Final Report: From Headlines to Highlights: Boosting Summarization by Fine-Tuning Language Models on Mainstream News Data

Team Members:
Christian Kobriger (s5232686) - c.kobriger@student.rug.nl
Łukasz Sawala (s5173019) - l.h.sawala@student.rug.nl
Csenge Szőke (s5139090) - c.szoke@student.rug.nl

GitHub Repository:
`https://github.com/03chrisk/PEFT-T5-on-CNN-dailynews`

September 21, 2024

Contents

1 Abstract

This project investigates fine-tuning techniques to enhance the summarization capabilities of the T5-base model on the CNN/Daily Mail dataset. Given the model's limited baseline performance in generating abstractive summaries, we applied instructional prompting, parameter-efficient fine-tuning with LoRA, and full fine-tuning. Each method was evaluated based on improvements in ROUGE, BLEU, and BERT scores. Results indicate that prompting alone yielded a modest performance boost, while combining prompting with LoRA significantly enhanced phrase overlap and fluency, with a 27% average increase across metrics. Full fine-tuning demonstrated the highest performance, achieving a further 5% improvement over LoRA, suggesting superior adaptability in generating coherent and relevant summaries. Despite these gains, the model remains below state-of-the-art benchmarks, highlighting the computational limitations faced. This work highlights the effectiveness of fine-tuning techniques and points toward future research directions to bridge the gap with larger, specialized language models.

2 Introduction

The rapid expansion of data in the contemporary digital world has outpaced our ability to process it effectively. With an overwhelming volume of information generated daily, there is an urgent need for new technologies able to make this information available in a user-friendly manner. With the recent rise of deep learning, multiple solutions have been proposed - from bottom-up approaches like intelligent search engines to direct top-down solutions like efficient summarization. "The volume of data around us is growing to the point that we need to find a solution that will deliver accurate and timely summary information."(Rehman et al., 2022) Automatic text summarization is a possible solution in addressing this overwhelming volume of information as it allows users to extract meaningful insights from a large amount of data quickly.

2.1 Text Summarization

Abstractive text summarization is a challenging task in natural language processing (NLP) that involves generating concise summaries by paraphrasing the original content. Unlike extractive summarization, which selects key sentences directly from the source text, abstractive summarization requires a model to generate new sentences that capture the important points of the original text.

Recent advancements in large language models (LLMs) have dramatically reshaped this field of NLP, pushing the boundaries of what automatic summarization systems can achieve. Several state-of-the-art architectures have been developed, each addressing different challenges associated with the task. Autoregressive models, pre-trained encoder-decoder architectures, and instruction-tuned models have emerged as leading approaches. These models vary in terms of their architecture, performance, and computational requirements, which has led to active exploration and comparison within the research community.

2.2 Project Scope

This project aims to fine-tune the T5 model on the CNN-dailymail dataset to evaluate if its performance in abstractive text summarization can be enhanced compared to the pre-trained results reported in the paper "An Analysis of Abstractive Text Summarization Using Pre-trained Models."(Rehman et al., 2022) Using widely-used metrics like ROUGE and BLEU, it will be evaluated whether fine-tuning can significantly enhance the quality of generated summaries, building on the work of previous research in the field (Rehman et al., 2022).

3 Review of Relevant Literature

3.1 State-of-the-Art Models for Abstractive Summarization

The field of abstractive text summarization has seen significant advances, driven by the development of various architectures and model types. These include pre-trained encoder-decoder models, large autoregressive language models, and instruction-tuned models, as can be seen in Figure 1 each with distinct characteristics and advantages (Retkowski, 2023).

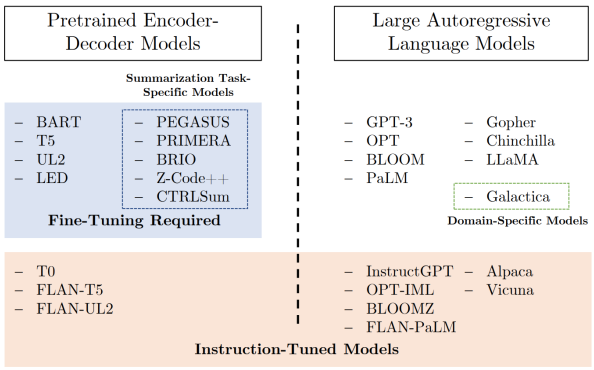

Figure 1: Broad Classification of current state-of-the-art models for abstractive summarization (Retkowski, 2023).

Pre-trained Encoder-Decoder

Pre-trained encoder-decoder models have become a fundamental part of modern summarization systems. These models are typically trained on large, unlabeled datasets in a self-supervised manner and then fine-tuned on task-specific datasets. One of the most widely used models in this category is BART (Lewis et al., 2020), a denoising autoencoder with a transformer-based encoder-decoder structure. BART has approximately 400 million parameters and can generate coherent and concise summaries in various domains. Its performance is robust in human and automatic evaluations, but it generally requires fine-tuning for task-specific applications.

T5 (Raffel et al., 2023), another prominent model, uses a similar architecture to BART but generalizes tasks using a unified text-to-text framework, a modeling approach where all tasks, regardless of their nature, are framed as text-based input-output tasks. The T5 family includes models of varying sizes, ranging from 60 million to 11 billion parameters, with T5-Large containing 770 million parameters. Although T5 shows strong transfer learning capabilities in several NLP tasks, including summarization, it also benefits from fine-tuning for optimal performance. The availability of many different sizes makes T5 more flexible than BART, particularly in multi-task scenarios.

PEGASUS (J. Zhang et al., 2020), on the contrary, is explicitly designed for summarization tasks. Its unique pre-training objective, Gap Sentences Generation, enhances its ability to generate abstractive summaries without the need for significant fine-tuning. With 568 million parameters, PEGASUS consistently outperforms BART and T5 in summarization-specific tasks, especially in low-resource settings, which is why this model is a great choice for abstractive summarization. However, it lacks the versatility of models such as BART and T5, which perform well in a broader range of tasks.

Instruction tuned Models

Instruction-tuned models are a relatively new development in abstractive summarization and have shown to be quite promising, especially in zero-shot learning scenarios. These models are fine-tuned on a diverse range of tasks using natural language instructions, allowing them to generalize better to new tasks without requiring task-specific fine-tuning (Chung et al., 2022).

FLAN-T5 (Chung et al., 2022) is an instruction-tuned version of T5 that performs exceptionally well in zero-shot summarization. By training on a broad range of instruction-based tasks, FLAN-T5 can adapt to new tasks more easily than standard T5, without needing additional fine-tuning.

3.2 T5 Model in a Wider Context

T5, or Text-to-Text Transfer Transformer, is a series of Large Language Models (LLMs) developed by Google to handle any potential natural language processing task (Raffel et al., 2023). This model has been developed to excel at transfer learning, a subtype of Machine Learning where we use knowledge from a previous training process in training a new skill, as is done in the pre-training→fine-tuning paradigm, a standard approach used in LLMs. To excel at as many tasks as possible, the model is based on an encoder-decoder architecture, allowing it to learn deep bidirectional encodings and excellent content generation.

As LLMs have been a hot field of research and development for the past few years, even though T5 is a relatively new model (2019/2020), it is already starting to fall behind the more recent models, which are developed differently to prepare them for a particular task. However, it is still one of the most widely used LLMs, as it comes in multiple versions and sizes while being able to learn any task relatively well.

3.3 Training and Evaluation Challenges & Methods

Because of its unique properties, training and evaluation of text summarization has always posed a challenge that does not have a fit-for-all solution. This stems from the issues described in the Challenges

of Text Summarization subsection, but mainly because of the subjectivity of most of the evaluations. To try to mitigate this issue and develop maximally standardized and human-aligned evaluations, multiple solutions have been proposed.

In general, the evaluation techniques can be split into two categories: Reference-based evaluation, matching the generated summary with labels in the dataset, checking for similarities (label-based), and Reference-free evaluation, matching the generated summary to the text used for the generation process, checking for i.e. information loss or hallucinations (label-free).

Most evaluation methods can be applied in two ways: manually or automatically. Manual methods consist of human labeling (rarely used due to scalability problems) or process control (adjusting metrics during training to address i.e. hallucinations or lack of clarity) and are widely used in LLM training. More generally, any evaluation can be applied by the use of LLM-based methods, for example, providing the summary made by the model, the reference, and a query (i.e. "evaluate the clarity of a summary") to an already fine-tuned, general-purpose model i.e. GPT 4 (Chiang & Lee, 2023; Shakil et al., 2024; Wu et al., 2023). The more conventional (but still widely used), automatic methods are described in the methods section.

An overview of those methods and more can be found in Figure 2.

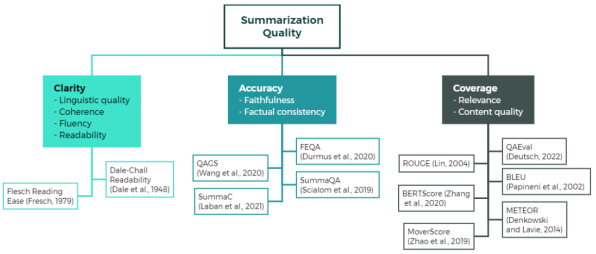

Figure 2: Evaluation Methods for Text Summarization, Nguyen et al., 2024

Research has strongly emphasized the reference problem - creating reference summaries is expensive, slow, and leads to bias, while not using them leads to multiple issues in training. Some studies even reach extreme conclusions, suggesting that no metric is related to human satisfaction no matter what model is used with those metrics (Nguyen et al., 2024) and hence, LLM-based evaluation is the only automated method that can be applied.

Because of the abovementioned problems, even though automatic methods are integral to efficient training on large datasets, manual techniques combined with oracle-based architectures should always be considered in any text summarization task.

3.4 Fine-Tuning and Prompting Techniques

Fine-Tuning Techniques

As explained above, fine-tuning has become an essential method for optimizing LLMs for specific tasks, particularly as these models grow in size and complexity. Traditionally, smaller models like BERT and RoBERTa were fine-tuned entirely, which is called full fine-tuning. However, full fine-tuning larger models like T5-Large poses computational challenges. To overcome this, methods like Low-Rank Adaptation (LoRA) have been developed, as discussed by Jeong (2024), which freeze most of the model's pre-trained layers and only fine-tune new parameters. This approach reduces computational cost without significantly affecting performance on tasks. Such fine-tuning methods are called Parameter-Efficient Fine-Tuning (PEFT), where only a small set of parameters are fine-tuned. LoRA targets specific weight matrices within each transformer layer, decomposing them into low-rank matrices for more efficient fine-tuning. By doing that, this method selectively focuses on the most essential parameters thus minimizing computational costs.

Recent advances in fine-tuning include techniques such as prompt modification (hard and soft prompt tuning, and prefix-tuning), adapter methods like LLaMA-Adapter, and efficient parameterization techniques. An example is QLoRA, an enhanced version of LoRA, which allows LLM fine-tuning on personal computers by combining a high-precision computing technique with a low-precision storage method. This allows large models to run with limited memory, although with minor information loss (Jeong, 2024).

These techniques are useful alternatives to use instead of full fine-tuning. By targeting only the most important parameters in the transformer layers, they make it possible to adapt models to new tasks with significantly lower computational costs, making them essential for large-scale language models like T5.

Prompting Techniques

While fine-tuning is a commonly used approach to optimizing language models for specific tasks, prompting techniques are an efficient alternative. Prompting involves guiding the model's generation through designed input prompts, which can help in guiding the model's outputs without extensive retraining. Particularly in the domain of text summarization, prompting can improve model performance by directly steering the generation process towards task-specific goals. This technique assumes that a pre-trained model has seen so much data that it is able to understand language, hence instead of giving the model a text and expect it to make a summary by itself, we can hint what we want from it, by adding a sentence i.e. "generate a summary for this text. Focus on main stakeholders affected" to the standard input. This method has been inspired by human learning and allows for utilizing human intuition in model learning.

Incorporating prompting techniques into LLMs like T5 can make the task of summarization more flexible and resource-efficient and can reduce the need for extensive fine-tuning. Although prompting techniques, such as soft prompts, hierarchical chunking, and prompt chaining, are proven to be effective in guiding large language models for specific tasks like abstractive summarization, they have certain limitations. Current prompting strategies often struggle with maintaining coherence and reasoning across complex tasks with multiple steps. This is mostly because of a lack of effective integration between problem understanding, reasoning, and iterative refinement, which may lead to inconsistency when handling more demanding tasks (Lingo et al., 2024).

4 Methods

4.1 Model Architecture

As mentioned above, the model we are going to use is the T5-Base model(Raffel et al., 2023) from the HuggingFace transformers library. This matches the model used in the baseline paper (Rehman et al., 2022). This model is the most popular in the Text-to-Text Transfer Transformer (T5) family, with only 220 million parameters, hence requiring less computational resources for fine-tuning.

T5 is an encoder-decoder model, pre-trained using both supervised and self-supervised learning on a wide variety of NLP tasks, such as translation, classification, and summarization. The input to both the encoder and decoder is always in the form of text, which is why T5 is referred to as a text-to-text model.

The encoder is responsible for processing the input text by converting it into a set of continuous representations that capture the meaning and context of the entire input sequence. It uses self-attention mechanisms to analyze relationships between the tokens, which will allow the model to understand the overall structure and significance of the input.

The decoder generates output text by predicting one token at a time based on the encoded representations from the encoder and the previously generated tokens. It employs cross-attention to incorporate information from the encoder while using its own autoregressive mechanism to create coherent and contextually relevant outputs.

4.2 Prompting

As highlighted in the Literature Review, there are multiple prompting techniques applicable to any LLM training. Therefore, it is really important to choose the prompting technique that strikes a good balance between fitting the task, the model, and the data. As mentioned in the previous subsection, the T5 models are pre-trained on multiple tasks to ensure high adaptability. The standard choice of prompting for that specific model family is Task-specific Prompting (adding a prefix "Summarize: " to every summary), as this is one of the ways in which those models are pre-trained. Another slightly more complex method could be Instructional Prompting: adding "Provide a detailed and concise summary of this article: ", which is more informative because it clearly directs the model to perform a specific task—in this case, summarizing an article. Those were the two methods we tested for in our research.

4.3 Full Fine-Tuning

The most standard way of fine-tuning pre-trained models is called full fine-tuning. This allows for adjustment of all of the model parameters, which can be very slow and inefficient. However, since this method is simple and does not make any assumptions about model inference, it is good practice to apply it if computational resources are available, hence we decided to implement it in this project as well.

4.4 Parameter Efficient Fine-Tuning

Due to computational inefficiencies of full fine-tuning, a class of alternatives has been proposed and widely approved within the field. These alternative methods, called Parameter-Efficient Fine-Tuning (PEFT) methods operate only on a subset of the parameters (as described in the Literature Review). The model we decided to use is LoRA (Low-rank adaptation) due to its simplicity and popularity.

4.5 Training Techniques

To train our models we decided to use the most popular loss function for text comparisons: the cross-entropy loss. The formula for cross-entropy loss is given by:

$$\text{Cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|x, y_{1:i-1})$$

Where N is the number of tokens in the target sequence, y_i is the correct token at position i in the target sequence and x is the input.

To increase the efficiency of the training we used two optimizers - ADAM and a Linear Learning Rate scheduler.

ADAM (Adaptive Moment Estimation)

ADAM is an optimizer algorithm applied to gradients used in training to improve the convergence and efficiency of the process. It combines two techniques: RMSprop and Gradient Descend with Momentum. RMSprop (Beta2) helps to stabilize learning by adjusting the learning rate based on the magnitude of the previous gradients, while momentum (Beta1) combines previous gradients with the current one to smooth out learning. The ADAM optimizer was obtained from the torch.optim library. The specific formula used by ADAM to adjust the values of the gradient is complex and out of the scope of this project. A curious reader should look up the Pytorch library documentation instead.

Learning Rate (LR) Scheduler

Independent from ADAM's RMSprop, we use a linear LR Scheduler with warmup (initial number of steps when the LR is not adjusted) from HuggingFace transformers library. The scheduler is used to gradually decrease the learning rate to reduce the noise in learning in the latter parts of training. The decrease is done based on a basis function, defined in the arguments of the scheduler, with linear being the most popular (used in this project as well).

Hyperparameter tuning

Because of limitations in available computational resources, which will be described in the Discussion section, standard hyperparameter search methods like grid search or random validation were not viable in this project. We acknowledge those limitations and expect that better results can be obtained with higher time and computation availability.

Gradient Accumulation

As a result of the above-mentioned limitation, a maximum of batch size of 8 could be afforded while avoiding memory issues. To combat this problem, the effect of batching on gradient variance was simulated by applying Gradient Accumulation (GA), which is a common technique used in LLM training. GA simulates batch training by waiting until a few smaller batches are computed and combining their losses before backpropagating the error, leading to the same effect as standard batching while requiring a fraction of GPU memory.

Data Collator

Applying a data collator from the HF transformers library was chosen as another adjustment to reduce the usage of computational resources. Data collators are used as a way to reduce the input dimensionality as much as possible by applying padding over each batch rather than over the whole dataset. This is done because in the latter case, each text is padded to a maximum length equal over the whole dataset, leading to inefficient memory usage as the length of the inputs has high variance.

Early Stopping

Lastly, to reduce training time as much as possible, early stopping with patience was applied, automatically stopping the training if no improvement is observed in the validation metrics.

5 Experiments and Results

5.1 Data

As highlighted multiple times in this paper, abstractive summarization is a complex task that requires the model to extract deep meaning from the text to

choose which pieces of information are relevant and should be extracted.

Therefore, the learning process is necessarily highly complex and requires learning small nuances related to semantics. It is hence required that the dataset used for fine-tuning any model is complex enough to capture those nuances and do so from multiple angles. Based on these requirements it has been decided to use the CNN-dailymail dataset, which is a popular dataset consisting of over 300000 unique samples of news texts, written in English. The texts in the dataset are long and complex, hence it is mainly used for tasks based on Q&A and summarization. The dataset also contains original labels in the form of short highlights (2-4 sentences) linked with the texts. The distributions of lengths of articles and highlights can be seen in Figures 3 and 4.

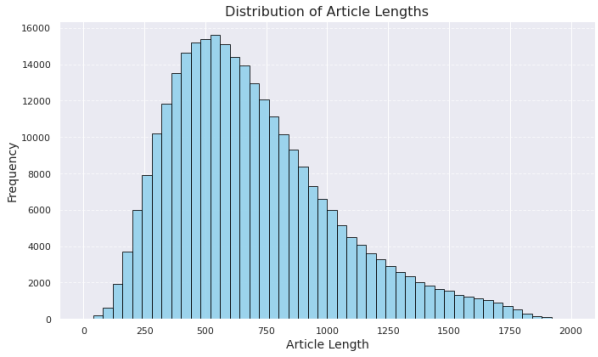

Figure 3: Histogram of Article Lengths

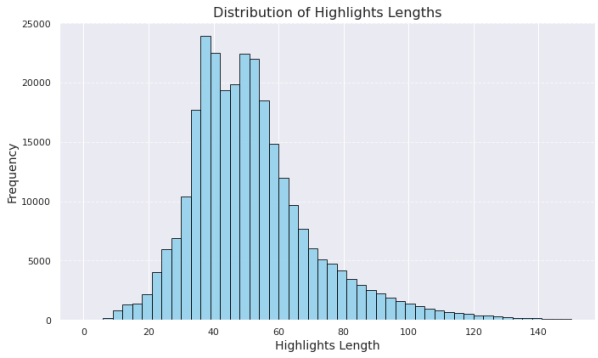

Figure 4: Histogram of Highlight Lengths

Abstractive summarization poses a challenge to NLP models, requiring a deep and general understanding of human language and its nuances like sarcasm or subtexts. News texts also include a lot of redundant information used to engage with the reader, posing challenges to transformers. Furthermore, it requires a very thorough handling of long-term dependencies, which is still one of the main metrics used to evaluate NLP models. Therefore, we expect substantial improvement in the model after fine-tuning this dataset.

Due to the limitations of the study, only a subset (around 10%) of the data was chosen to perform fine-tuning. This issue is thoroughly explained in the limitations section. It is important to note that the test set was used in its entirety, to ensure that the final evaluations of the models are comparable to benchmark values.

5.2 Dataset limitations

An attentive reader might notice that the labels in our dataset are not the true labels of that task - the highlights tend to be shorter than summaries and contain more general information i.e. by skipping names. Therefore, it could be argued that abstractive summarization cannot be learned by applying the dataset. This statement however rational is not valid because of a few reasons. First of all, the task of fine-tuning is only used to adjust the weights of the existing model, making it excel at a particular task rather than on general language processing. Therefore, most of the knowledge obtained during pre-training is preserved. Furthermore, the models from the T5 family are prompted during pre-training and hence their performance on summarization tasks is already fairly good off the shelf. Therefore, we only want the model to change from a general summarizer to an expert concise one. In that manner, the labels are not the ultimate goal to be reached, but rather a strong nudge toward its direction. This is why this project is not aiming for a very low training loss but rather focuses on metric values and visual evaluations.

5.3 Data Preprocessing

The data preprocessing is an important step for preparing the CNN-DailyMail dataset for training the T5-small model for the abstractive summarization task. The preprocessing steps involved tokenizing the text, and formatting the data for efficient training. Below is a detailed description of the preprocessing workflow.

For the tokenization process, we employed the T5Tokenizer from the Hugging Face Transformers library. The tokenizer is responsible for converting raw text into a format suitable for the T5 model by transforming sentences into sequences of token IDs. To be able to properly tokenize the words, the

T5Tokenizer handles all the preprocessing steps such as lemmatization or trimming.

We defined a function to handle the tokenization process, where input articles were concatenated with the prompt and tokenized, ensuring that the resulting sequences did not exceed a maximum length of 1500 tokens (which leaves most of our dataset non-truncated while allowing for faster inference). Any excess tokens were truncated. The target summaries (highlights) were tokenized with a maximum length of 200 tokens.

After the tokenization process, we removed unnecessary columns from the tokenized datasets, specifically highlights, article, and id, leaving only the essential columns: input_ids, attention_mask, and labels. Finally, the datasets were formatted to use PyTorch tensors, making them ready for input into the model during training and evaluation.

As already mentioned in the paper, we use a data collator to facilitate efficient batch training. Specifically, the DataCollatorForSeq2Seq from the Transformers library was used.

5.4 Evaluation Metrics

The most widely used metrics for text summarization are ROGUE (Recall-Oriented Understudy for Gisting Evaluation) and BLEU(Bilingual Evaluation Understudy). ROGUE is a reference-based method that can be split into two types: ROGUE-N and ROGUE-L.

ROUGE-N is a group of methods matching variable size n-grams of the summary into the reference. For example, ROUGE-1 would check whether the words used in the summary are also used in the reference document. This metric has been developed for extractive summarization but is also used in the abstractive counter- part because of its simplicity. These can be further split into ROUGE-Nr or ROUGE- Np, which are based on recall or precision scores instead.

ROUGE-L Measures the longest subsequence contained in both the candidate summary and the reference summary.

BLEU is another reference-based method, consisting of a weighted average of multiple ROUGE-N scores. This score is an improvement over the standard ROUGE-based assessment as the resulting value is more general and can be used for evaluating the models more effectively.

Knowing the limitations of those metrics (as expressed in the Literature Review and Future Research), we decided to add an extra metric, BERTscore, which has been shown to correlate well with human judgment, as opposed to BLEU scores (T. Sun et al., 2022; T. Zhang et al., 2019). Conceptually, it is very similar to BLEU/ROGUE scores but focuses not on direct word matching, but on contextual representations produced by the model. Therefore, it shifts the focus from trying to force how the model chooses the words to match the reference, it tried to force the model to "reason" in a desired way while performing the task.

We conclude this section with benchmark values for the metrics mentioned in this section, as seen in Table 1.

Metric	Baseline Value	Source
ROUGE-1 (in %)	44-46	"CNN/Daily Mail Benchmark", 2024
ROUGE-2 (in %)	21-23	"CNN/Daily Mail Benchmark", 2024
ROUGE-L (in %)	40-43	"CNN/Daily Mail Benchmark", 2024
BLEU (in %)	35-45	"Google Cloud tutorial", 2024
BERTScore (F1-score)	0.71	T. Zhang et al., 2019

Table 1: Benchmark Values on chosen metrics, in a familiar scope

It is important to note that those are the most standard values of each metric for general models, as there are multiple versions of most of the metrics proposed here. The paper we build ours upon(Rehman et al., 2022) provides us with a different benchmark, as seen in Table 2.

	ROUGE-1			ROUGE-2			ROUGE-L			BLEU
	R	P	F1	R	P	F1	R	P	F1	
T5-base	33.96	30.35	31.10	13.37	11.67	12.00	31.43	28.10	28.79	55.70

Table 2: Summarization performance on CNN/DailyMail dataset(Rehman et al., 2022)

5.5 Experimental Details

Experiments

We conducted a total of four experiments to analyze the effects of different fine-tuning techniques, including instructional prompting, parameter-efficient fine-tuning, and full fine-tuning. The performance of the pre-trained T5-base model on the summarization task serves as our baseline. The 4 model configurations and their training parameters will be outlined in the following section.

Model Configurations

We evaluated the T5-base model in various configurations, outlined below:

- **Baseline Configuration:** The pre-trained T5-base model was used without any additional fine-tuning to establish a performance baseline on the summarization task.

- **Instructional Prompting Configuration:** To determine the impact of instructional prompting alone, we applied a prompt (e.g., "Summarize:") to each input text and evaluated the pre-trained T5-base model's performance on this modified input.
- **Parameter-Efficient Fine-Tuning (PEFT) Configuration:** Using LoRA (Low-Rank Adaptation), we fine-tuned the T5-base model by adjusting a subset of parameters. Key hyperparameters for LoRA included:
  - **Rank:** 16
  - **Alpha:** 32
  - **Dropout:** 0.1
  - **Learning Rate:** 1e-5
  - **Weight Decay:** 0.1
  - **Batch Size:** 6 per GPU (with gradient accumulation of 2 steps)

  These values were selected based on best practices for PEFT and optimized to balance performance and memory efficiency.
- **Full Fine-Tuning Configuration:** For this configuration, all parameters in the T5-base model were fine-tuned. Key hyperparameters included:
  - **Learning Rate:** 5e-5
  - **Weight Decay:** 0.001
  - **Dropout:** 0.1
  - **Batch Size:** 8 per GPU (with gradient accumulation of 2 steps)

Early stopping with patience of 3 was used to avoid overfitting in both the PEFT and full fine-tuning experiments.

5.6 Results

The evaluation of different conditions that were tested in this research can be seen in Table 3 (FT stands for fine-tuning).

Metric	Model			
	T5-base	**Prompting**	**LoRA**	**Full FT**
ROGUE-1	35.17	35.60	40.12	**41.73**
ROGUE-2	14.17	14.92	18.90	**19.10**
ROGUE-L	23.86	24.56	29.40	**29.63**
BLEU	11.18	11.57	18.18	**18.58**
BERT score	20.53	21.37	31.99	**32.10**

Table 3: Performance metrics for different T5-base model variations on the entire test set (11490 samples). For both LoRA and full fine-tuning, prompting was applied.

An example article combined with the summaries generated by all of the conditions tested in the project is provided in Appendix A.

Prompting

Applying prompting to the base model already improves its performance by around 3% on average over all of the metrics. This result suggests that the prompt does not heavily influence the summaries generated by the model, but applying it is still helpful in guiding the model. Therefore, the base model used in this research (T5-base) seems to require fine-tuning to be able to adapt to a task.

LoRA + Prompting

Applying LoRA on top of prompting leads to an additional increase of around 33% on average over all the metrics. The notable increases in ROGUE scores indicate a higher overlap in essential phrases and sentence structures between the generated summaries and the target references. The BLEU score, which reflects overall fluency and syntactic accuracy, also improved significantly, highlighting the model's ability to produce cohesive, human-like text. Lastly, a steep increase in the BERT score(by 49% compared to the prompting-only configuration), the most human-aligned metric in our evaluation, suggests that the model consistently produces summaries that human users would find more informative and relevant.

Full Fine-Tuning + Prompting

The results from applying full fine-tuning with prompting on the T5-base model show the highest performance across all evaluation metrics, marking a further improvement over the LoRA + prompting configuration. Full fine-tuning achieves an approximate 2% average increase in performance over LoRA across ROUGE, BLEU, and BERT scores, indicating superior phrase alignment, fluency, and semantic accuracy in generated summaries compared to all other conditions that were tested for in this study. The small gap of performance between this method and LoRA highlights the inefficiency of this method, as a similar metric score can be obtained while tuning a fraction of model parameters.

5.7 Analysis

While the model shows a remarkable improvement compared to the baseline metrics (especially for the BERT score, which is the most relevant metric used in this research), the results are still not overly optimistic when compared to the benchmark values used in the field, as seen in Table 1. However, as highlighted in the Literature Review, this result was to be expected as the only model currently able to compete with the state-of-the-art models is the largest version, T5-11b (with 11 billion parameters) with more than 50 times more tunable parameters. While in general ML, larger models do not automatically translate to better performance, it can be argued that in such a complex task as abstractive summarization, this increase in learning capabilities might be necessary.

Another thing to consider is that T5 is a family developed with general language processing models in mind and hence is not perfectly suited to handle this task in particular. Therefore, it is hard to directly compare a small version of a general model to current state-of-the-art specialized models with billions of parameters.

We conclude this section by stating that the increase seen after fine-tuning is substantial and with more resources, a wider study of models could be built upon our baseline to validate the concepts highlighted in this analysis. The code necessary to reproduce the baseline with new hyperparameters or datasets can be accessed through the GitHub repository of the project[1].

6 Conclusion

6.1 Discussion

The advancements driven by the development of Large Language Models (LLMs) such as T5 (Raffel et al., 2023), BART (Lewis et al., 2020), and PEGASUS (J. Zhang et al., 2020), as well as autoregressive models like GPT-3 (Brown et al., 2020) and instruction-tuned models like FLAN-T5 (Chung et al., 2022), have led to significant improvements in performance over a wide scope of NLP tasks, with summarization being one of them. Despite these models' successes, several gaps hinder optimal performance in various domains.

Our experiments in this project demonstrate that both prompting and fine-tuning techniques notably improve the performance of the T5-base model on abstractive summarization tasks compared to the baseline. While our models do not reach benchmark levels seen in large, task-specific models, these results illustrate that fine-tuning, even in a smaller, general-purpose model, can result in substantial improvements in performance, showing potential for expanded studies with larger models and more resources. Furthermore, the results obtained are in line with the expectations and are enough to spark a wider analysis.

Our study also underlined how many issues can arise from tuning a model for a specific purpose, highlighting the fact that LLM studies in the field of summarization are more of an art than a study. Training in this scope is different than standard Neural Networks training, as the labels and metrics are not fully in line with the desired behavior of the model. The following sections highlight the issues and potential improvements.

6.2 Limitations

One of the primary bottlenecks for this project was the limited availability of computational resources, specifically GPU cores and memory. Initially, we attempted to train on the entire dataset, which took approximately 5 hours per epoch with a batch size of 6, utilizing gradient accumulation. However, this approach proved to be unsustainable as the training process was often inconsistent and prone to divergence, likely due to factors such as model configuration, hyperparameter values or nuances in the dataset.

To address this, we adjusted our approach to training on a smaller subset of the data, around 10% of the full dataset, which allowed lower computational time and cost while reducing the risk of divergence. Even with this subset, we had to carefully balance the batch size per core to ensure non-noisy convergence as well as remaining within the bounds of our computational resources. On average, training for one epoch with these settings took around 25 minutes.

The limited access to resources of our university's HPC cluster and the restricted availability of team members further constrained our training process, making it challenging to conduct experiments over longer periods. Table 4 provides an overview of the memory usage and time spent when training on the subset of the data.

[1] https://github.com/03chrisk/PEFT-T5-on-CNN-dailynews

Technique	**Time** (in minutes)	**CPU usage** (GB)	**GPU usage** (GB)
Full tuning	26.4	2.2	26.8
LoRA	24.2	3.3	21.9

Table 4: Requirements for training on 10% of the dataset

Furthermore, we could not overcome an issue with our validation set forcing the summaries to be truncated after a low max-generated-token threshold, making it hard to independently evaluate intermediate metric values observed during training.

6.3 Future Research Directions

Based on the identified gaps, future research can address several areas:

Development of Improved Evaluation Metrics

One of the existing challenges is the issue of model evaluation. Commonly used metrics such as ROUGE and BLEU, while standard in summarization tasks, fail to capture deep semantic relationships and do not align well with human judgments (Reiter, 2018). Although more advanced metrics such as BERTScore (T. Zhang et al., 2019) and SummaC (Laban et al., 2022) have emerged, these methods require considerable computational resources, and their effectiveness in practical settings is still subject to debate (Nguyen et al., 2024).

The current reliance on automatic metrics like ROUGE and BLEU creates the need for development of more robust, human-aligned evaluation techniques. Future studies should explore the use of LLM-based evaluation methods that incorporate deeper semantic understanding. For instance, metrics like FineSurE, which focus on fine-grained summarization, offer a promising direction (Song et al., 2024). A hybrid model that combines automated evaluation with human-in-the-loop techniques could also offer a solution to this problem.

Enhancing Prompting Techniques

While prompting techniques offer a computationally efficient alternative to full fine-tuning, current methods tend to lack coherence and logical consistency in complex tasks. Future research should focus on improving these methods, perhaps by expanding on techniques like prompt chaining (S. Sun et al., 2024), which guides models through iterative stages of summarization. Exploration on how multi-modal data, such as images and texts, can be integrated into prompting techniques could also enhance summarization.

Parameter-Efficient Fine-Tuning

Methods such as LoRA (Jeong, 2024) and IA^3 (Xu et al., 2023) have demonstrated the potential to reduce computational costs while maintaining strong performance. Further research should aim to optimize these methods for broader applications. For instance, ensuring that PEFT can be extended to larger models such as GPT-4 and PaLM (Chowdhery et al., 2022), could result in access to high-performance LLMs in resource-constrained environments.

Domain-Specific Summarization

Another path for future research is the development of domain-specific summarization models. Current LLMs, while effective across general domains, struggle with highly specialized texts such as legal, medical, or technical documentation. Further investigation is needed into task-specific fine-tuning techniques and datasets tailored for these specialized areas, to ensure that models can effectively navigate domain-specific terminologies and structures.

Mitigating Model Hallucinations

One persistent issue in abstractive summarization is the occurrence of hallucinations - where models generate factually inaccurate information. Recent advances in instruction tuning have helped mitigate this problem, as seen in models like FLAN-T5 (Chung et al., 2022), but further research is necessary. Integrating fact-verification processes or external knowledge bases during the generation process could enhance the factual accuracy of the output.

General Improvements

As mentioned above, there were multiple issues with the model development process, such as the computational bottleneck or validation problems. Those and other issues should be addressed in a future research to check for the true improvement over the base model.

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chiang, C.-H., & Lee, H.-Y. (2023). Can large language models be an alternative to human evaluations? *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 15607–15631.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., ... Fiedel, N. (2022). Palm: Scaling language modeling with pathways. https://arxiv.org/abs/2204.02311

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., ... Wei, J. (2022). Scaling instruction-finetuned language models. https://arxiv.org/abs/2210.11416

Cnn/daily mail benchmark [*Papers with Code*]. (2024). https://paperswithcode.com/sota/abstractive-text-summarization-on-cnn-daily

Google cloud tutorial. (2024). https://cloud.google.com/translate/docs/advanced/automl-evaluate

Jeong, C. (2024). Fine-tuning and utilization methods of domain-specific llms. *arXiv preprint arXiv:2401.02981*.

Laban, P., Zeng, W., Holtzman, A., & Hovy, E. (2022). Summac: Factual summarization evaluation via semantic consistency. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. https://arxiv.org/abs/2110.08550

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lingo, R., Arroyo, M., & Chhajer, R. (2024). Enhancing llm problem solving with reap: Reflection, explicit problem deconstruction, and advanced prompting. *arXiv preprint arXiv:2409.09415*.

Nguyen, H., Chen, H., Pobbathi, L., & Ding, J. (2024). A comparative study of quality evaluation methods for text summarization. https://arxiv.org/abs/2407.00747

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2023). Exploring the limits of transfer learning with a unified text-to-text transformer. https://arxiv.org/abs/1910.10683

Rehman, T., Das, S., Sanyal, D. K., & Chattopadhyay, S. (2022). An analysis of abstractive text summarization using pre-trained models. In *Proceedings of international conference on computational intelligence, data science and cloud computing* (pp. 253–264). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1657-1_21

Reiter, E. (2018). A structured review of the validity of bleu. *Computational Linguistics*, *44*(3), 393–401.

Retkowski, F. (2023). The current state of summarization. https://arxiv.org/abs/2305.04853

Shakil, H., Mahi, A. M., Nguyen, P., Ortiz, Z., & Mardini, M. T. (2024). Evaluating text summaries generated by large language models using openai's gpt. https://arxiv.org/abs/2405.04053

Song, H., Su, H., Shalyminov, I., Cai, J., & Mansour, S. (2024). Finesure: Fine-grained summarization evaluation using llms. https://arxiv.org/abs/2407.00908

Sun, S., Yuan, R., Cao, Z., Li, W., & Liu, P. (2024). Prompt chaining or stepwise prompt? refinement in text summarization. *arXiv preprint arXiv:2406.00507*.

Sun, T., He, J., Qiu, X., & Huang, X. (2022). Bertscore is unfair: On social bias in language model-based metrics for text generation. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 3726–3739.

Wu, N., Gong, M., Shou, L., Liang, S., & Jiang, D. (2023). Large language models are diverse role-players for summarization evaluation. *arXiv preprint arXiv:2303.15078*.

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L. (2023). Parameter-efficient fine-tuning

methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *International Conference on Machine Learning*, 11328–11339.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1904.09675

A Original Article and Reference Summary

A.1 Original Article

(CNN) He's a blue chip college basketball recruit. She's a high school freshman with Down syndrome. At first glance Trey Moses and Ellie Meredith couldn't be more different. But all that changed Thursday when Trey asked Ellie to be his prom date. Trey – a star on Eastern High School's basketball team in Louisville, Kentucky, who's headed to play college ball next year at Ball State – was originally going to take his girlfriend to Eastern's prom. So why is he taking Ellie instead? "She's great... she listens and she's easy to talk to" he said. Trey made the prom-posal (yes, that's what they are calling invites to prom these days) in the gym during Ellie's P.E. class. Trina Helson, a teacher at Eastern, alerted the school's newspaper staff to the prom-posal and posted photos of Trey and Ellie on Twitter that have gone viral. She wasn't surprised by Trey's actions. "That's the kind of person Trey is," she said. To help make sure she said yes, Trey entered the gym armed with flowers and a poster that read "Let's Party Like it's 1989," a reference to the latest album by Taylor Swift, Ellie's favorite singer. Trey also got the OK from Ellie's parents the night before via text. They were thrilled. "You just feel numb to those moments raising a special needs child," said Darla Meredith, Ellie's mom. "You first feel the need to protect and then to overprotect." Darla Meredith said Ellie has struggled with friendships since elementary school, but a special program at Eastern called Best Buddies had made things easier for her. She said Best Buddies cultivates friendships between students with and without developmental disabilities and prevents students like Ellie from feeling isolated and left out of social functions. "I guess around middle school is when kids started to care about what others thought," she said, but "this school, this year has been a relief." Trey's future coach at Ball State, James Whitford, said he felt great about the prom-posal, noting that Trey, whom he's known for a long time, often works with other kids. Trey's mother, Shelly Moses, was also proud of her son. "It's exciting to bring awareness to a good cause," she said. "Trey has worked pretty hard, and he's a good son." Both Trey and Ellie have a lot of planning to do. Trey is looking to take up special education as a college major, in addition to playing basketball in the fall. As for Ellie, she can't stop thinking about prom. "Ellie can't wait to go dress shopping," her mother said. "Because I've only told about a million people!" Ellie interjected.

A.2 Reference Summary (Highlights)

College-bound basketball star asks girl with Down syndrome to high school prom. Pictures of the two during the "prom-posal" have gone viral.

B Generated Summaries

B.1 T5 base

He was originally going to take his girlfriend to Eastern's prom. "She's great... she listens and she's easy to talk to" he said. "I'm so happy for him," Ellie's mom said. "I'm so happy for him," Trey's coach said

B.2 T5 base + Prompring

"That's the kind of person Trey is," says a teacher at eastern high school. "That's the kind of person Trey is," says a teacher. "It's exciting to bring awareness to a good cause," says a mother.

B.3 T5 base + LoRA

High school basketball star Trey Moses asks Ellie Meredith to be his prom data. Ellie is a freshman with Down syndrome

B.4 T5 base + Full Fine-Tuning

Trey Moses, a college basketball recruit, asked Ellie Meredith to be his prom date. Ellie is a high school freshman with Down syndrome. Trey's mom says he's a good son, but she says she's numb to the moment.