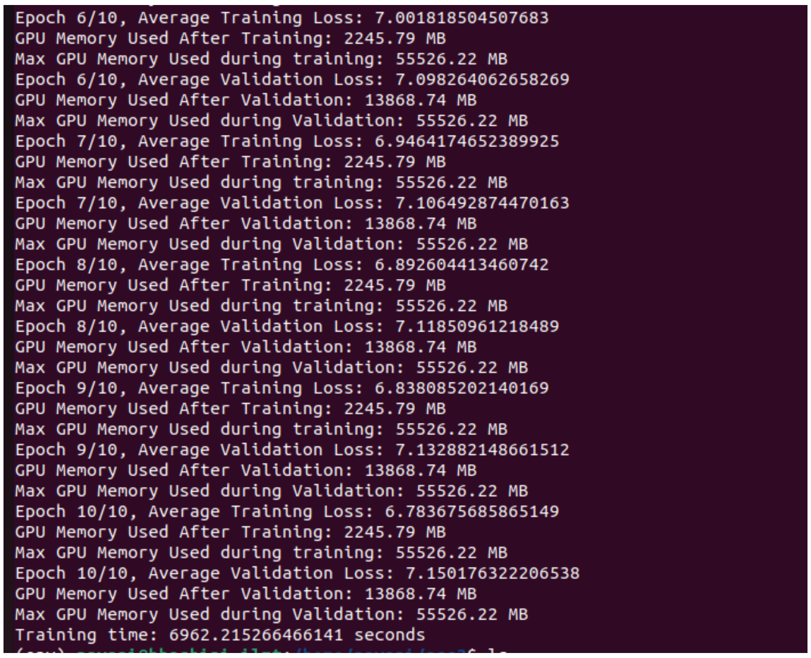
```
max_length = 512
batch_size = 32
n_epochs = 10
learning_rate = 1e-4
optimizer = 'Adam'
```



```
Train samples: 21000
Val samples: 6000
Number of trainable parameters: 45685248
```

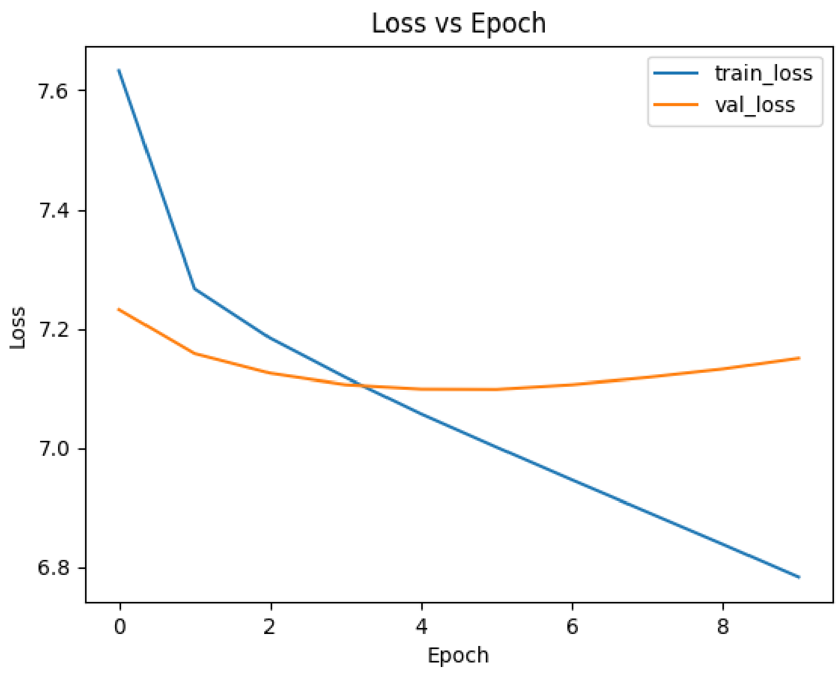


```
Epoch 1/10, Average Training Loss: 7.633040622670538
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 34840.13 MB
Epoch 1/10, Average Validation Loss: 7.232029443091535
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 2/10, Average Training Loss: 7.267222192552355
GPU Memory Used After Training: 2245.28 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 2/10, Average Validation Loss: 7.158966921745463
GPU Memory Used After Validation: 13868.23 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 3/10, Average Training Loss: 7.184486921882339
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 3/10, Average Validation Loss: 7.125693303473453
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 4/10, Average Training Loss: 7.118605778460452
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 4/10, Average Validation Loss: 7.106524226513315
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 5/10, Average Training Loss: 7.057918118922497
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 5/10, Average Validation Loss: 7.0987083734350005
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
```

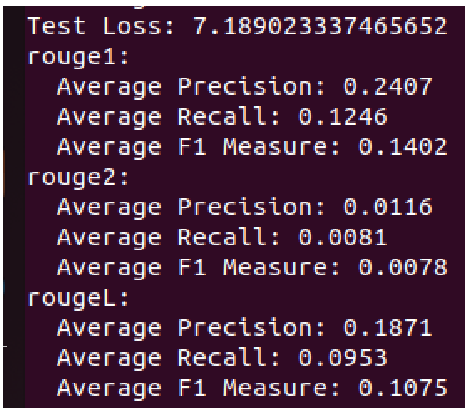


```
Epoch 6/10, Average Training Loss: 7.001818504507683
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 6/10, Average Validation Loss: 7.098264062658269
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 7/10, Average Training Loss: 6.9464174652389925
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 7/10, Average Validation Loss: 7.106492874470163
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 8/10, Average Training Loss: 6.892604413460742
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 8/10, Average Validation Loss: 7.11850961218489
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 9/10, Average Training Loss: 6.838085202140169
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 9/10, Average Validation Loss: 7.132882148661512
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Epoch 10/10, Average Training Loss: 6.783675685865149
GPU Memory Used After Training: 2245.79 MB
Max GPU Memory Used during training: 55526.22 MB
Epoch 10/10, Average Validation Loss: 7.150176322206538
GPU Memory Used After Validation: 13868.74 MB
Max GPU Memory Used during Validation: 55526.22 MB
Training time: 6962.215266466141 seconds
```

Epoch-Loss Curve



Testing:



```
Test Loss: 7.189023337465652
rouge1:
  Average Precision: 0.2407
  Average Recall: 0.1246
  Average F1 Measure: 0.1402
rouge2:
  Average Precision: 0.0116
  Average Recall: 0.0081
  Average F1 Measure: 0.0078
rougeL:
  Average Precision: 0.1871
  Average Recall: 0.0953
  Average F1 Measure: 0.1075
```

Analysis:

Fine-tuning approach	Trainable Parameters	Training time	GPU memory used	Test loss	Rouge-L F1 Score
Prompt tuning	4608	1.8 hours	55.35 GB	7.93	0.0803
Lora	4718592	3.166 hours	35.011 GB	7.14	0.1022
Traditional FT	45685248	1.93 hours	55.526 GB	7.18	0.1075

1. **Prompt tuning** utilizes significantly fewer trainable parameters (4,608) compared to **Lora** (4,718,592) and **Traditional FT** (45,685,248). This indicates that prompt tuning may be more efficient in leveraging existing model capabilities without extensive retraining.
2. **Lora** takes the longest to train (3.166 hours), which is unexpected given its parameter count. This may be due to server load since **Traditional FT**, with more trainable parameters, completes training in a shorter

time (1.93 hours). This indicates that external factors may significantly impact training duration.

3. The **ROUGE-L F1 Score** indicates that **Traditional FT** (0.1075) slightly outperforms **Lora** (0.1022) and **Prompt tuning** (0.0803). While Traditional FT demands more resources, it appears to yield marginally better summarization quality, highlighting the trade-off between resource consumption and performance across different fine-tuning methods.