HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION COMMUNICATION TECHNOLOGY

IT4772E - Natural Language Processing

Machine Reading Comprehension

Supervised by:

Assoc. Prof. Le Thanh Huong

Group 22 Members:

Bui Nguyen Khai - 20225501

Tran Ngoc Quang - 20225523

Nguyen Viet Tien - 20225533

Nguyen Trong Phuong Bach - 20225473

Vu Minh Hieu - 20225494

Hanoi - Vietnam

2025

Contents

We thank Dr. Le Thanh Huong for reading our group's project report. If there are any mistakes, or misconceptions in the report, we would like to hear the feedback, and comments from you on our group's report. Our work can be found in this repository: https://github.com/buinguyenkhai/nlp-20241-mrc-squad-lora

Abstract

Machine Reading Comprehension (MRC) has seen significant advancements with the advent of large-scale pre-trained language models. However, fine-tuning these models for specific tasks can be computationally expensive and memory-intensive, particularly as model sizes continue to grow. In this project, we explore the performance of several transformer-based models: RoBERTa, SpanBERT, and ELECTRA - on the SQuAD 1.1 benchmark using various fine-tuning strategies. These include full fine-tuning, and parameter-efficient fine-tuning with Low-Rank Adaptation (LoRA). Our experiments compare the effectiveness of these strategies based on F1 and Exact Match (EM) scores, offering insights into the trade-offs between performance and computational efficiency in MRC tasks.

1 Introduction

Machine Reading Comprehension (MRC) is a subfield of Natural Language Processing (NLP) that focuses on the task of building a system that understands the passage to answer questions related to it. The input to the Reading Comprehension model is a question and a context/passage. The output to the model is the answer from the passage.

This project aims to present a study on the fine-tuning of several transformer-based models including RoBERTa [4], SpanBERT [3], and ELECTRA[1] on the Stanford Question Answering Dataset (SQuAD 1.1). We investigated distinct fine-tuning strategies: full fine-tuning, and a parameter-efficient approach: Low-Rank Adaptation (LoRA). Our experiments evaluate these combinations based on F1 score and Exact Match (EM).

2 Dataset

For this project, we utilized the Stanford Question Answering Dataset (SQuAD 1.1). Stanford Question Answering Dataset (SQuAD) [5] is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. Below are the techinical detail of the dataset:

- **Structure**: Each example in SQuAD 1.1 includes a context passage, a question pertaining to that passage, and a human-annotated ground-truth answer. A key characteristic of SQuAD 1.1 is that the answer to every question is a direct span of text from the corresponding context passage.
- **Size**: contains 107,785 question-answer pairs on 536 articles. It is split into a training set (approximately 87,000 pairs) and a development set (approximately 10,000 pairs) which we use for evaluation.
- **Task**: The task is extractive question answering. Given a context and a question, the model must predict the start and end positions of the answer span within the context.

The resulting human performance score on the test set is 77.0% for the exact match metric, and 86.8% for F1. Mismatch occurs mostly due to inclusion/exclusion of non-essential phrases (e.g., monsoon trough versus movement of the monsoon trough) rather than fundamental disagreements about the answer. [5]

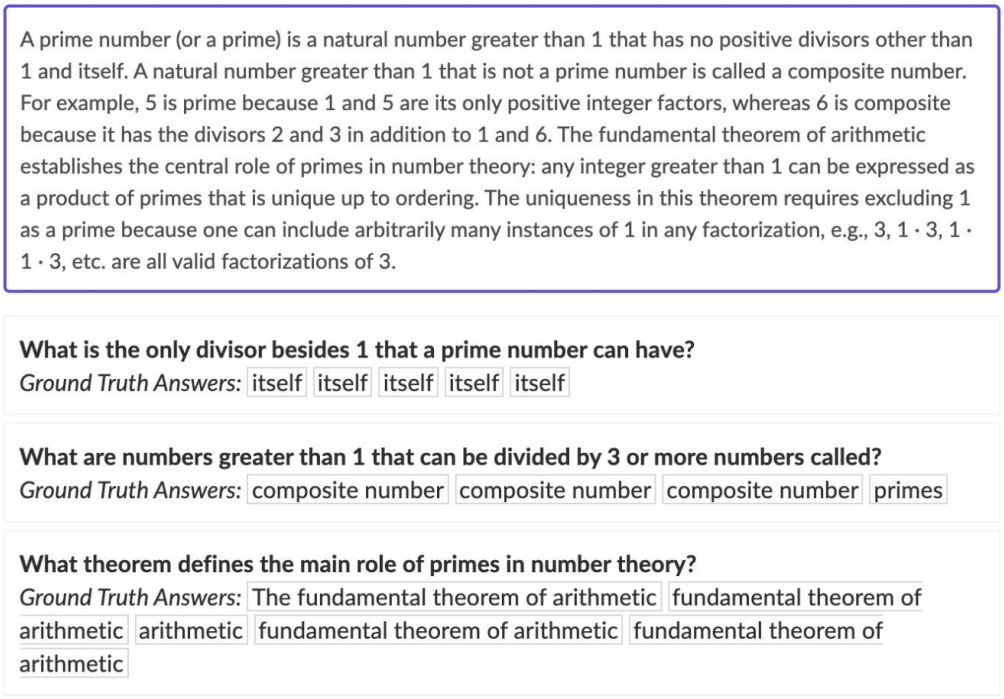

Figure 1: Example of contexts and questions in SQuAD dataset.

3 Models and Methods

3.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a powerful language representation model introduced by Google in 2018 that brought significant advances to the field of NLP.

It is built upon the Transformers architecture, which is the foundation of most modern NLP models due to their flexibility, efficiency, and superior performance across a wide range of tasks. Transformers rely on Attention mechanisms to weigh the importance of each word relative to others in the sequence. A basic transformer consists of an encoder-decoder structure, where the encoder processes the input sequence, and the decoder generates the output sequence. Each encoder and decoder are made up of layers containing self-attention mechanisms and feedforward neural networks. In models like BERT, only the encoder is used.

Unlike earlier models that processed text in a single direction - either left-to-right or right-to-left - BERT is designed to understand the context of a word based on both its left and right surroundings simultaneously. This bidirectional capability allows BERT to grasp deeper and more nuanced meanings within sentences.

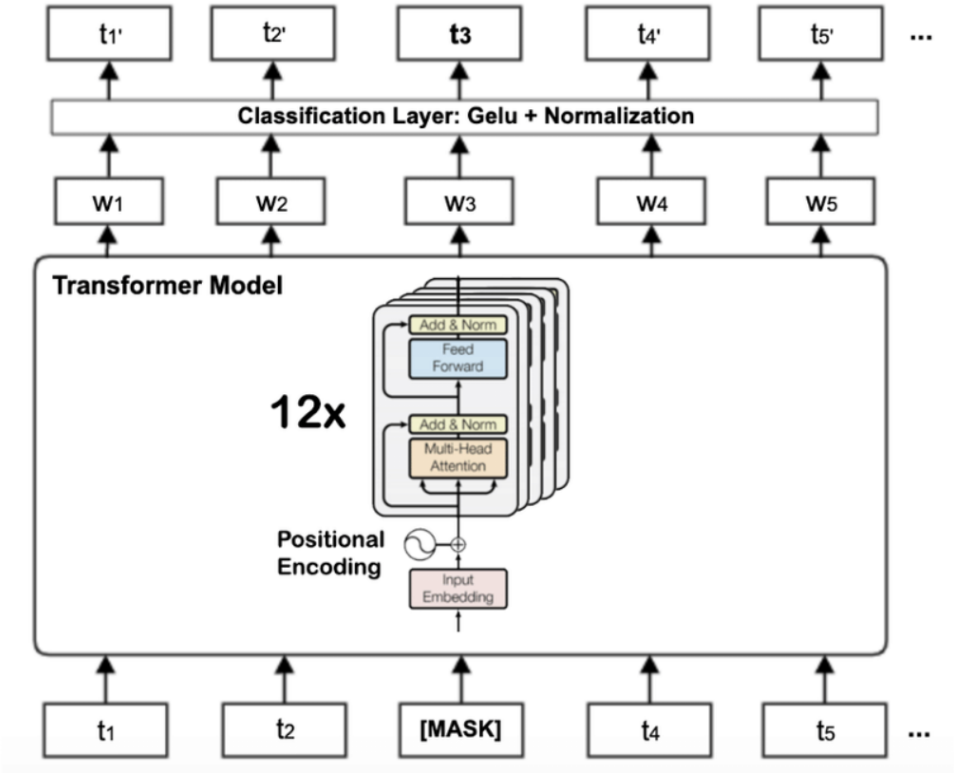

Figure 2: Transformer based BERT base architecture with 12 encoder blocks.

3.2 BERT's Variants

3.2.1 RoBERTa

RoBERTa, which stands for Robustly Optimized BERT Approach, is an improved variant of the original BERT model, introduced by Facebook AI in 2019. It retains the same underlying architecture as BERT, but modifies the training procedure to make the model more robust and powerful. The main motivation behind RoBERTa was to explore how BERT's performance could be enhanced simply by optimizing its training process, without changing the structure of the model.

One of the key differences is that RoBERTa removes the Next Sentence Prediction (NSP) objective used in BERT's pre-training. Research showed that NSP did not significantly help and could even harm performance, so RoBERTa was trained solely on the Masked Language Modeling (MLM) task. Additionally, RoBERTa uses dynamic masking, which generates different masked tokens for each training epoch, whereas BERT uses static masking, which keeps the same masked tokens throughout training. This change leads to more diverse training examples and better generalization.

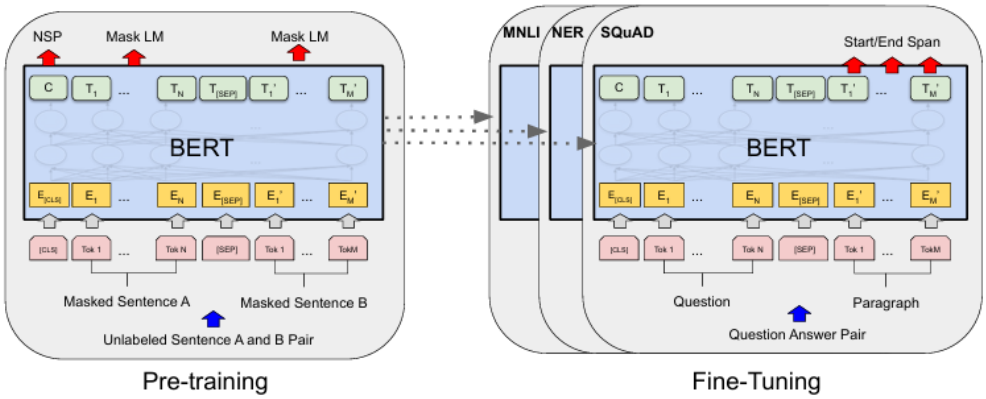

Figure 3: Overall pre-training and fine-tuning procedures for BERT.

RoBERTa also benefits from being trained on much more data. While BERT was trained on 16GB of text (BooksCorpus and English Wikipedia), RoBERTa used a significantly larger corpus of over 160GB, including data from sources such as Common Crawl and OpenWebText. Moreover, RoBERTa was trained for more steps with larger batches, which allowed it to learn more robust language representations.

3.2.2 SpanBERT

SpanBERT is a variant of BERT specifically designed to better capture and represent spans of text, rather than just individual tokens. Introduced by Joshi et al. in 2020, SpanBERT modifies BERT's pre-training process to improve performance on tasks that involve understanding relationships between text spans, such as question answering, coreference resolution, and named entity recognition.

The main difference between SpanBERT and the original BERT lies in the pre-training objective. While BERT uses Masked Language Modeling (MLM) by randomly masking individual tokens and training the model to predict them, SpanBERT replaces spans of contiguous text (i.e., multiple consecutive tokens) with a single mask token and trains the model to predict the entire span. This is known as the Masked Span Prediction (MSP) objective. Because of this, SpanBERT encourages the model to develop a deeper understanding of how groups of words work together semantically, rather than just focusing on single-word prediction.

Another key addition of SpanBERT is the use of a span boundary objective, where the model is trained to predict the content of a masked span using only the information from the tokens at the span's boundaries. This further improves the model's ability to encode span-level representa-

tions. Despite using the same architecture as BERT, these changes in the training objective make SpanBERT more effective for tasks that require reasoning over phrases or segments of text, rather than just individual word-level understanding.

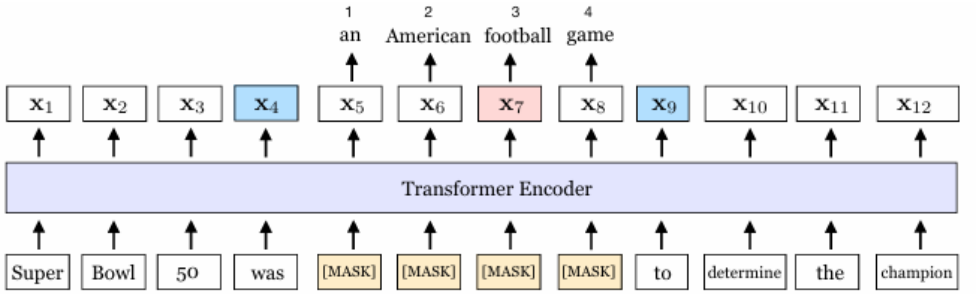

Figure 4: An illustration of SpanBERT training. The span boundary objective (SBO) uses the output representations of the boundary tokens, x4 (in blue), to predict each token in the masked span.

3.2.3 ELECTRA

ELECTRA [1] (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is a language representation model introduced by Google Research in 2020, designed to improve the efficiency and effectiveness of pre-training compared to previous NLP models. While it uses the same transformer-based architecture as BERT, ELECTRA introduces a novel training method called replaced token detection, which makes it significantly more sample-efficient and capable of achieving strong performance even with less training data and computation.

The core methodology of ELECTRA involves a two-part transformer setup, which consists of a generator and a discriminator. The generator, which is a small masked language model, predicts replacements for a subset of randomly masked tokens in the input text. Instead of training the model to predict the correct tokens directly, ELECTRA uses these predictions to create a corrupted version of the input sentence by replacing the original tokens with the generated ones. The modified sentence is then passed to the discriminator, a larger transformer model, which is trained to classify each token as either "original" (unchanged from the input) or "replaced" (generated by the generator).

This replaced token detection task offers a much richer learning signal than MLM because it enables the discriminator to make predictions over every token, not just the masked ones. As a result, ELECTRA is significantly more sample- and computation-efficient, achieving impressive performance on many NLP benchmarks while using fewer training resources. Additionally, because

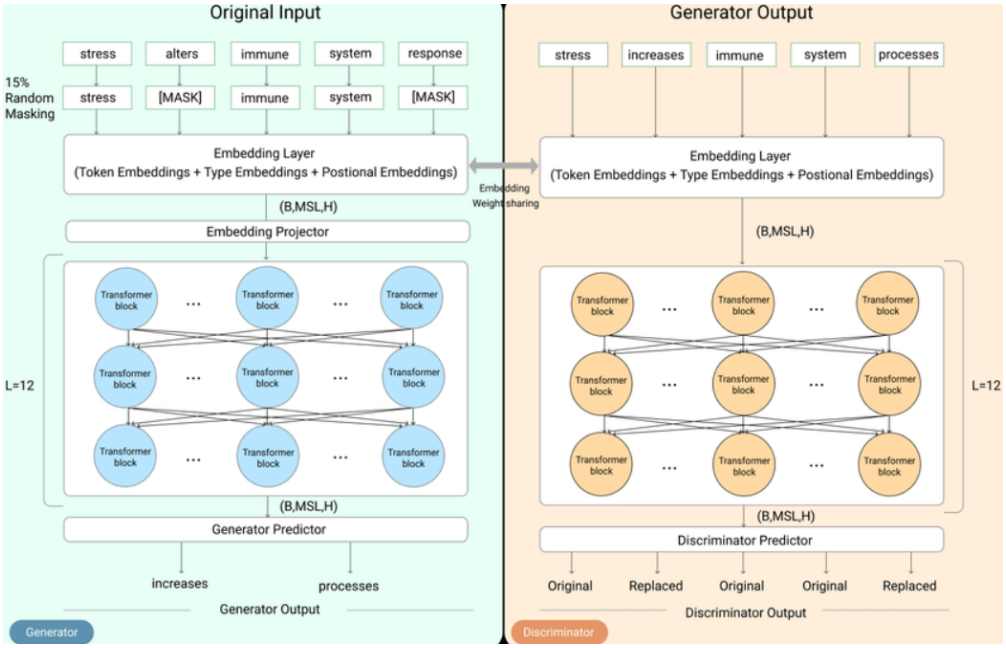

Figure 5: ELECTRA's architecture.

the generator is only needed during pre-training and is deliberately kept small, the fine-tuned ELECTRA model remains lightweight and fast during inference.

Another key advantage of ELECTRA's methodology is its flexibility and scalability. By focusing on detecting subtle token-level differences, the model learns more robust contextual representations, making it particularly effective for downstream tasks such as sentiment analysis, question answering, and text classification.

3.3 LoRA

LoRA [2], or Low-Rank Adaptation, is a technique developed to enable efficient fine-tuning of large pre-trained models, particularly transformer-based architectures such as BERT, GPT, or LLaMA. As large language models continue to grow in size and complexity, the cost of full fine-tuning - in which all parameters are updated - has become increasingly prohibitive in terms of both computation and memory. LoRA addresses this challenge by significantly reducing the number of trainable parameters needed to adapt a model to a new task and, as a result, lowering the resource requirements while maintaining high performance.

The central idea of LoRA is to freeze the original pre-trained model weights and introduce a small number of trainable parameters in the form of low-rank matrices into specific layers of the model, typically the attention and feed-forward layers of transformers. In standard transformer

layers, weight matrices - especially those in query, key, and value projections - tend to be large. LoRA modifies these by expressing the weight update as a product of two smaller matrices: one projecting from the original dimension to a lower dimension, and another one projecting back. Mathematically, if W is a weight matrix, LoRA approximates the update as

$$\Delta W = AB,$$

where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$, and r is much smaller than d and k. These matrices are initialized in a way such that their product is close to zero at the beginning of training, to ensure that the model behaves like the original until learning occurs.

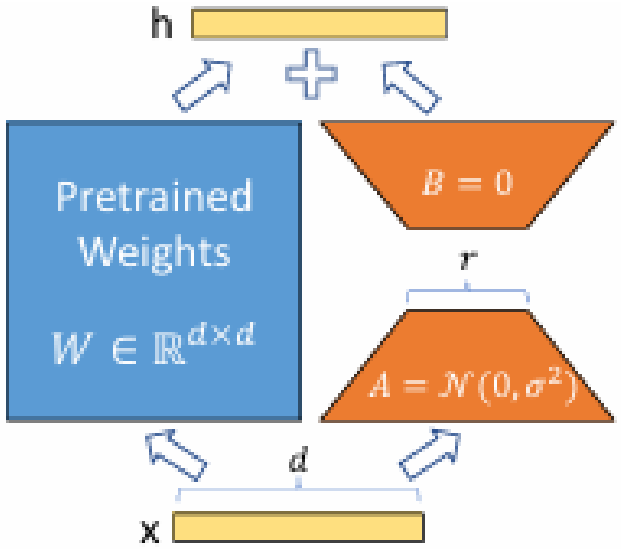

Figure 6: LoRA reparametrization. Only A and B are trained, instead of pretrained weights W.

This approach allows LoRA to introduce task-specific adaptability with minimal impact on memory and computation. Because only the newly added low-rank matrices are updated during fine-tuning, the number of trainable parameters is drastically reduced - often by orders of magnitude compared to full fine-tuning. Furthermore, LoRA can be deployed in a modular fashion: different sets of low-rank adapters can be trained for different tasks and swapped in and out of the base model as needed, which enable efficient model reuse across multiple domains.

In practice, LoRA has demonstrated performance comparable to, and sometimes exceeding, full fine-tuning across a variety of NLP benchmarks. It has become a widely used component in parameter-efficient fine-tuning (PEFT) frameworks, particularly in environments where computa-

tional efficiency, scalability, and task adaptability are priorities.

4 Experiments and Results

4.1 Setup

4.1.1 Data Preparation

The data preparation process for training and evaluating models on the SQuAD dataset involves several critical steps to ensure alignment between token-level inputs and span-level supervision:

- **Dataset Loading**: The dataset is typically provided in JSON format and is partitioned into training and validation subsets. Standard libraries such as the Hugging Face datasets library are commonly used to parse and manage the dataset. For each example, the context, question, and corresponding answer(s) are extracted.

- **Dataset Splitting**: We split the original train set into two sets: 80 percents for training, and 20 percents for validation, and use the original validation set for testing.

- **Feature Construction**: Each tokenized input is converted into a set of model-ready features, including input ID, attention mask, and, where applicable, token type ID. Additionally, the start and end token indices of the answer span are included as training labels.

- **Tokenization**: Both the question and the context passage are tokenized using a subword tokenizer corresponding to the pre-trained model (e.g., RoBERTa, SpanBERT). To accommodate the input constraints of transformer architectures, the question and context are concatenated into a single sequence and truncated or split to respect the model's maximum sequence length, which is 512 tokens in this case. If the combined length of the question and context exceeds 512 tokens, it will be truncated accordingly. Also, retaining the full question is typically more important than retaining the entire context, and contexts are often longer and may require truncation to fit within the model's limit, so we make sure that only the context is truncated.

- **Padding**: We pad all sequences to the fixed length of 512 tokens, regardless of their original length. This uniform length simplifies batching and is standard practice during training or evaluation when using models with fixed input sizes.

- **Answer Span Alignment**: It is necessary to align the character-level start and end positions of the answer in the context with the corresponding token indices after tokenization. This mapping is performed using the tokenizer's offset mappings, which record the start and end character positions of each token relative to the original text. Instances for which the answer span does not fall within the current window are omitted from training.

4.1.2 Training Configuration

The training configuration encompasses a range of hyperparameters and environment settings that govern the behavior and efficiency of the training process:

- **Number of Training Epochs**: Since it takes around 1 hour for a full training epoch, and pre-trained transformers like BERT converge relatively quickly when fine-tuned, we set number of epochs to 5 for decent performance.

- **Learning Rate**: Initial learning rate is 2e-5, and use Cosine Annealing schedule to reduce learning rate in a smooth, non-linear fashion, allowing for larger updates at the beginning and finer adjustments toward the end of training.

- **Loss Function**: Default loss of pretrained Transformers model, which is the average of two cross-entropy losses: one for predicting the start position of the answer span and another for predicting the end position. This loss was optimized using the AdamW optimizer.

- **Gradient Accumulation**: Use step size of 2, which allows the model to simulate a larger batch size without requiring more GPU memory. This makes the effective training batch size 32.

- **Additional hyperparameters for LoRA method**: In LoRA (Low-Rank Adaptation), the two most important hyperparameters are r and α (alpha). These hyperparameters control the capacity and scaling of the learned weight adaptation, and tuning them appropriately is crucial for achieving good performance while maintaining parameter efficiency.

  - **r (rank of the low-rank decomposition)**: The parameter r determines the rank of the low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ used to approximate the update to a full weight matrix $W \in \mathbb{R}^{d\times k}$. A smaller r leads to fewer trainable parameters and better efficiency, but may limit the expressiveness of the update. Conversely, a larger r increases capacity but consumes more memory and compute.

– α **(the scaling factor)**: The parameter α is a **scaling factor** applied to the low-rank update AB. Since AB is usually initialized with very small values (close to zero), scaling helps control the strength of the adaptation during training. The actual update applied to the model is:

$$\Delta W = \frac{\alpha}{r} \cdot AB$$

This normalization ensures that as r increases, the scale of the update remains controlled.

– After many tests, we decided to use r = 64 and $\alpha = 1024$ for better performance, and avoid unnecessary increase in training time.

4.1.3 Evaluation Method

The final goal of the model involves not only predicting the most likely answer spans, but also mapping those predictions back to the original context for comparison against human-annotated ground truth answers. Because of this, we conducted evaluation using two primary metrics: Exact Match (EM) and F1 score.

- **Exact Match (EM)**: measures the percentage of predictions that match any one of the ground truth answers exactly, after normalizing both the predicted and actual spans. This normalization process includes steps such as removing punctuation, lowercasing, and stripping articles (like "a," "an," and "the"). It is a strict measure, which provides a binary judgment of correctness - either the prediction is identical to a reference answer or not.

- **F1 Score**: evaluates the overlap between the predicted and ground truth answer spans at the token level. It is the harmonic mean of precision (the percentage of predicted tokens that are correct) and recall (the percentage of ground truth tokens that are correctly predicted).

$$\text{Precision} = \frac{|\hat{A} \cap A|}{|\hat{A}|}, \quad \text{Recall} = \frac{|\hat{A} \cap A|}{|A|}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Unlike EM, the F1 score provides a graded assessment of partial correctness, which is particularly useful when there are multiple valid phrasings or partial matches.

4.2 Results

Model	EM	F1	Inference Time(s)	Train Time(hrs)	Trainable Parameters(%)
Full Training					
SpanBERT	83.08	90.49	0.016	5 hrs 9 min	100%
RoBERTa	84.64	**91.29**	**0.015**	5 hrs 10 min	100%
ELECTRA	**84.67**	91.15	0.016	5 hrs 32 min	100%
LoRA					
SpanBERT	82.23	89.80	0.018	**4 hrs 26 min**	4.20%
RoBERTa	84.41	91.10	0.023	**4 hrs 26 min**	3.67%
ELECTRA	84.37	90.91	0.032	4 hrs 50 min	4.15%

Table 1: Results on evaluation set of SQuAD 1.1

Under the full fine-tuning approach where 100% of the parameters were trained, all three models achieved high performance. ELECTRA recorded the highest Exact Match (EM) score at 84.67, while RoBERTa obtained the top F1 score of 91.29. SpanBERT also demonstrated strong results with an EM of 83.08 and an F1 score of 90.49. In terms of efficiency, RoBERTa had the lowest inference time at 0.015 seconds per example, though training times for all models were comparable, with the slowest being ELECTRA at 5 hours and 32 minutes.

When using the LoRA fine-tuning method with α=64 and r=1024, there was a significant reduction in the number of trainable parameters—down to 4.20% for SpanBERT and 3.67% for both RoBERTa and ELECTRA. This led to a notable decrease in training time; for example, the training duration for RoBERTa was reduced from 5 hours 10 minutes to 4 hours 26 minutes. This gain in efficiency came with only a minimal trade-off in performance. The RoBERTa model adapted with LoRA produced the best results in this category, with an EM of 84.41 and an F1 score of 91.10. A slight increase in inference time was observed for all LoRA-adapted models.

5 Discussion

The results provide valuable insights into the trade-offs between performance and computational efficiency in Machine Reading Comprehension. The full fine-tuning approach, while delivering the highest performance scores, is also the most resource-intensive in terms of training time. RoBERTa and ELECTRA slightly outperformed SpanBERT in the full fine-tuning setting, which may be attributed to RoBERTa's robust pre-training optimization and ELECTRA's highly efficient pre-training task.

The LoRA strategy stands out as an exceptionally effective parameter-efficient fine-tuning

method. By training as few as 3.67% of the total parameters, LoRA substantially reduces the computational resources required. This efficiency is clearly demonstrated by the shorter training times; for instance, RoBERTa's training was over 40 minutes faster.

Most importantly, this efficiency is achieved with a negligible decrease in performance. RoBERTa with LoRA saw its F1 score drop by only 0.19 (from 91.29 to 91.10) compared to the fully fine-tuned version. This proves that it is possible to effectively adapt large models for specific tasks without updating all their parameters, making LoRA a compelling choice for environments with limited computational resources. The minor increase in inference time is an expected consequence of processing the additional low-rank matrices during the forward pass and is a small price to pay for the significant gains in training efficiency.

6 Conclusion

This project successfully investigated the performance of three popular BERT-based models—RoBERTa, SpanBERT, and ELECTRA—on the SQUAD 1.1 benchmark for Machine Reading Comprehension. We compared the effectiveness of traditional full fine-tuning against the parameter-efficient LoRA method.

Our experiments show that while full fine-tuning yields the best performance, particularly with RoBERTa achieving a 91.29 F1 score, the LoRA technique offers a remarkable balance between performance and computational efficiency. By drastically reducing the number of trainable parameters to as low as 3.67%, LoRA decreased training times by up to 14% while having a negligible impact on EM and F1 scores. These findings confirm that parameter-efficient fine-tuning methods are a viable and practical solution for adapting large-scale language models in resource-constrained settings without a significant compromise in quality. The consistent high performance of RoBERTa across both tuning methods further highlights the benefits of its robustly optimized pre-training approach for downstream tasks.

References

[1] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555, 2020.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.

[3] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529, 2019.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.