



Reducing the Resources Required for Domain Adaptation of Pre-Trained RAG Models: A Parameter-Efficient Approach

by

Matt Mallen

This thesis has been submitted in partial fulfillment for the
degree of Master of Science in Artificial Intelligence

in the
Faculty of Engineering and Science
Department of Computer Science

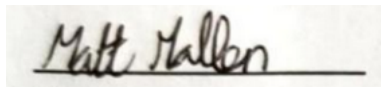
May 2025

Declaration of Authorship

This report, Reducing the Resources Required for Domain Adaptation of Pre-Trained RAG Models: A Parameter-Efficient Approach, is submitted in partial fulfillment of the requirements of Master of Science in Artificial Intelligence at Munster Technological University Cork. I, Matt Mallen, declare that this thesis titled, Reducing the Resources Required for Domain Adaptation of Pre-Trained RAG Models: A Parameter-Efficient Approach and the work represents substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged. I confirm that:

- This work was done wholly or mainly while in candidature Master of Science in Artificial Intelligence at Munster Technological University Cork.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University Cork or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

A handwritten signature in black ink, appearing to read 'Matt Mallen', is written over a horizontal line.

Date: 12 May, 2025

MUNSTER TECHNOLOGICAL UNIVERSITY CORK

Abstract

Faculty of Engineering and Science

Department of Computer Science

Master of Science

by Matt Mallen

This thesis addresses a critical challenge in adapting Retrieval Augmented Generation (RAG) models: the prohibitive computational resources typically required for domain-specific fine-tuning. While RAG models effectively mitigate hallucinations in large language models by grounding responses in external knowledge, adapting them to new domains traditionally demands extensive resources—re-encoding large knowledge bases and updating billions of parameters. We introduce a parameter-efficient approach to RAG domain adaptation, systematically evaluating multiple Parameter-Efficient Fine-Tuning (PEFT) techniques across three diverse domains: biomedical literature, news articles, and conversational data. Our comprehensive experiments demonstrate that Low-Rank Adaptation (LoRA) consistently outperforms prompt-based methods, achieving significant improvements in the exact match ($1.75\text{--}3.06\times$) and F1 scores ($1.69\text{--}3.02\times$) while modifying only 0.12% of model parameters. Additionally, we explore a novel document adaptation technique to eliminate knowledge base re-encoding requirements. Our framework substantially reduces computational overhead, enabling effective RAG adaptation on a single 6GB GPU, whereas previous approaches required multiple high-memory GPUs. This research advances efficient domain adaptation, making RAG technology more accessible for resource-constrained environments and enabling broader application across specialised knowledge domains without prohibitive infrastructure investments.

Acknowledgements

I would like to thank my supervisor, Dr Hamdan Awan, for his supervision of this project. I am also grateful for friends, family, and loved ones for their support throughout my studies.

Contents

Declaration of Authorship	i
Abstract	ii
Acknowledgements	iii
List of Figures	viii
List of Tables	ix
Abbreviations	x
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Structure of Thesis	4
2 Literature Review	5
2.1 Retrieval Augmented Generation (RAG)	5
2.1.1 Background and Motivation	6
2.1.2 Clarifying Overloaded Terms	6
2.1.3 Understanding RAG	7
2.1.3.1 Basic Concept	7
2.1.3.2 Architecture	7
2.1.4 Theoretical Framework	8
2.1.4.1 Retrieval Component	8
2.1.4.2 Generation with Retrieved Context	8
2.1.5 Limitations of Retrieval Enhanced Research Generally	9
2.1.6 Advancements Beyond Original RAG	11
2.1.6.1 Pre-Retrieval Optimisations	11
2.1.6.2 Post-Retrieval Optimisations	13
2.1.6.3 Adapters	13
2.1.6.4 End-to-End vs. Pipeline Training	14
2.1.7 Challenges of Domain Adaptation	14
2.1.8 Comparative Overview of RAG Adaptation Approaches	15

2.1.9	Potential Solution	15
2.2	Parameter-Efficient Fine-Tuning (PEFT)	16
2.2.1	Adapter-based Methods	16
2.2.2	Prompt-Based Methods	17
2.2.3	Low-Rank Decomposition Methods	18
2.2.4	Comparative Analysis of PEFT Methods	19
2.2.4.1	Fine-tuning versus Shallow Prompting:	19
2.2.4.2	Deep versus Shallow Prompting:	20
2.2.4.3	Adaptation for Different Tasks:	20
2.2.4.4	Memory Efficiency:	21
2.2.5	Limitations of PEFT Research Generally	21
2.3	PEFT for Retrieval-Enhanced Models	23
2.3.1	Limitations of Current PEFT for RAG Research	23
2.3.2	Our Approach	25
2.3.3	RAG Adaptation Overview	26
2.3.4	Cross-Domain Comparison	26
3	Experimental Methodology	28
3.1	Focus on Prompt-Based and LoRA Methods	28
3.2	Experimental Framework	29
3.3	Phased Experimental Approach	30
3.4	Datasets and Evaluation Methodology	31
3.4.1	Dataset Selection and Processing	31
3.4.2	Evaluation Metrics	31
3.4.2.1	Evaluation Metrics Formal Definitions	32
3.4.2.2	Statistical Significance Testing	33
3.5	Implementation Challenges and Limitations	34
3.6	Computing Infrastructure	34
3.7	Reproducibility Considerations	35
4	Artefact Design and Implementation	36
4.1	Integration with RAG-end2end Framework	36
4.1.1	Fine-tuning Pipeline Modifications	37
4.2	Stratified Random Sampling for Resource-Constrained Validation	38
4.3	Integration of PEFT	39
4.4	Custom P-tuning v2 Implementation for DPR	41
4.5	Evaluation Artefact	41
4.5.1	PEFT Model Loading	42
4.5.2	Enhanced Retrieval Precision	42
4.5.3	Statistical Significance Testing	43
4.5.4	Integration with Original Framework	43
5	Novel Document Adapter	44
5.1	Motivation	44
5.2	Architectural Overview	44
5.2.1	Embedding Adapter Module	46
5.2.2	Enhanced Retrieval Process	46

5.2.2.1	Design Rationale	46
5.2.2.2	Implementation Details	47
5.3	Training Methodology	47
5.3.1	Periodic Index Updates	48
5.3.1.1	Design Rationale	48
5.3.1.2	Implementation Details	48
5.3.2	Autoencoding-based Retriever Training	49
5.4	Theoretical Foundation	49
5.4.1	Embedding Space Transformation	49
5.4.2	Gradient Flow Preservation	50
5.4.3	Index Staleness Management	50
5.5	Framework Integration	50
5.6	Limitations	51
5.7	Summary	51
6	Evaluation	52
6.1	Model Selection Experiments	52
6.1.1	RAG-token-base versus RAG-token-nq Initialisations	52
6.1.2	Shallow versus Deep Prompting Techniques	53
6.1.3	Assessment of LoRA adaptations	53
6.1.4	Document Adapter Architecture Experiments	54
6.2	Final Evaluation	55
6.2.1	Experimental Setup	55
6.2.2	Evaluation Metrics	55
6.2.3	Results and Analysis	56
6.2.3.1	COVID-19 QA Results	56
6.2.3.2	News QA Results	57
6.2.3.3	Conversation QA Results	57
6.2.4	Cross-Dataset Comparison	58
6.2.5	Parameter Efficiency Analysis	58
6.2.6	Comparative Analysis with Prior Work	59
7	Discussion and Conclusions	61
7.1	Discussion	61
7.1.1	Integration of PEFT Methods with RAG	62
7.1.2	Comparative Performance of PEFT Methods	62
7.1.3	Document Adaptation Without Re-encoding	63
7.1.4	Practical Implications	64
7.1.5	Limitations and Challenges	64
7.2	Conclusion	65
7.3	Future Work	66
7.3.1	Improved Document Adaptation Techniques	67
7.3.2	Technical Enhancements	67
	Bibliography	69

A Custom P-tuning v2 Implementation Details	75
A.1 Adaptation of P-tuning v2 for DPR Question Encoders	75
A.2 Key Components of the Implementation	76
A.3 Technical Challenges and Solutions	76
A.4 Divergence from Reference Implementation	77
B Model Selection Experiments	78
B.1 RAG-token-base versus RAG-token-nq Initialisations	78
B.2 Shallow versus Deep Prompting Techniques	79
B.3 Assessment of LoRA adaptations	80
B.4 Exploration of Increasing Adapter Complexity	81
B.5 Assessment of KL Divergence Loss	83

List of Figures

5.1	Overview of the novel document adapter approach.	45
B.1	Comparison: <code>rag-token-base</code> vs. <code>rag-token-nq</code> initialisation (Appendix)	78
B.2	Comparison: Deep vs. Shallow Prompt Tuning (Appendix)	79
B.3	Comparison: LoRA vs. Deep Prompt Tuning (Appendix)	81
B.4	Comparison of Adapter Architectures (Appendix)	82
B.5	Comparison: Standard Loss vs KL Divergence Loss (Appendix)	83

List of Tables

2.1	Methodological Analysis of Key RAG Adaptation Approaches	15
2.2	Quantitative Comparison of PEFT Methods	19
2.3	Theoretical and Practical Trade-offs in RAG Adaptation Methods	27
6.1	Baseline vs. LoRA performance comparison	56
6.2	Relative performance improvements across adaptation methods	59
6.3	Retrieval precision improvements across methods	60

Abbreviations

LLM	L arge L anguage M odel
PEFT	P arameter E fficient F ine- T uning
NLP	N atural L anguage P rocessing
RAG	R etrieval- A ugmented G eneration
KB	K nowledge B ase
LoRA	L ow- R ank A daptation
GPU	G raphics P rocessing U nit
REALM	R etrieval- A ugmented L anguage M odel pre-training
RETRO	R etrieval- E nhanced T ransfo R mer
FAISS	F acebook A I S imilarity S earch
BART	B idirectional and A uto- R egressive T ransformer
IR	I nformation R etrieval
BERT	B idirectional E ncoder R epresentations from T ransformers
KGP	K nowledge G raph P rompting
BM25	B est M atching 25
PRCA	P luggable R eward-Driven C ontextual A dapter
AAR	A ugmentation- A dapted R etriever
BGM	B ridging the G ap M odel
DPR	D ense P assage R etrieval
GPT	G enerative P re-trained T ransformer
T5	T ext-to- T ext T ransfer T ransformer
NLU	N atural L anguage U nderstanding
LSTM	L ong S hort- T erm M emory
MLP	M ultilayer P erceptron
QA	Q uestion A nswering

Chapter 1

Introduction

Large Language Models (LLMs) have recently gained significant traction for their ability to generate fluent, context-aware responses in a range of Natural Language Processing (NLP) tasks [1–3]. However, a persistent challenge facing these models is their tendency to produce hallucinations—statements that, while syntactically and semantically coherent, are factually incorrect [4]. This issue can undermine trust in automated systems where accuracy is critical, such as academic, medical, and legal applications [5]. To address this challenge, Retrieval-Augmented Generation (RAG) has emerged as a promising approach, integrating a retriever and a generative model, enabling the system to incorporate information from an external Knowledge Base (KB) during generation [6]. By doing so, RAG can provide provenance for each piece of information, significantly reducing hallucinations.

Despite their merits, RAG systems often require extensive computational resources when being adapted to some new task or domain for which they were not trained initially, including re-encoding large external KBs and fine-tuning massive numbers of model parameters [7]. Parameter-Efficient Fine-Tuning (PEFT) methods — such as prompt tuning, prefix tuning, and Low-Rank Adaptation (LoRA) [8–10] — have been proposed as more efficient alternatives to full fine-tuning, but they have yet to be rigorously explored by researchers in the context of RAG domain adaptation. This thesis investigates how PEFT methods can reduce the resource overhead of adapting RAG systems for domain-specific tasks without compromising performance.

1.1 Motivation

From my work as a software engineer, I have observed how tools like ChatGPT can often perform admirably in complex generative tasks (boilerplate code, fixing bugs, et

cetera.) but frequently lack the necessary grounding in verified facts (e.g. when carrying out tasks that require familiarity with business logic or some niche software library). As an industry anecdote, a recent Wall Street Journal piece highlights the costs of bridging specialised domain knowledge in LLMs [11]. However, academic investigations underline the formal computational barriers of re-encoding large corpora [7, 12]. I chose to pursue this research after observing the immense potential of RAG models. RAG has demonstrated performance gains in various knowledge-intensive tasks. For instance, BioinspiredLLM, a RAG-based conversational model fine-tuned on biological materials science literature, performs superior knowledge recall and hypothesis generation [13]. RAG has also been used to improve lay language generation in the biomedical domain, enhancing the accessibility of such information [14]. Furthermore, RAG has been used for clinical guideline interpretation, demonstrating applicability in clinical domains [15].

However, deploying and continually updating a RAG model can be expensive and time-consuming. Large commercial organisations have the resources to maintain clusters of high-performance Graphics Processing Units (GPUs) to fine-tune every aspect of these massive models. Even in the research implementation of Siriwardhana et al. [7], they required six Tesla V100 GPUs with 32GB of memory each to adapt a pre-trained model to new domains, with four dedicated to training and two for re-encoding. For smaller organisations or research teams that lack access to such computational infrastructure, the requirements for frequent re-encoding of the KB and training millions of parameters can be prohibitive.

PEFT methods present an appealing solution. Rather than adjusting the billions of parameters in a language model or re-training entire retrieval components, PEFT methods optimise only a small subset of parameters that guide the frozen model to generate task-specific outputs while keeping most of the model frozen; this significantly reduces the computational burden and makes it feasible to adapt RAG systems more economically [16]. It also holds the promise of flexible domain adaptation: an organisation may have multiple niche domains to cover, and training small sets of PEFT parameters for each could be much more straightforward than extensively re-training or re-indexing for every domain shift. This approach has been demonstrated in practice by organisations like the PGA Tour, which uses RAG to handle specialised golf knowledge by incorporating their 190-page rulebook and other domain-specific content with Claude [11].

The key motivation behind this project is the prospect of making RAG models more accessible and easier to customise. If PEFT proves effective, it could drastically lower the barrier to entry for smaller players, enabling them to harness advanced retrieval-augmented text generation without substantial hardware investments. Therefore, I aim

to address a key technical challenge and contribute to more equitable access to state-of-the-art NLP technology.

1.2 Contributions

Problem Statement: Despite the growing success of RAG systems for mitigating hallucinations in LLMs, adapting these models to new domains remains prohibitively expensive. Large KBs often must be re-encoded, while billions of model parameters typically require fine-tuning. This high resource demand blocks small organisations or research labs from fully leveraging RAG for domain-specific tasks, and it also complicates maintenance in real-world settings where KBs must be updated frequently.

Goals and Research Questions:

- *Primary Goal:* Reduce the computational overhead required to adapt RAG systems to new domains.
- *Research Questions:*
 1. How can PEFT methods be effectively integrated into retrieval-generation architectures like RAG?
 2. Which existing PEFT methods realise the best performance gains (accuracy, F1-score, top-k retrieval)?
 3. How can PEFT methods be applied to eliminate the need for KB re-encoding when adapting RAG retrieval end to end?

Main Contributions:

1. **Systematic evaluation of parameter-efficient methods for RAG adaptation.** We empirically compare multiple PEFT approaches (including LoRA and prompt-based methods) for the RAG architecture, identifying optimal strategies for efficient adaptation with minimal parameter updates. Our results demonstrate that parameter-efficient techniques can be effectively extended to more complex, hybrid retrieval-generation architectures, previously explored primarily for single-component language models.
2. **Novel exploration of techniques for avoiding re-encoding when adapting RAG models.** We investigate an approach that keeps the passage encoder frozen

while training a small adapter network to leverage the existing encoded context. This exploratory technique eliminates the need to re-encode the external KB when adapting retrieval end to end, which could potentially reduce training time and hardware requirements. Though optimisation challenges emerged in our evaluation, this exploration provides valuable insights into the complexities of efficient document adaptation and identifies promising directions for future research.

Advancing the State of the Art: We provide a new, lightweight paradigm for domain adaptation in RAG. Our proposed parameter-efficient approach lowers the resource barrier to deployment while delivering competitive performance in knowledge-intensive tasks; this opens the door for broader usage of RAG, where frequent updates to knowledge or multiple niche tasks would otherwise make full fine-tuning infeasible.

1.3 Structure of Thesis

This report has seven chapters:

- **Introduction:** The current chapter establishes the motivation for the work, outlines the main contributions and previews the report structure.
- **Literature Review:** This chapter surveys existing research on RAG systems, PEFT methods, and the sole study (to our knowledge) combining the two.
- **Experimental Methodology:** The methodology chapter details how the research is set up, including the experimental framework, data collection, evaluation protocols, and computing infrastructure.
- **Artefact Design and Implementation:** This chapter describes the design rationale and implementation details for integrating PEFT techniques into the RAG architecture.
- **Novel Document Adapter:** This chapter presents our technique for freezing the passage encoder while enabling end-to-end retrieval adaptation.
- **Evaluation:** The chapter details our experimental results, including preliminary experiments and final evaluations.
- **Discussion and Conclusions:** This chapter interprets the key findings of the study and its limitations, summarises the contributions, and discusses potential directions for expanding this research.

Chapter 2

Literature Review

This literature review examines the intersection of two complementary research directions: techniques for grounding language model outputs in external knowledge (RAG) and methods for efficiently adapting large models to new domains (PEFT). These approaches address different aspects of a common challenge—how to create language models that are factually reliable and adaptable to specialised domains without prohibitive computational costs.

The theoretical connection between these approaches lies in their representation of and access to knowledge. RAG systems externalise knowledge storage to retrieve relevant information on demand. PEFT methods optimise how models internally process and generate from available information. Combining these approaches offers a powerful paradigm for efficient, domain-specific language models that maintain factual grounding while minimising computational overhead.

The following sections analyse these approaches independently before examining their integration, highlighting theoretical and practical considerations for creating efficient, adaptable, and factually grounded language models.

2.1 Retrieval Augmented Generation (RAG)

This section outlines and analyses the extant research landscape concerning RAG.

2.1.1 Background and Motivation

Recent advances in **LLMs** have transformed many areas of **NLP**, from conversational agents and text summarisation to question answering (QA) and code generation [1–3]. However, as noted in the introduction (Chapter 1), LLMs remain vulnerable to generating *hallucinations*: they sometimes provide coherent but factually incorrect or unsubstantiated information [4]. These incorrect outputs pose risks for real-world usage in domains such as healthcare where factual reliability is paramount [5]. RAG has emerged as a promising technique to alleviate hallucinations by integrating a *retriever* (which fetches text passages from an external KB) with a *generator* [6].

2.1.2 Clarifying Overloaded Terms

The term “RAG” has become overloaded in the literature, with some researchers using it to refer to any model enhanced with retrieval capabilities. This ambiguity necessitates precise terminology, particularly when discussing architectural differences between retrieval-enhanced approaches.

In this thesis, RAG refers to the architectural pattern introduced by Lewis et al. [6]. This approach builds upon earlier retrieval-augmented methods like the Retrieval-Augmented Language Model (REALM) by extending retrieval capabilities to sequence-to-sequence models [17]. The key components of RAG include:

- A dense retriever that identifies relevant documents based on the input
- A generator that produces output conditioned on both the input and retrieved information
- A marginalisation mechanism that either treats all outputs as dependent on the same document (RAG-Sequence) or allows different tokens to depend on different documents (RAG-Token)

The above differs fundamentally from RETRO (Retrieval-Enhanced Transformer) [18], which integrates retrieval directly into the transformer architecture through chunked cross-attention mechanisms. RETRO divides its input into chunks, and after each chunk is processed, the model retrieves relevant documents for the next chunk, with specialised attention mechanisms to incorporate this information.

While both approaches enhance language models with retrieval capabilities, their implementations have significant implications:

- RAG works with existing encoder-decoder architectures with minimal architectural modifications
- RETRO requires specific modifications to the transformer’s attention mechanisms
- RAG typically retrieves once at the beginning of generation, while RETRO retrieves at chunk boundaries throughout the text

This distinction is crucial because it affects how researchers and developers deploy, fine-tune, and optimise these models. When discussing retrieval-enhanced models more broadly, “retrieval-enhanced models” serve as a more accurate umbrella for RAG-style frameworks and architecturally integrated approaches like RETRO.

2.1.3 Understanding RAG

In this subsection, we explain how RAG works at a high level.

2.1.3.1 Basic Concept

The central idea of RAG is simple: instead of relying solely on a model’s internal, parameterised knowledge, the system retrieves relevant external documents, appends them to the input prompt, and conditions the generator to produce answers rooted in that supporting evidence. This process enhances factual grounding and ensures provenance for claims [6]. As the user can trace outputs back to sources, confidence in the reliability of the generation increases.

2.1.3.2 Architecture

The typical RAG architecture consists of three primary components:

- **KB:** An external repository containing reference documents, typically processed into chunks of manageable size (e.g., 100 words) and indexed for efficient retrieval using a vector store such as Facebook AI Similarity Search (FAISS) [19].
- **Retrieval Module:** Responsible for identifying and retrieving relevant information from the KB given an input query; this employs dense vector retrieval techniques, where the module encodes questions and documents into a shared embedding space [12].

- **Generation Module:** A sequence-to-sequence language model - e.g. Bidirectional and Auto-Regressive Transformer (BART) - that generates responses based on the original input and the retrieved context [20].

2.1.4 Theoretical Framework

RAG builds on modern neural approaches to information retrieval (IR) while sharing conceptual foundations with classical IR principles. The model combines dense neural retrieval with sequence-to-sequence generation in a probabilistic framework.

2.1.4.1 Retrieval Component

The retrieval mechanism in RAG employs dense representations to map queries and documents into a shared vector space. While this approach shares similarities with classical vector space models [21], it leverages modern neural encoders to compute these representations. Given a query Q and a KB \mathcal{K} , the retriever computes a probability distribution over documents:

$$p_{\eta}(z|x) \propto \exp(d(z)^{\top} q(x))$$

where $d(z)$ and $q(x)$ are dense embeddings of documents and queries respectively, computed using separate Bidirectional Encoder Representations from Transformers (BERT) encoders [6].

2.1.4.2 Generation with Retrieved Context

RAG treats the retrieved documents as latent variables in the generation process. The model provides two distinct formulations for combining retrieval with generation:

RAG-Sequence treats the same retrieved document as responsible For the entire output sequence:

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) \prod_i^N p_{\theta}(y_i|x, z, y_{1:i-1})$$

RAG-Token allows different documents to influence different output tokens:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z, y_{1:i-1})$$

where p_θ represents the generator that attends over both the input x and retrieved document z [6].

This retrieval-generation pipeline provides several advantages over pure parametric approaches:

- One can update the external KB without retraining
- One can trace generated content to source documents
- The model can access information beyond its parameter capacity
- Retrieved context helps ground generation in factual evidence

The separation of retrieval and generation, combined with the probabilistic treatment of retrieved documents as latent variables, enables RAG to combine parametric and non-parametric knowledge effectively while maintaining transparency and updateability.

2.1.5 Limitations of Retrieval Enhanced Research Generally

While REALM, RAG, and RETRO all leverage external knowledge retrieval, they differ fundamentally in their architectural designs and training paradigms, each with distinct trade-offs.

REALM’s approach to masked language pre-training demonstrates significant performance improvements over contemporary retrieval methods on Open-QA tasks. However, the paper primarily focuses on overall end-task performance metrics rather than isolating specific factors. While the authors highlight the interpretability benefits of retrieval-enhanced approaches, they do not specifically analyse whether the pre-training alignment between retrieval and generation independently enhances factual correctness beyond the general performance gains. Instead, REALM emphasises how its masked language model pre-training creates effective representations for downstream question-answering tasks while offering additional qualitative benefits such as interpretability and modularity [17].

RAG’s approach to retrieval-augmented generation represents an important step forward, though it contains several theoretical and practical limitations worth considering. The formulation $(p_\eta(z|x) \propto \exp(d(z)^\top q(x)))$ employs dot-product similarity between query and document embeddings, which may not capture all complex semantic relationships. While effective in practice, as demonstrated by state-of-the-art results across multiple tasks, future work might explore more sophisticated similarity metrics that better model non-linear relevance patterns.

The paper presents two approaches to document handling that reveal theoretical trade-offs: RAG-Sequence, which conditions a single document for generating the entire output, and RAG-Token, which can leverage different documents for different generation parts [6]. This distinction creates fundamental assumptions about knowledge utilisation—RAG-Sequence enforces document-level consistency but may miss complementary information across documents, while RAG-Token offers flexibility but risks generating inconsistent outputs by mixing information sources. These theoretical distinctions have practical implications for factuality versus comprehensiveness. Figure 2 of their paper shows that the model can dynamically shift attention between documents during generation, suggesting a more nuanced approach than simple independence [6].

Regarding retrieval depth, the authors explicitly investigate how performance varies with different numbers of retrieved documents (Figure 3 of their paper), finding that “retrieving more documents at test time monotonically improves Open-domain QA results for RAG-Sequence.” This empirical approach acknowledges the importance of retrieval depth [6], even if it does not establish a theoretical framework for dynamically determining optimal k values per query.

Separating the retrieval and generation components introduces an implicit information bottleneck at the retrieval stage. While this modular design enables straightforward KB updates, it also creates a theoretical ceiling on performance—the generator can never exceed the quality of the information the retriever provides, regardless of how sophisticated its parameterisation becomes. This bottleneck is particularly significant when considering domain adaptation, where retrieval quality may degrade substantially in unfamiliar domains.

The computational costs of RAG during inference deserve scrutiny. The model must perform dense embedding computation and similarity search for each query, adding non-negligible latency compared to pure parametric approaches [22]. This latency-accuracy trade-off is often underexplored in the existing literature, with few works providing comprehensive benchmarks across diverse retrieval corpus sizes and query complexities.

However, these criticisms must balance against retrieval-enhanced models’ significant practical advantages. Unlike pure parametric approaches, RAG systems provide explicit provenance for generated content, addressing a fundamental accountability concern in deployed AI systems. Furthermore, the modular nature of RAG architectures facilitates targeted improvements—researchers can independently enhance retrieval precision or generation quality without complete system retraining. One may argue that the computational overhead of retrieval is justified by improved factuality and interpretability and that while pure parametric models might achieve comparable performance on benchmark

tasks, RAG offers superior trustworthiness for high-stakes applications where verifiability outweighs latency concerns. Moreover, as vector database technology advances (e.g., FAISS optimisations [19]), the retrieval latency gap narrows, potentially rendering this critique less significant.

Lewis et al. [6] provides a limited analysis of RAG’s ability to handle ambiguous queries or reconcile contradictory information from different retrieved passages, an area requiring further investigation. Despite this, RAG’s empirical effectiveness across diverse knowledge-intensive tasks demonstrates the practical value of combining parametric and non-parametric approaches to knowledge access, even with these theoretical constraints and implementation challenges.

In contrast to REALM and RAG, RETRO’s chunked cross-attention mechanism allows for a more granular integration of retrieved content [18], potentially enabling a more nuanced use of external knowledge. However, this comes at the cost of a more complex architecture.

None of these works thoroughly investigate the robustness of their approaches when faced with adversarial inputs or domain shifts—a critical consideration for real-world deployment [17, 18, 23].

2.1.6 Advancements Beyond Original RAG

Having outlined the core RAG theory and architecture, we now examine key innovations optimising retriever and generator components, critically assessing their relative strengths and limitations. While the original RAG architecture [6] demonstrated a straightforward way to integrate external retrieval with a sequence-to-sequence model (e.g., BART), subsequent work has aimed to push beyond this naive formulation. These attempts focus on mitigating computational bottlenecks during adaptation, improving the consistency and quality of the retrieved passages, and unifying or restructuring pipeline steps. However, many of these approaches introduce different trade-offs that are often underexplored in the original papers.

2.1.6.1 Pre-Retrieval Optimisations

One notable strand of research addresses how to make retrieval more effective *before* the generator consumes it:

- **Query Reformulation:** Instead of using the user-provided query directly, the model refines or expands it (possibly employing a learned rewriting module or

chain-of-thought prompt) to better expose relevant keywords [24]. This approach mitigates issues with ambiguous or underspecified queries and often improves retrieval recall. However, it introduces additional computation and requires a trade-off between generalisation and specialisation. When using a trainable rewriter, the approach shows some dependency on the quality of pseudo-data used for training, and in some cases (as shown in the PopQA dataset results), the trainable rewriter does not match the performance of the LLM rewriter - suggesting challenges in distilling the more complex query rewriting capabilities from larger models to smaller ones.

- **Better Index Structures:** Knowledge Graph Prompting (KGP) has emerged as a practical approach for multi-document question answering by organising document content in graph structures [25]. This method constructs knowledge graphs with passages as nodes and their semantic/lexical relationships as edges, facilitating traversal between related content across documents. While KGP demonstrates improved performance over traditional retrieval methods, the quality of results depends on the graph construction approach used. The authors propose several complex construction methods, from simple TF-IDF keyword matching to more sophisticated approaches like TAGME entity linking. Their experiments show KGP can achieve higher accuracy than embedding-based methods while maintaining comparable efficiency, though performance and latency trade-offs exist as graph density increases. This approach is particularly valuable for questions requiring reasoning across multiple documents or understanding document structures such as pages and tables.
- **Blended Retrieval:** The “Blended RAG” approach combines different retrieval strategies (keyword-based similarity search, dense vector-based, and sparse encoder-based) with hybrid query formulations to improve retrieval accuracy [26]. Sawarkar et al. [26] demonstrate that this approach achieves superior performance across multiple benchmark datasets (NQ, TREC-COVID, SQuAD), even without domain-specific fine-tuning. Their experiments show that sparse encoder-based semantic search combined with “Best Fields” queries often yields the best results. The approach does involve computational trade-offs (with dense vector indices requiring more storage but faster indexing, while sparse vector indices offer more efficient querying despite slower indexing), and it has limited effectiveness for datasets lacking metadata. However, the authors demonstrate that their method can set new benchmarks for retriever accuracy and overall RAG performance, allowing for better results even with relatively minor LLMs.

2.1.6.2 Post-Retrieval Optimisations

After initial retrieval, more advanced methods can *re-rank* or *re-compress* the fetched passages, though with varying computational implications:

- **Learned Re-rankers:** A small ranker model can improve retrieval performance through post hoc re-ranking, i.e., reordering the top- k retrieved passages to place the most relevant and credible ones at the front [27]. While effective, this approach increases both training complexity and inference latency. Moreover, while Ram et al. [27] provides evidence that a GPT-2 110M model can achieve similar re-ranking performance to larger models, suggesting computational efficiency benefits, they do not include detailed analysis of inference latency or throughput trade-offs. Such benchmarks could further clarify the practical deployment considerations of their approach, particularly for resource-constrained environments.
- **Context Filtering:** Filtering irrelevant content from retrieved passages before generation improves performance across diverse tasks [28]. Different filtering strategies work optimally for different tasks: string inclusion for extractive QA, lexical overlap for dialogue, and information-theoretic approaches for complex reasoning tasks. While effective, this approach requires training separate filtering models, which adds complexity to the pipeline. The paper’s evaluation is limited to Wikipedia-based datasets, and performance in specialised domains remains unexplored. Nevertheless, filtering reduces input length by 44-64%, improving accuracy and efficiency by removing distractions from relevant passages and minimising harmful content from irrelevant ones.

2.1.6.3 Adapters

An emerging class of RAG enhancements employs various “adapters” - external or intermediate trainable components that aim to align the retriever and generator for specific domains while keeping the core model weights frozen. Notable examples include the Pluggable Reward-Driven Contextual Adapter (PRCA) [29], the Augmentation-Adapted Retriever (AAR) [30], and the Bridging the Gap (BGM) model [31]. PRCA uses a reward-driven contextual adapter to refine the retriever output before passing it to the generator, requiring reinforcement learning to achieve its goals - a fundamentally different training paradigm to the base model [29]. AAR requires another LLM during training to learn documents “preferred” by LLMs [30]. BGM introduces a seq2seq model that transforms raw retrieved content to align more effectively with the generator’s expectations, adding a distinct model to the hybrid architecture [31]. While such methods

have shown promise in improving domain adaptation and retrieval accuracy, they often come at the cost of additional architectural complexity and inference overhead.

2.1.6.4 End-to-End vs. Pipeline Training

While initially, RAG trained in two stages - retriever pre-training - e.g., Dense Passage Retrieval (DPR) - and generator fine-tuning - some advanced approaches unify these into a single end-to-end learning process [7]. This technique back-propagates through retrieval and generation, ostensibly enabling the generator to communicate its mistakes (e.g., retrieving the wrong passages) to the retriever, and improves many performance metrics compared to keeping the retriever frozen. However, end-to-end RAG can be *extremely* computationally intensive because each training step might involve searching huge external indexes, re-encoding and re-ranking passages [7, 23]; this is particularly daunting for smaller organisations with limited GPU clusters. Indeed, Siriwardhana et al. [7] used multiple high-memory GPUs specifically to handle the re-encoding step alone.

2.1.7 Challenges of Domain Adaptation

An essential topic in RAG research is *domain adaptation*, i.e., how to adapt a pre-trained RAG model to new tasks or knowledge domains. The original RAG work used Wikipedia as the KB [6], but practical use cases often require retrieving from specialised corpora (e.g., biomedical literature, technical documentation, or financial statements).

- **Retriever–Generator Misalignment:** If the retriever was initially trained on general-purpose text (e.g., Wikipedia) and the generator is pre-trained on a broad distribution, there is no guarantee that it remains well-aligned after partial updates on domain-specific data. For instance, if one only tunes the generator, the retriever may fetch the “wrong” passages; if one only tunes the retriever, the generator might ignore subtle domain cues [7].
- **Costs of Re-indexing:** Re-embedding all passages with a newly updated passage encoder can be extremely expensive for a large external corpus. One typical result is that practitioners freeze the passage encoder to avoid re-indexing overhead - thus, domain adaptation can become suboptimal if the passage embeddings never see domain data [7].
- **Fine-tuning Cost:** The typical approach of fine-tuning a pre-trained RAG for each new domain or task is both resource-intensive and time-consuming due to the

large number of parameters involved, often in the order of millions or billions [7, 23]; this poses a significant barrier to the widespread adoption and customisation of RAG models.

- **Risk of Catastrophic Forgetting:** When researchers and developers fine-tune LLMs for narrower tasks, they may degrade performance in previously mastered areas. RAG can mitigate forgetting by externally retrieving knowledge, yet the generator’s distributional shift still poses a risk [32].

2.1.8 Comparative Overview of RAG Adaptation Approaches

Table 2.1: Methodological Analysis of Key RAG Adaptation Approaches

Approach	Methodological Strengths	Underlying Assumptions
Original RAG [6]	Strong probabilistic foundation; Integration with existing pre-trained models	Retrieved documents contain sufficient information; Retrieval bottleneck is acceptable
End-to-End RAG [7]	Eliminates retrieval-generation mismatch; Potentially higher task performance	Differentiable retrieval is tractable; GPU memory scaling is available
Query Reformulation [24]	Addresses query-document vocabulary mismatch; Reduces retrieval failures	Query problems represent primary retrieval bottleneck
Blended Retrieval [26]	Combines complementary retrieval signals; Mitigates vector space limitations	Dense and sparse retrievals have complementary strengths
Context Filtering [28]	Reduces irrelevant content; Potentially improves generation focus	Filtering heuristics generalize across domains

Note: This table presents representative approaches for RAG adaptation. Additional methods (e.g., adapter-based approaches, knowledge graph prompting, and learned re-rankers) are discussed in the text.

2.1.9 Potential Solution

Some research focuses on adversarial prompt attacks on RAG rather than using PEFT as a training method to improve RAG’s performance or efficiency [33]. Specifically, it looks at how malicious prompts can manipulate RAG’s retrieval results [33], which may support the viability of prompt-based PEFT for RAG since it demonstrates that RAG models are sensitive to prompt manipulation. Section 2.2 will dive deeper into these ideas of prompt learning and parameter-efficient adaptation more broadly.

2.2 Parameter-Efficient Fine-Tuning (PEFT)

While RAG presents challenges in adaptation, recent advances in PEFT offer promising directions for improvement. The remarkable success of large pre-trained LLMs such as T5, BERT, and Generative Pre-trained Transformer 3 (GPT-3) has spurred interest in how to adapt these massive models for downstream tasks without incurring massive computational and memory costs. Traditional full fine-tuning, where every parameter of the LLM is updated, is straightforward but can be prohibitive for large models - both in terms of the GPU memory needed for backpropagation and the disk storage required to save a fine-tuned copy of the model for each new task. Consequently, the NLP community has explored a variety of PEFT methods. These techniques adapt only a relatively small subset of the full pre-trained model parameters (or introduce additional small modules) while most model weights remain frozen. We can group PEFT methods into three categories: (1) **adapter-based methods**, (2) **prompt-based methods**, and (3) **low-rank decomposition methods**. While all three families ultimately aim to reduce the resource overhead of model adaptation, they differ in where they inject trainable parameters and how they modulate the pre-trained weights.

2.2.1 Adapter-based Methods

Adapter-based methods such as Houlsby-style adapters insert small “adapter” modules within each Transformer block [34]. Each adapter is typically a two-layer feedforward block with a low-dimensional bottleneck, making the adapter’s parameter count much smaller than the original layer’s parameter count. During adaptation, the adapter layers are trained on task-specific data while the rest of the pre-trained model is frozen. This approach yields robust performance gains with only a minor fraction (e.g., 1–3%) of the model’s parameters being updated, substantially reducing memory overhead in training [34]. After the success of Houlsby-style adapters for NLP, a plurality of adapter architectures—residual and cross-lingual adapters—have been proposed [35, 36]. Adapter-based training promises that one can maintain multiple domain-specific adapters for a single base model, enabling domain or task switching with minimal overhead [34]. When the user can fully specify a model’s task with a short textual prompt, there is an open question of whether adapters are overkill. If one can inject all the domain or task information at the model’s input and then use the frozen backbone, we can do away with multilayer adapters in favour of simpler schemes.

2.2.2 Prompt-Based Methods

Prompting refers to prepending task-specific text or tokens to input to steer the model toward a specific behaviour - most famously exemplified by GPT-3’s few-shot “in-context learning” [37]. However, text prompts (also called “hard prompts”) can be suboptimal: it is not obvious how to select discrete words to achieve optimal performance, and a single prompt might not suffice to convey complex domain knowledge.

Hence, the development of **prompt tuning**, in which the prompt is represented by learned continuous vectors (so-called “soft prompts”) that do not necessarily correspond to human-meaningful words; only these specially introduced prompt vectors update via gradient descent, while the entire pre-trained model stays frozen. During inference, the trained prompt vectors prepend to the token embeddings of an input sequence. This approach retains the advantage of not duplicating an entire copy of the model for each task. Moreover, once the model finishes training, a user can combine or swap out prompts for different tasks, domains, or user contexts with minimal overhead in model size. However, this sort of prompt tuning can be done at different “depths” in the transformer:

- **Shallow prompt tuning** (often just called “prompt tuning” or “soft prompting”) modifies only the input embeddings; this was introduced by Lester et al. [38] for Text-to-Text Transfer Transformer (T5), building on how GPT-3 uses textual prompts. Instead of providing plain text as a prompt, the approach learns “soft tokens”, i.e., new embeddings. The prompt encoder prepends the learned prompt to the front of an input sequence’s embeddings.
- **Prefix tuning** (or “deep prompting”) goes further. Here, a tunable prefix of vectors is prepended not only to the input layer but to every attention layer’s key/value states; Li and Liang [8] first popularised it, showing strong results on generative tasks by tuning GPT-2 and BART efficiently. Because these “prefix” vectors appear at every self-attention layer, the approach can more strongly steer the model’s hidden states than shallow prompt tuning. It also increases the number of trainable parameters compared to shallow Prompting but is still far less than full fine-tuning.
- **P-Tuning** was introduced by Liu et al. [39] primarily for knowledge probing and Natural Language Understanding (NLU) tasks, though it has also shown effectiveness for some generation tasks. It uses a Long Short-Term Memory (LSTM) or Multilayer Perceptron (MLP) - based prompt encoder to generate continuous prompts, allowing the learned prompts to capture dependencies between tokens.

While conceptually similar to soft prompting methods, its creators designed it to elicit knowledge from language models like BERT and GPT.

- **P-Tuning v2** [40] is not merely a variation in prompt generation but an optimised implementation of deep prompt tuning specifically for NLU tasks. It applies continuous prompts at every network layer (similar to prefix tuning) and demonstrates that properly optimised prompt tuning can match fine-tuning performance universally across model scales and tasks, even with only 0.1%-3% of the parameters. This work showed that the previously observed performance gap between prompt tuning and fine-tuning disappears with appropriate optimisation.

Notably, in RAG contexts - where a model retrieves external knowledge and then generates output - one can use prompt-based methods for both the retriever and the generator. For example, one might apply prefix tuning to the generative BART model and P-Tuning v2 to the BERT-based retriever, effectively using only a small number of additional parameters while the huge pre-trained components remain fixed. Despite the differences in naming conventions (prefix tuning, P-Tuning, P-Tuning v2), they share the underlying concept of pushing gradient updates into newly introduced or decomposed prompts, leaving the main model weights untouched.

2.2.3 Low-Rank Decomposition Methods

Another line of work is LoRA, introduced by Hu et al. [10]. In LoRA, each trained transformer weight matrix $W_0 \in \mathbb{R}^{d \times k}$ is left frozen, and its potential update ΔW is decomposed into a product of two smaller matrices: a $d \times r$ matrix B and an $r \times k$ matrix A , where r is a small number chosen to be much less than $\min(d, k)$. Formally, $\Delta W = BA$, where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$. The number of additional parameters is small since $r \ll \min(d, k)$. During training, the modified forward pass becomes $h = W_0x + \Delta Wx = W_0x + BAx$, typically scaled by a factor $\frac{\alpha}{r}$. At inference time, one can precompute the full effective weight matrix $W = W_0 + \Delta W$, introducing no additional latency compared to the original model.

Like prefix tuning, LoRA's memory usage is minimal at inference: in principle, the forward pass must multiply ΔW as well, but in practice, LoRA can partially or wholly fuse its overhead with the existing matrix multiplication. Critically, the number of new parameters in LoRA grows with the number of adapted linear layers. Typically, one might apply LoRA to just the query/key/value transforms in all or some fraction of the attention blocks.

A key advantage of LoRA is that it adds no extra sequence length or intermediate tokens. Soft Prompting typically adds tokens to the input for each inference (although the rest of the forward pass can shadow the overhead). On the other hand, prefix tuning modifies only the hidden states in self-attention, requiring no overhead in multiplications per se, but it must store new states for each token at each layer. In practice, LoRA, prefix tuning, and shallow prompt tuning are each used in contexts where their trade-offs align best with the memory, speed, and parameter overhead needed.

2.2.4 Comparative Analysis of PEFT Methods

Different PEFT methods offer distinct trade-offs regarding parameter efficiency, memory usage, inference overhead, and task performance. Table 2.2 quantitatively compares the major PEFT approaches. The performance and efficiency characteristics of these methods vary significantly.

Table 2.2: Quantitative Comparison of PEFT Methods

Method	Trainable Params	Inference Overhead	Performance vs. FT	Modification?
Fine-tuning	100%	None	Baseline	All parameters
Adapter [34]	1-4%	Moderate Increase	Competitive at 3% params	Additional layers between blocks
LoRA [10]	0.1-0.3% ¹	None	Competitive at 0.1% params	Low-rank matrices parallel to weights
Prompt [9]	<0.01%	Very low	Smaller scales: -5% to -30% Comparable at 10B+	Continuous prompts at input layer only
P-tuning v1 [39]	<0.01%	Very low	> discrete prompts; stabilises training	Soft & Hard prompts with LSTM/MLP reparameterisation
Prefix [8]	0.1-2%	Low	Comparable to fine-tuning for NLG	Continuous prompts in all layers
P-tuning v2 [40]	0.1-3%	Low	Matches fine-tuning across most scales	Deep prompts with optional reparam.

¹ Lower bound (0.01%) applies to extreme-scale models (e.g., GPT-3 175B). Typical RAG implementations (e.g., BART/T5) fall in the 0.1–1% range.

2.2.4.1 Fine-tuning versus Shallow Prompting:

There is an apparent scale dependence when comparing full fine-tuning to shallow prompting methods (Prompt Tuning and P-tuning v1 [9, 39]). These shallow methods significantly underperform fine-tuning on small to medium-sized models. [9] demonstrate

that on T5 models smaller than 11B parameters, Prompt Tuning generally underperforms model tuning on SuperGLUE tasks, with the gap shrinking as model size increases. At the XXL size (11B parameters), prompt tuning finally matches the performance of model tuning. Liu et al. [39] similarly observe this scale effect, showing that P-tuning’s effectiveness increases with model size, with Figure 1 in their paper showing progressive improvement across model scales. The gap between these shallow prompting methods and full fine-tuning narrows consistently as the model size increases, with comparable performance only emerging at the largest scales tested [9]. This relationship between model scale and the effectiveness of prompt-based learning appears consistent across different language model architectures [9, 39].

2.2.4.2 Deep versus Shallow Prompting:

Deep prompting approaches (Prefix Tuning [8] and P-tuning v2 [40]) substantially outperform shallow methods (Prompt Tuning [9]), especially at smaller model scales. P-tuning v2 matches fine-tuning performance across model scales as small as 330M parameters on SuperGLUE tasks [40], whereas Prompt Tuning [9] achieves comparable results only when models exceed billions of parameters. Looking at specific SuperGLUE tasks with RoBERTa-large (355M parameters), P-tuning v2 achieves strong results (e.g., 84.8 on BoolQ, 100 on CB, 93 on COPA) that are comparable to full fine-tuning (86.9, 98.2, 94.0), while Prompt Tuning significantly underperforms (62.3, 71.4, 63.0). This pattern is consistent across tasks, demonstrating that applying continuous prompts at multiple layers is crucial for effective parameter-efficient tuning of smaller models.

2.2.4.3 Adaptation for Different Tasks:

Methods also vary in their effectiveness across task types. Prefix-tuning achieves strong performance on generation tasks, closely matching fine-tuning on table-to-text generation (e.g., within 1–2 BLEU points) while using 0.1% of the parameters [8]. For summarisation, it slightly underperforms fine-tuning (e.g., 35.05 versus 37.25 ROUGE-L with 0.1% parameters), though narrowing the gap to 1–2 ROUGE-L points with 2% parameters [8]. Notably, prefix-tuning excels in low-data settings and generalises better to unseen topics, highlighting its parameter efficiency and robustness [8]. P-tuning v2 [40] demonstrates strong performance on classification and sequence labelling tasks, closing the gap to fine-tuning on challenging tasks like extractive QA, where Prompt Tuning underperforms by up to 80 F1 points on SQuAD [40].

2.2.4.4 Memory Efficiency:

LoRA offers substantial memory savings during training, reducing GPU memory requirements by up to $3\times$ on large models since one need not store optimiser states for frozen parameters [10]; this translates to a reduction from 1.2TB to 350GB of VRAM during training for GPT-3 175B [10]. Similarly, deep prompting methods like Prefix Tuning and P-Tuning v2 achieve significant memory efficiency by freezing the language model parameters and only updating a small set of continuous prompt embeddings (0.1%-3% of the total parameters) [8, 40]. While their approaches differ—LoRA adds low-rank adapter matrices in parallel to existing weights, and prompting methods prepend trainable vectors to inputs and potentially at each layer—both strategies substantially reduce memory requirements compared to full fine-tuning while maintaining comparable performance.

2.2.5 Limitations of PEFT Research Generally

Despite their computational advantages, existing parameter-efficient fine-tuning approaches exhibit notable limitations.

A methodological strength of Li and Liang [8] is their inclusion of several ablation studies examining architectural choices. The authors systematically investigate how prefix length affects task performance, compare full prefix-tuning against embedding-only variants, explore positional effects through prefix-tuning versus infix-tuning comparisons, analyse initialisation strategies, and evaluate data efficiency across training set sizes. However, their exploration has notable weaknesses in how prefix-tuning interacts with different architectural components of the transformer. Despite thorough ablations on prefix length and positioning, they do not investigate whether certain attention heads or layers benefit more from adaptation than others—an omission leaves open questions about whether more targeted, sparse adaptation strategies might achieve comparable performance with even fewer parameters. Additionally, their evaluation of generation tasks relies predominantly on surface-level metrics like BLEU and ROUGE without deeper analysis of factuality or reasoning capabilities, which is particularly crucial for knowledge-intensive applications like RAG.

Prompt-based methods have evolved rapidly. While early techniques showed inconsistent performance across model scales, recent approaches like P-Tuning v2 match fine-tuning performance universally across model sizes (330M to 10B parameters) using only 0.1-3% of trainable parameters [40]. However, despite its strengths in comprehensive cross-scale comparison, Liu et al. [40]’s evaluation methodology suffers from a task selection bias

toward classification problems rather than generation tasks. This methodological choice creates substantial uncertainty about transferability to sequence generation scenarios typical in RAG applications. Furthermore, while they claim universal effectiveness across model scales, their experiments exclude huge models ($>10\text{B}$ parameters), creating an empirical gap in understanding whether their findings extend to frontier model scales increasingly deployed in production systems.

Similarly, Hu et al. [10] conduct thorough empirical investigations of their low-rank adaptation approach, including experiments on which weight matrices to adapt (W_q , W_k , W_v , W_o) and detailed analyses of the rank-deficiency properties of adaptation matrices. Section 7 of their paper provides valuable insights into the subspace similarity between different LoRA configurations and the relationship between pre-trained weights and adaptation matrices, suggesting that LoRA “potentially amplifies the important features for specific downstream tasks that were learned but not emphasised in the general pre-training model.” While these empirical findings are substantial, a more comprehensive theoretical framework explaining why specific layers benefit more from adaptation than others would further strengthen their approach.

The low-rank decomposition approach of LoRA [10], while computationally elegant, makes a strong claim that weight updates have low intrinsic rank. While the authors provide substantial evidence supporting this approach for many NLP tasks—demonstrating competitive performance with ranks as low as 1 or 2—there remain open questions about how well this assumption generalises to scenarios with significant distribution shifts or multi-task adaptation requirements. The authors acknowledge this limitation, noting that tasks requiring substantial changes to the model (such as adaptation to entirely new languages) might benefit from higher ranks or full fine-tuning. Further research exploring the theoretical boundaries of low-rank adaptation across diverse adaptation scenarios would be valuable to the field.

Adapter layers, while effective, introduce sequential computation that cannot be easily parallelised with the base model computations, potentially increasing inference latency as documented by Hu et al. [10]. This increase in latency becomes particularly problematic in real-time applications or when working with long contexts.

These methodological gaps become particularly evident when applying PEFT methods to complex hybrid architectures like retrieval-augmented generation. The study by Ficek et al. [16] makes important strides in this direction by systematically comparing three PEFT methods (P-tuning, Adapters, and LoRA) across GPT and RETRO architectures at multiple model scales. Their comprehensive evaluation across six retrieval-based tasks reveals valuable insights, such as RETRO’s superior zero-shot performance but limited PEFT improvement ceiling compared to GPT.

While their approach of adapting the retrieval encoder and decoder components provides useful empirical findings, future work could benefit from a more granular investigation of how different components contribute to overall performance gains. For instance, a more detailed examination of how PEFT adaptations affect retrieval quality versus generation quality would deepen our understanding of these hybrid architectures. The authors acknowledge this limitation, noting that they prioritised the breadth of experiments over deeper component analysis. Nevertheless, this work represents an important step toward understanding the intersection of PEFT and retrieval-augmented architectures.

2.3 PEFT for Retrieval-Enhanced Models

A recent paper by NVIDIA researchers provides the first systematic comparison of PEFT methods applied to two distinct approaches for retrieval-enhanced language models: GPT with RAG (where retrieved documents prepend to input) and RETRO (which integrates retrieval via chunked cross-attention) [16]. Their empirical results show notable performance differences between architectures and PEFT methods:

- On average, RETRO models outperform GPT in zero-shot settings (21.79 vs. 17.65 averaged across six datasets in the Extra Large model size), but GPT models show higher performance potential with PEFT methods (e.g., LoRA: 43.72 for GPT vs. 37.41 for RETRO).
- For the Natural Questions (NQ) dataset [41], P-tuning underperforms drastically with RETRO (24.53 vs. 47.27 F1 in Extra Large models), while Adapters and LoRA maintain stronger performance across both architectures.
- Both architectures demonstrate performance saturation around 8B parameters, suggesting medium-sized models with PEFT offer an optimal efficiency trade-off.

2.3.1 Limitations of Current PEFT for RAG Research

While this NVIDIA study makes important contributions, it is extremely recent, published after our research commenced in an area lacking dedicated PEFT-for-RAG studies. Despite their substantial technical resources, several key limitations affect the generalisability of their findings:

- **Limited PEFT Spectrum:** The authors implement only shallow P-tuning with an MLP prompt encoder, overlooking deeper variants like Prefix Tuning and P-tuning v2 [8, 40] that might better address the performance gap they observed

with RETRO models. They note that “P-tuning’s weaker ability in all RETRO model sizes could lie in architecture differences” [16], suggesting deeper prompt methods could provide better results.

- **Model Scale Discrepancy:** Their experiments focus on huge models (823M to 48B parameters), whereas practical RAG implementations often use more moderate-sized models like BART-Large. The performance characteristics of PEFT methods can differ substantially across model scales, as discussed in Section 2.2.4.
- **Pre-training Scale:** Their models were pre-trained on 1.2 trillion tokens [16], significantly more than typical T5 or BART-based RAG setups trained on billions rather than trillions of tokens. This substantial difference in pre-training data volume may affect how well PEFT methods can adapt the models (and likely give them a substantial zero-shot advantage).
- **Retriever Tuning Omission:** The study leaves the external retriever completely frozen, applying PEFT only to the generator. While they tune RETRO’s integrated retrieval encoder, this fundamentally differs from tuning the external retriever component that initially selects relevant passages.
- **Dataset Selection:** Their evaluation relies on datasets (NQ [41], TriviaQA [42], NarrativeQA [43], QASPER [44], QuALITY [45], QMSum [46]) that inadequately test domain adaptation capabilities:
 - **Task variation versus domain shift:** Their selection emphasises different task capabilities (QA, summarisation, comprehension) rather than focusing on domain adaptation challenges.
 - **Wikipedia alignment:** Both NQ and TriviaQA are derived primarily from Wikipedia, which forms a substantial portion of most LLM pre-training corpora (and that of neural retrievers like DPR or in their case Dragon+); this reduces the adaptation challenge as the model has already seen similar content during pre-training.
 - **Academic familiarity:** QASPER uses NLP papers, a domain likely over-represented in LLM training data, given the technical orientation of many pre-training corpora.
 - **Missing recent domain shifts:** A more rigorous evaluation would include domains with temporal shifts (e.g., COVID-19 research) containing post-cutoff information not present in the original training data.

While these datasets provide diverse task types and reasonable evaluation benchmarks for architectural comparisons, they may not fully challenge the models' domain adaptation capabilities. Future research on domain adaptation could benefit from datasets featuring more pronounced domain shifts with specialised terminology, unfamiliar document structures, and content temporally distinct from pre-training data; this would provide deeper insights into how these PEFT methods perform when faced with significant distribution shifts—an important consideration for real-world applications requiring adaptation to highly specialised domains.

- **Architectural Disparity:** While they study GPT+RAG and RETRO, it is important to note that RETRO fundamentally differs from the classic RAG architecture defined by Lewis et al. [6]. As detailed in Section 2.1.2, RETRO integrates retrieval directly into the transformer architecture through chunked cross-attention, whereas RAG employs a separate retriever with distinct encoder components. These architectural differences significantly impact how PEFT methods interact with each system.

These limitations highlight the need for a more comprehensive exploration of PEFT methods for RAG models, particularly focusing on practical deployment scenarios with moderate-sized models, efficient dual-component adaptation, and true domain shift challenges. Our research aims to address these gaps with a more targeted approach despite having access to significantly fewer computational resources than industrial research labs.

2.3.2 Our Approach

Our approach directly addresses these research gaps by:

1. Implementing a comprehensive range of PEFT methods, including deeper prompt variants (P-tuning v2, Prefix tuning) that the NVIDIA study omitted
2. Focusing on moderate-sized models (BART-Large) that represent practical deployment scenarios for most organisations
3. Explicitly targeting domain adaptation challenges through carefully selected datasets with genuine distribution shifts
4. Developing a novel document adapter approach (detailed in Chapter 5) that specifically addresses the computational burden of knowledge base re-encoding

5. Providing a unified experimental framework that enables fair comparison across different PEFT methods specifically for the classic RAG architecture

By addressing these specific limitations, our research provides a complete understanding of how PEFT methods can make RAG systems more accessible and adaptable for resource-constrained environments.

While our document adapter approach offers significant computational advantages, it introduces several theoretical and practical limitations. First, the adapter’s expressiveness depends on the adapter’s capacity. This lightweight transformation may prove insufficient for dramatic domain shifts requiring substantial representational changes compared to full re-encoding with a domain-tuned encoder. Second, since we are adapting pre-computed embeddings rather than training them from scratch, the quality of adaptation depends partially on how well the original encoder captured the semantic structure of the documents; this creates a potential ceiling effect where the adaptation can only be as good as the initial embedding space allows.

Additionally, the periodic nature of re-indexing introduces potential staleness in the adapted embeddings between updates, which could impact retrieval quality during training. While our implementation attempts to mitigate this through efficient background re-indexing, there remains an inherent tension between computational efficiency and embedding freshness. Finally, since our approach decouples the passage encoder adaptation from query encoder training, there is a risk of developing asymmetric representation spaces, particularly for highly specialised domains with unique terminology not well-represented during pre-training.

2.3.3 RAG Adaptation Overview

Table 2.3 provides a concise overview of the theoretical and practical trade-offs of the various approaches to RAG domain adaptation.

2.3.4 Cross-Domain Comparison

The challenges facing RAG domain adaptation share intriguing parallels with research in computer vision on domain adaptation for object detection models. Just as RAG models struggle with domain-specific terminology and knowledge structures, visual recognition systems face domain shifts in lighting, perspective, and object appearance. The computer vision literature has developed innovative techniques like Adversarial Discriminative Domain Adaptation [47] that explicitly model the domain shift through adversarial

Table 2.3: Theoretical and Practical Trade-offs in RAG Adaptation Methods

Approach	Theoretical Advantages	Practical Limitations
Full Fine-tuning	Maximum expressive and theoretically optimal for domain adaptation	Resource intensive; Risk of catastrophic forgetting; Requires complete KB re-encoding
End-to-End RAG	Joint optimisation of retrieval and generation; Better alignment between components	Extremely compute-intensive; Difficult to implement efficiently; Requires multiple high-memory GPUs
Adapter-based	Modular domain specialisation; Maintains pre-trained knowledge	Increased inference latency; Sequential computation overhead; Sub-optimal for major domain shifts
LoRA for RAG	Parallel computation possible; No inference latency impact; Strong empirical performance	Assumes low intrinsic rank in weight updates; May not capture complex domain-specific transformations
Prompt-based for RAG	Minimal parameter overhead; Efficient task switching	Scale-dependent performance; May struggle with extreme domain shifts; Theoretical limitations in expressivity
Document Adapter (Ours)	End-to-end retrieval adaptation without KB re-encoding	Expressiveness bounded by adapter capacity; Quality dependent on initial embeddings; Potential for stale gradients

training. Surprisingly, similar adversarial approaches remain unexplored for aligning retrieval and generation components in RAG, despite their potential to address the retriever-generator misalignment problem identified in Subsection 2.1.7.

Similarly, research in federated learning has developed sophisticated techniques for parameter-efficient model personalisation that maintain a shared core model while adapting to local data distributions [48]. These approaches offer valuable insights for RAG adaptation, where maintaining a generic knowledge base while efficiently specialising in multiple domains presents analogous challenges. The lack of cross-fertilisation between these research areas represents a missed opportunity for advancing RAG adaptation techniques beyond the current state of the art.

Chapter 3

Experimental Methodology

This chapter details the experimental framework and evaluation methodology used to validate our parameter-efficient approach for RAG domain adaptation. We outline the overall experimental design, describe our datasets and evaluation metrics, and the computing environment. Finally, we outline the resource limitations, technical challenges faced, and measures we took to ensure reproducibility.

3.1 Focus on Prompt-Based and LoRA Methods

We chose not to investigate traditional or RAG-specific adapter-based approaches for the generator and query encoder components for several practical reasons:

- **Architectural Complexity:** Traditional adapters require decisions about placement (after which layers), bottleneck dimensions, and activation functions - introducing additional hyperparameters to tune compared to prompt-based methods and LoRA.
- **Resource Constraints:** We had a single 6GB GPU, and adapters would require additional memory during training and inference compared to our chosen methods.
- **Implementation Overhead:** Integrating adapters with the RAG architecture would require more extensive code modifications to the underlying frameworks than prompt-based and LoRA approaches, which were more readily supported by existing libraries.
- **Comparative Analysis:** Despite not investigating adapters for generator and query encoder components, our work still provides valuable comparisons between multiple parameter-efficient approaches (prompt-based methods, LoRA, and our

novel document adapter), offering insights into their relative effectiveness for RAG domain adaptation.

This strategic decision allowed us to allocate our limited computational resources to thoroughly parameter-efficient domain adaptation for RAG.

3.2 Experimental Framework

We designed our experimental framework to evaluate two key hypotheses:

1. That PEFT methods can achieve competitive performance in RAG domain adaptation compared to full fine-tuning while requiring significantly fewer trainable parameters
2. That document adaptation through our novel approach offers performance benefits over applying PEFT methods to the generator and query encoder alone

To test these hypotheses, we employ a systematic experimental design that compares different model variants and initialisation points, with implementations building upon the RAG end-to-end framework [7]. Our framework includes several foundational technical updates, with our core technical contributions detailed in Chapters 4 and 5:

- **Validation Set Random Sampling:** We implemented a stratified random sampling approach for validation due to our modest GPU (6GB). Specifically, we:
 - Sample 500 examples from the validation set using proportional stratification based on answer length distribution
 - Maintain the same sampled validation subset across all model variants to ensure a fair comparison

This sampling strategy provides a good balance between computational feasibility and reliable validation signals. Our implementation uses a fixed random seed to ensure reproducibility, and we found that increasing the validation sample size beyond 500 examples provided diminishing returns in correlation with full-set performance while significantly increasing training time.

- We adapted the retrieval precision calculation to work without golden passages.
- We adapted a P-tuning v2 reference implementation to the DPR Question Encoder.
- We adapted the rag-end2end implementation to work with the PEFT library.

- We fixed the original rag-end2end implementation to work with the latest PyTorch Lightning version.

We perform validation every third of an epoch to balance getting timely feedback on model performance and maintaining training efficiency given our GPU constraints.

3.3 Phased Experimental Approach

Due to resource constraints, we employed a two-phase experimental strategy to maximise research efficiency:

1. **Model Selection:** Initial validation experiments on a single dataset to compare architectures and validate approach viability:
 - Comparison between RAG-token-base and RAG-token-nq initialisations (finding superior performance with the latter)
 - Evaluation of shallow versus deep prompting techniques (finding superior performance with the latter)
 - Assessment of LoRA adaptations (finding them to be superior in performance to prompt-based methods)
 - Exploration of increasing adapter complexity in our document adaptation approach
 - Testing of KL divergence loss for our document adapter approach
2. **Final Evaluation:** Comprehensive experiments with the optimal architecture across all datasets:
 - Three complete runs with different random seeds for each dataset
 - Full statistical analysis of performance differences
 - Comparative evaluation against baselines

This phased approach balances thorough exploration with practical resource constraints, allowing us to focus computational resources on the most promising techniques. In the Results chapter, we summarise the model selection experiments and comprehensively evaluate our final approach.

3.4 Datasets and Evaluation Methodology

3.4.1 Dataset Selection and Processing

We utilise the same datasets as Siriwardhana et al. [7] to enable meaningful comparison with existing end-to-end RAG approaches. The researchers evaluated their RAG-end2end approach using three diverse domain-specific datasets. The **COVID-19 QA** dataset was built from scientific literature in the CORD-19 corpus, with synthetic training data and human-labeled test questions. The **News QA dataset** utilised CNN/DM news articles with human-annotated questions focusing on high-level content reasoning. The **Conversation QA** dataset uses multi-party dialogues in the QAConv dataset, with questions specifically about conversational content. Each domain had its knowledge base of passages and appropriate reconstruction signals derived from abstracts, summaries, or generated conversation synopses.

The choice of these datasets is motivated by two key factors:

1. The datasets represent non-Wikipedia domain adaptation scenarios, aligning with real-world use cases.
2. Their established baseline performance provides valuable context for comparing our PEFT results to fully tuned evaluations (which we lacked the resources to perform ourselves).

We use the Conversation QA (ConvQA) dataset for the model selection experiments because it represents a substantial domain shift from Wikipedia-based pretraining. It requires adaptation to conversational structures and dialogue-specific language patterns that differ significantly from the encyclopedic knowledge the base models were originally trained on.

3.4.2 Evaluation Metrics

We implement and utilise three primary evaluation metrics:

- **Exact Match (EM)**: Measures the percentage of generated answers that exactly match the reference answer after normalisation.
- **F1 Score**: Computes the overlap between generated and reference answers at the token level.

- **Enhanced Retrieval Precision@k:** We describe our modified implementation of the retrieval precision metric in Subsubsection 3.4.2.1, which addresses several limitations of the standard one.

Regarding reproducibility, it is vital to note that we diverge from the official implementation of Siriwardhana et al. [7] due to its reliance on unpublished gold reference passages - contrary to the description given in the paper [7]. Instead, we have implemented the metric described in the paper (checking for matches between ground truth answers and retrieved documents), making it fully reproducible while maintaining its effectiveness. Our retrieval metric implementation provides several key improvements over the original work:

- Fuzzy matching using token set ratios to handle semantic variations
- Dynamic threshold adjustment based on answer length
- No reliance on golden passages

3.4.2.1 Evaluation Metrics Formal Definitions

Our evaluation metrics are formally defined as follows:

Exact Match (EM):

$$\text{EM}(a, a^*) = \begin{cases} 1, & \text{if } \text{normalise}(a) = \text{normalise}(a^*) \\ 0, & \text{otherwise} \end{cases} \quad (3.1)$$

where a is the generated answer, a^* is the reference answer, and $\text{normalise}(\cdot)$ performs lowercasing, punctuation and article removal.

F1 Score:

$$\text{F1}(a, a^*) = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.2)$$

Where:

$$\text{Precision} = \frac{|\text{tokens}(a) \cap \text{tokens}(a^*)|}{|\text{tokens}(a)|} \quad (3.3)$$

$$\text{Recall} = \frac{|\text{tokens}(a) \cap \text{tokens}(a^*)|}{|\text{tokens}(a^*)|} \quad (3.4)$$

Custom Retrieval Precision@k:

$$\text{Retrieval Precision@k} = \frac{\text{correct_retrievals}}{|Q|} \times 100\% \quad (3.5)$$

$$\text{correct_retrievals} = \sum_{q \in Q} \mathbb{1} [\exists i \in \{1, 2, \dots, k\} : \text{match}(d_{q,i}, a_q^*)] \quad (3.6)$$

$$\text{match}(d, a) = \begin{cases} \text{True} & \text{if } a \subseteq d \\ \text{True} & \text{if } \text{token_set_ratio}(a, d) \geq \tau(a) \\ \text{False} & \text{otherwise} \end{cases} \quad (3.7)$$

$$\tau(a) = \begin{cases} \max(\text{threshold} + 10, 85.0) & \text{if } \text{len}(a.\text{split}()) \leq 2 \\ \text{threshold} & \text{otherwise} \end{cases} \quad (3.8)$$

where q is the query, d_i is the i -th retrieved document, a^* is the reference answer, $\mathbb{1}[\cdot]$ is the indicator function, $\text{fuzzy_match}(\cdot, \cdot)$ computes token set ratio, and $\tau(a^*)$ is a dynamic threshold based on answer length.

3.4.2.2 Statistical Significance Testing

To determine whether observed performance differences between our PEFT methods and baseline are statistically significant, we employ a bootstrapped paired t-test approach that accommodates our resource constraints [49, 50]:

1. We perform three training runs with different random seeds for each model variant.
2. For each model pair comparison, we create 1,000 bootstrap samples by random sampling with replacements from the test examples.
3. For each bootstrap sample and each sampled example, we randomly select one of the three runs, preserving run-to-run variability.
4. For each bootstrap sample, we compute the mean difference in performance metrics between the selected runs.
5. We calculate the 95% confidence interval from these differences' distribution using the bias-corrected accelerated (BCa) bootstrap method to account for potential skewness [51, 52].
6. If the confidence interval does not include zero, we consider the difference statistically significant.

We use the original `rag-token-nq` as the baseline just as Siriwardhana et al. [7] did.

3.5 Implementation Challenges and Limitations

We encountered and addressed several significant technical challenges:

- **Training Document Limit:** Due to hardware constraints, we are limited to feeding the generator five retrieved documents; this could hinder model performance.
- **Memory Management:** We implemented random sampling of the validation set to make training times tractable - we needed to use a subset of 500 examples.
- **Baseline Comparability:** Due to these constraints and limited documentation of hyperparameters in prior work [7], our absolute performance numbers are not directly comparable to published results. Instead, we focus on relative performance improvements over consistent baselines.
- **Dual Approach to Variability:** Our approach addresses both model initialisation variance (through multiple training runs) and test set sampling uncertainty (through bootstrapping); this provides a more complete picture of statistical significance while remaining computationally feasible within our hardware constraints.

Despite these limitations, our experimental framework remains valid for comparing the relative effectiveness of different parameter-efficient adaptation techniques as we maintain consistent conditions across all experiments.

3.6 Computing Infrastructure

We ran our experiments on a single-GPU workstation with the following relevant specifications:

- GPU: NVIDIA GeForce GTX 1060 6GB (Pascal architecture)
- CPU: Intel Core i5-6500 (4 cores @ 3.6GHz)
- RAM: 24GB DDR4
- Storage: 1TB Western Digital HDD + 120GB SanDisk SSD

This modest hardware setup reflects the resource constraints some face when deploying RAG systems, making it an appropriate testbed for evaluating parameter-efficient adaptation approaches.

3.7 Reproducibility Considerations

To ensure reproducibility, we have documented several important technical details that were either ambiguous or missing in prior work:

- Fixed random seeds for all randomised processes
- Explicit documentation of all hyperparameters
- Reproducible retrieval metric implementation

Chapter 4

Artefact Design and Implementation

This chapter details the design and implementation of our research artefact for parameter-efficient RAG domain adaptation.¹ The artefact has several interlocking components that form a comprehensive framework for experimenting with and evaluating parameter-efficient approaches to domain adaptation. Although Chapter 5 outlines our new document adaptation technique (part of the artefact), this chapter focuses on the broader technical infrastructure that enabled our experiments.

4.1 Integration with RAG-end2end Framework

Our artefact builds upon the RAG-end2end framework, which provides core functionality for training and evaluating RAG models.² Understanding how our modifications integrate with this existing framework is crucial for appreciating the technical contributions of our work.

The RAG-end2end framework provides several essential components that our implementation leverages:

- A RAG training pipeline built on PyTorch Lightning
- Dataset loading and preprocessing utilities
- KB creation and indexing tools
- Evaluation metrics calculation

¹Thesis GitHub Repository

²RAG-end2end Framework GitHub Repository

4.1.1 Fine-tuning Pipeline Modifications

A central integration component adapts the framework’s `finetune_rag.py` module, which orchestrates the training pipeline. Our implementation in `finetune_rag.py` extends this module with parameter-efficient capabilities through several critical modifications:

- **Custom Dataset Integration:** We replace the original `Seq2SeqDataset` with our `RandomSamplingSeq2SeqDataset` to enable stratified sampling for resource-constrained validation.
- **PEFT Module Substitution:** We substitute the framework’s `GenerativeQAModule` with our parameter-efficient variants (`PEFTGenerativeQAModule` or `AdaptivePEFTGenerativeQAModule`) (our document adapter version), enabling seamless integration of PEFT techniques within the existing training flow.
- **Reproducibility Enhancements:** While the original RAG-end2end framework used PyTorch Lightning’s `seed_everything()` function, our implementation extends this with a more comprehensive `set_seed()` function that coordinates random state across additional components. This function:
 - Maintains the PyTorch Lightning seeding from the original framework
 - Adds explicit `torch.cuda.manual_seed_all(seed)` to ensure GPU operations use the same random values
 - Configures `torch.backends.cudnn.deterministic = True` for consistent algorithm selection
 - Disables CuDNN benchmarking to prevent optimization-induced variations
 - Sets the `PYTHONHASHSEED` environment variable to maintain consistent dictionary ordering

These additional seeding controls provide stronger reproducibility guarantees for our experiments on the same hardware configurations, which is critical when comparing parameter-efficient techniques and during document adaptation re-indexing. However, we cannot guarantee perfect determinism across different hardware configurations due to inherent differences in floating-point arithmetic implementations and parallel processing behaviours.

- **PyTorch Lightning Compatibility:** We implemented compatibility fixes for deprecated PyTorch Lightning functionality, addressing a critical issue documented in the Hugging Face transformers repository (Issue #25525³). Specifically, newer

³GitHub Issue #25525: End2end pytorch lightning errors

versions of PyTorch Lightning removed support for the `add_argparse_args` and altered the behaviour of the `from_argparse_args` methods, on which the original RAG-end2end framework relied. Our solution involves implementing custom replacements for these functions

These modifications enable our artefact to maintain full compatibility with the existing RAG-end2end pipeline while introducing parameter-efficient capabilities. Rather than creating an entirely new training loop, our implementation tactically modifies key components of the existing framework to accommodate our specialized PEFT and document adaptation techniques.

Our modifications and extensions integrate seamlessly with these components while expanding their capabilities for PEFT approaches.

4.2 Stratified Random Sampling for Resource-Constrained Validation

A fundamental challenge in our experimental setup was conducting meaningful validation on hardware with only 6GB of GPU memory. The original implementation’s validation approach proved computationally prohibitive, leading us to develop a stratified random sampling technique that maintained representativeness while reducing the validation set size. Our implementation, found in `process_dataset.py`, addresses several key requirements:

- Ensuring balanced representation across different answer length distributions
- Maintaining consistent sampling across training epochs for reliable model comparison
- Providing reproducibility through fixed random seeds
- Integrating seamlessly with the existing data pipeline

The core implementation extends the original `Seq2SeqDataset` class from the RAG-end2end framework with stratified sampling capabilities through our `RandomSamplingSeq2SeqDataset` class. This extension slots directly into the data pipeline by overriding the framework’s dataset class in the training process. Our sampling implementation provides three critical functionalities:

1. **Answer Length Stratification:** The class analyzes the distribution of answer lengths in the validation set and divides examples into bins based on these lengths; this ensures representative sampling across different answer complexities.
2. **Consistent Sampling:** By creating a persistent index mapping between the reduced validation set and the original data, the implementation ensures that the same subset is used consistently across training epochs; this enables reliable comparison of model performance over time.
3. **Proportional Representation:** The sampling algorithm ensures that the number of examples drawn from each bin is proportional to its size in the original dataset, maintaining the natural distribution of answer lengths. This approach allows us to reduce the validation set from over 3000 examples (in the case of ConvQA) to just 500 while maintaining a representative answer length distribution. Ad-hoc preliminary experimentation confirmed that - while reducing validation time significantly - this sampling approach provides validation signals that correlate strongly with full-set performance.

Additionally, we resolved a critical issue in the RAG-end2end KB creation pipeline related to handling maximum sequence lengths. The original implementation failed with inconsistent length errors during tokenization. We addressed this by implementing a `CustomDPRTokenizer` class that enforces consistent maximum length settings of 512 tokens. This implementation integrates with the `embed_update` and `add_index` functions from the RAG-end2end framework’s `kb_encode_utils` module through a patched tokenizer factory method. By intercepting the tokenizer creation process, our solution ensures that all tokenization operations have a consistent maximum length, resolving unpredictable truncation behaviour causing pipeline failures during document processing. The sampling and tokenization improvements form a foundational component of our technical artefact, enabling stable and reproducible experimentation despite hardware constraints while maintaining the integrity of the validation process.

4.3 Integration of PEFT

Our implementation leverages Hugging Face’s PEFT library, extending the capabilities of the existing RAG-end2end framework.⁴ Our PEFT integration, implemented in `peft_module.py`, addressed several challenges inherent in combining PEFT with the ragend2end framework.

⁴PEFT GitHub Repository

- **Automatic Model “Peftification”:** Our `PEFTGenerativeQAModule` class extends the `GenerativeQAModule` from the RAG-end2end framework, automatically applying PEFT configurations (such as prefix-tuning and LoRA) to both the generator and question encoder; this allows seamless experimentation and swapping of different parameter-efficient techniques without extensive manual changes.
- **Robust Handling of None Inputs During Generation:** A critical issue encountered after applying PEFT’s encoders to the generators was that its forward would fail during validation and inference. The PEFT library’s batch size detection mechanism requires either `input_ids` or `inputs_embeds`, but during BART’s autoregressive generation, these could become `None`, causing training to fail.

Our modified forward method specifically addresses this by:

1. Adding explicit checks and graceful handling for `None` values of inputs.
2. Ensuring a robust determination of batch size by conditionally utilizing available decoder inputs or embeddings.
3. Properly appending prompt embeddings to the encoder and decoder inputs, avoiding tensor shape mismatches during inference.

This custom forward method is dynamically patched onto the base BART generator model after applying PEFT (see `peftify_generator` method), thereby resolving validation errors related to missing inputs during generation.

- **Removal and Overriding of Legacy Hooks:** The original RAG-end2end framework relied on deprecated PyTorch Lightning hooks (such as `validation_epoch_end` and `test_epoch_end`), which became incompatible with recent PyTorch Lightning versions. Our implementation explicitly removes these outdated hooks and introduces updated mechanisms for metrics collection and tracking of training steps.
- **Checkpointing and Metrics Improvements:** To support parameter-efficient tuning, we customized the checkpointing process (`on_save_checkpoint`) to handle PEFT-enhanced models specifically; this involves explicitly saving and reconstructing the state dictionaries for the generator and question encoder, ensuring checkpoints remain consistent and usable. We revised the training loop to accurately count training batches rather than relying solely on validation step counts; this proved critical when deciding when to re-index during the document adaptation described in Chapter 5.

Regarding integration points with the original RAG-end2end framework, our modifications directly slot into the existing `GenerativeQAModule` infrastructure. Specifically, we

reuse existing dataset loading utilities, retriever initialization logic, and the overall PyTorch Lightning training pipeline, thereby maintaining compatibility with the broader RAG workflow.

4.4 Custom P-tuning v2 Implementation for DPR

While the PEFT library implements several PEFT techniques, it only offers shallow prompt tuning (P-tuning) for BERT-type models rather than the more sophisticated P-tuning v2 approach. This limitation necessitated our custom implementation of P-tuning v2 specifically tailored for Dense Passage Retrieval (DPR) question encoders. Our implementation adapts the reference implementation to accommodate DPR’s BERT-based architecture, making several critical modifications⁵.

The key challenge was adapting P-tuning v2 from its original span prediction task (used in question-answering) to the embedding generation task required for DPR’s retrieval mechanism; this required fundamentally reorienting the model architecture to produce dense embeddings suitable for similarity matching rather than token classification or answer span identification.

Our implementation consists of three primary classes that work together to enable P-tuning v2 for DPR: a foundational `PrefixEncoder`, a DPR-specific `DPRPrefixEncoder`, and a top-level `DPRPrefixQuestionEncoder` that manages the forward pass for embedding generation.

This implementation successfully bridges the gap between the reference implementation’s span prediction task and DPR’s embedding generation requirements. It maintains the parameter efficiency of P-tuning v2 while enabling its application to retrieval tasks, providing a foundation for our domain adaptation experiments within the resource constraints of our experimental setup.

Full details of the implementation, including technical challenges, architectural adaptations, and divergence from the reference implementation, are provided in Appendix A.

4.5 Evaluation Artefact

Our evaluation implementation extends the original RAG-end2end framework’s evaluation utilities with enhanced statistical analysis capabilities, custom metric implementations, and support for parameter-efficient models. The implementation, found in

⁵P-tuning-v2 GitHub Repository

`eval.py`, addresses several key requirements for robust evaluation of parameter-efficient RAG models, including statistical significance testing across configurations, improved retrieval precision calculation without requiring golden passage annotations, and support for multi-seed evaluation to assess model stability.

4.5.1 PEFT Model Loading

A central component of our evaluation implementation is the `load_peft_rag_model` function, which handles the complexities of loading parameter-efficient RAG models. This function intelligently determines the type of PEFT adaptation (LoRA or prefix tuning) by examining the configuration files in the checkpoint directory. Our implementation supports loading models with different PEFT techniques applied to different components—for example, a model might use LoRA for the generator and prefix tuning for the question encoder.

We implemented support for loading from both traditional binary checkpoints (`.bin`) and the more secure `safetensors` format, enhancing compatibility with recent Hugging Face model conventions. When loading baseline models, the function automatically uses the standard loading procedure, maintaining backward compatibility with non-PEFT models through a graceful fallback mechanism that ensures evaluation workflows remain consistent regardless of model type.

4.5.2 Enhanced Retrieval Precision

Our implementation significantly improves the retrieval precision calculation through the `patched_get_precision_at_k` function, which addresses a critical limitation in the original implementation—the reliance on golden passage annotations that were not publicly available. Instead of requiring explicit golden passage IDs, our implementation derives relevance by checking if retrieved passages contain the expected answer, making the evaluation reproducible with only the knowledge base and ground truth answers.

To support this content-based approach, we implemented the `efficient_fuzzy_match` function using the `RapidFuzz` library, which employs token set ratio algorithms to handle semantic variations, word reordering, and minor errors in the text. Our implementation automatically adjusts matching thresholds based on answer length, applying stricter thresholds for short answers (\leq two tokens) to prevent false positives while maintaining sensitivity for longer answers. Document content is cached after initial loading, reducing repeated disk I/O operations during evaluation.

4.5.3 Statistical Significance Testing

Implementing robust statistical significance testing through the `bootstrap_significance_test` function is a key enhancement in our evaluation framework. Our implementation specifically accounts for between-run variability by incorporating results from multiple training runs with different random seeds. Rather than using simple percentile-based confidence intervals, we employ the BCa (bias-corrected accelerated) bootstrap method, which adjusts for skewness in the bootstrap distribution and provides more accurate confidence intervals.

Our bootstrapping approach samples at the example level rather than aggregating metrics first, preserving the correlation structure between baseline and comparison model performance across individual examples. The function returns the statistical significance determination, mean difference, and confidence intervals, facilitating a more nuanced interpretation of results. This comprehensive approach to significance testing is particularly valuable when comparing parameter-efficient techniques with varying degrees of representational capacity.

4.5.4 Integration with Original Framework

Our evaluation implementation strategically uses Python’s monkey patching technique to integrate with the original RAG-end2end framework without requiring extensive modifications to its codebase. The `patch_get_args` and `patch_main` functions replace the original framework’s argument parsing and main evaluation loop with our enhanced versions, while our content-based implementation overrides the original `get_precision_at_k` function while maintaining the same function signature for compatibility.

Throughout our implementation, we carefully preserve the overall evaluation flow of the original framework, ensuring that existing evaluation scripts continue to work with our extensions. This non-invasive integration approach maintains compatibility with the broader RAG-end2end framework while extending its capabilities to support parameter-efficient models and enhanced statistical analysis. The result is a seamless evaluation experience that requires minimal adaptation for users familiar with the original framework yet provides substantial new capabilities for assessing parameter-efficient RAG adaptation techniques.

Chapter 5

Novel Document Adapter

This chapter introduces our novel document adapter approach, which addresses one of the key bottlenecks in RAG domain adaptation—the need to re-encode the entire knowledge base during training repeatedly. After outlining the problem, we present our architectural solution, describe its training methodology, and discuss the theoretical foundations that support our approach.

5.1 Motivation

As Chapter 2 highlights, one of the most significant resource barriers to adapting RAG systems to new domains is the computational expense of re-encoding the knowledge base (KB). In the traditional end-to-end RAG adaptation approach [7], both the question encoder and document/context/passage encoder are updated during training, which requires re-encoding and re-indexing the entire KB to reflect these updates.

This chapter’s central research question is: *How can PEFT methods be applied to eliminate the need for KB re-encoding when adapting RAG retrieval end to end?*

5.2 Architectural Overview

Our novel document adapter approach enables end-to-end retrieval adaptation without requiring frequent re-encoding of the KB by introducing a trainable adapter module between the original passage representations and their use in retrieval. Figure 5.1 illustrates this approach.

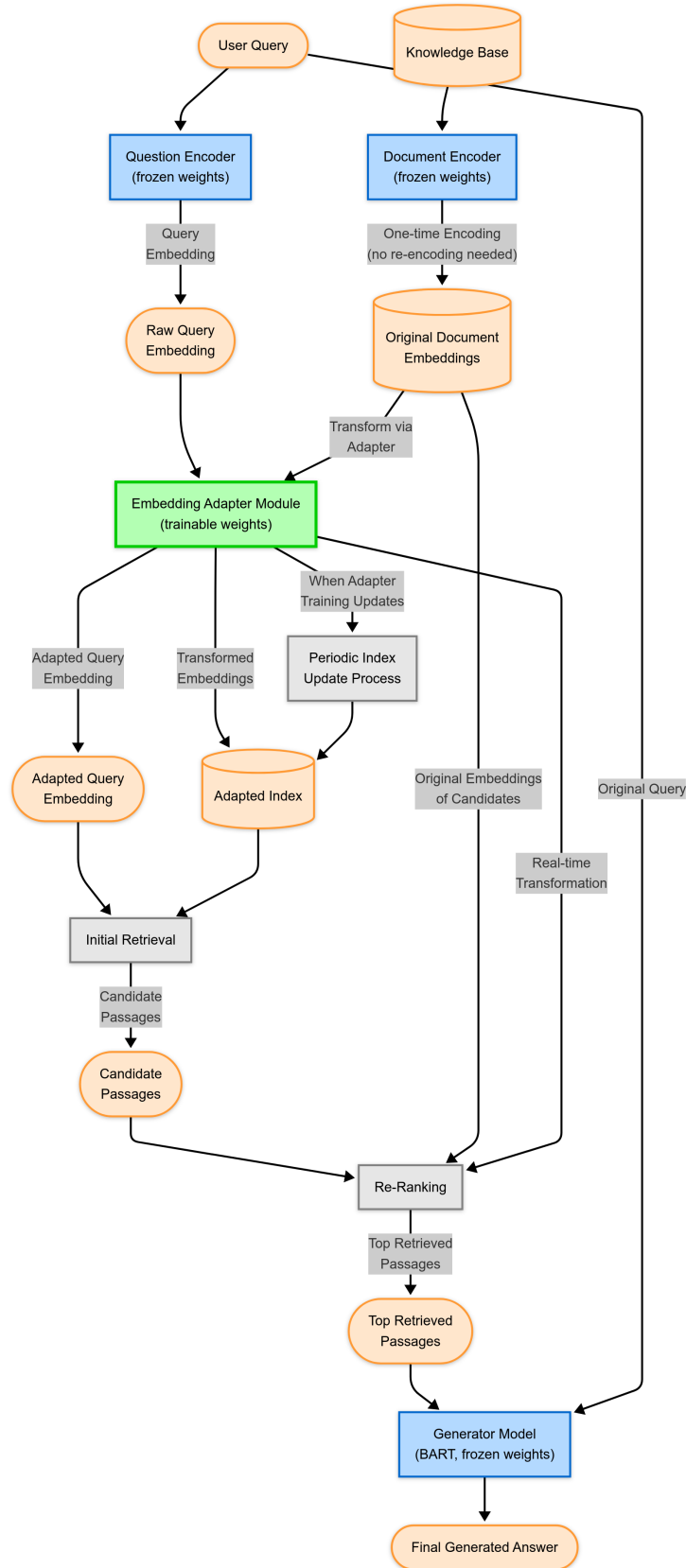


Figure 5.1: Overview of the novel document adapter approach. **Elements are colour coded:** blue represents frozen model components with fixed weights; green indicates trainable components; orange represents data elements; grey shows processes.

5.2.1 Embedding Adapter Module

The core of our approach is a learnable transformation module that adapts query and document embeddings to a new domain-specific embedding space. We implemented and evaluated three different adapter architectures:

- **ShallowMLPEmbeddingAdapter:** A lightweight adapter with a single hidden layer.
- **DeepMLPEmbeddingAdapter:** A more complex architecture with multiple hidden layers, providing greater representational capacity for complex domain shifts.
- **TransformerEmbeddingAdapter:** A lightweight transformer with a single self-attention layer with four heads, enabling more sophisticated context-aware transformations.

Inspired by LoRA [10], all adapters employ residual connections and zero-initialisation of weights, ensuring that the adapter outputs match their inputs exactly initially; this allows the model to learn a domain shift Δ rather than embeddings from scratch, starting from the pre-trained embeddings and gradually learning domain-specific transformations without disrupting early training stability.

5.2.2 Enhanced Retrieval Process

Our approach transforms the standard retrieval process into a two-stage procedure:

1. **Initial Retrieval:** The adapted query retrieves a larger set of candidate passages (e.g., 400) from the adapted index.
2. **Re-Ranking:** The retrieved passages are fetched from the original embedding store, transformed by the adapter, and re-ranked based on inner product similarity with the adapted query.

5.2.2.1 Design Rationale

With regular end-to-end training, the passage encoder learns new embeddings for the entire context at each training step. To best-mimic this with our approach, we fetch more documents than needed and teach the adapter a transformation Δ that would promote

more useful documents for QA to the top of the similarity rankings and deprioritise less useful ones. With this alone, the adaptation of the context space would be limited to this initial similarity “bubble” for each query as defined by the initial embeddings. However, where our approach differs from a standard re-ranking mechanism is that we allow the entire context space to be adapted periodically from what the adapter has learned - we describe this in Subsection 5.3.1

5.2.2.2 Implementation Details

We transform queries and documents to reside in the same latent space. Using the transformed index to decide which documents to retrieve but retrieving the original embeddings enables the adapter to transform the context space while only needing to learn a single domain transformation; this avoids a “moving target” problem and error accumulation.

A key technical challenge was maintaining the backward pass through the adapted embeddings while still using FAISS for efficient similarity search, which does not support gradients. Our solution creates a differentiable similarity computation between the adapted query and adapted documents for the top-k selection, preserving gradients while leveraging FAISS for the initial broader retrieval.

We implemented this through a custom `EncoderWithAdapter` class and a “enhanced_retrieve” function that replaces the standard retrieval method during training. This function intercepts the query embedding, processes it through the adapter, maintains a detached copy for initial retrieval, and ensures proper gradient propagation during re-ranking.

The `EncoderWithAdapter` class wraps around the original DPR question encoder. This approach maintains a clean computation graph without requiring direct gradient manipulation. The module first passes inputs through the original encoder, which has frozen weights. It then applies the embedding adapter to the output embeddings. This design ensures compatibility with DPR’s API, allowing seamless integration into existing systems. This architecture effectively extends the original encoder’s functionality while preserving its core behaviour and interface requirements.

5.3 Training Methodology

Our training approach combines parameter-efficient adaptation with a novel periodic re-indexing strategy to avoid the computational burden of re-encoding the entire KB

during training.

5.3.1 Periodic Index Updates

A key innovation in our approach is the periodic re-indexing strategy, which enables efficient updating of the KB index without re-encoding the original passages:

1. Every `reindex_frequency` steps (typically 750), we save the current adapter state
2. We launch a separate process (`update_index_process`) that:
 - Loads the stored adapter model
 - Loads the original document embeddings from the KB
 - Applies the adapter transformation to these embeddings
 - Builds a new FAISS index with the transformed embeddings
3. After completion, we swap in the new index and reload the retriever

5.3.1.1 Design Rationale

The asynchronous processing approach prevents training interruptions that would occur with synchronous re-indexing, maintaining consistent training dynamics while ensuring the index gradually aligns with the evolving adapter. This design prioritises efficient resource utilisation while preserving the end-to-end training signal critical for effective domain adaptation.

5.3.1.2 Implementation Details

Our implementation leverages Python’s multiprocessing library to run indexing separately, preventing training interruptions. The implementation includes safeguards to handle process failures, atomically swap in new indices, and maintain backup copies of previous indices.

The FAISS index construction is optimised using OpenMP parallelism and uses an HNSW (Hierarchical Navigable Small World) graph structure with an inner product metric.

5.3.2 Autoencoding-based Retriever Training

Inspired by recent work on unsupervised retriever training [53], we explored an Auto-encoding-based Retriever Training (ART) approach as one component of our training methodology after encountering optimisation challenges with the standard RAG loss. Our adapter likely required a more direct retrieval relevance signal during training. However, Siriwardhana et al. [7] had not made their golden passage annotations available. With ART, we could provide a loss that was more directly related to retrieval relevance in an unsupervised manner. In ART, we:

1. Use the current query encoder to obtain question embeddings
2. Retrieve passages using the adapted embeddings
3. Compute reconstruction scores measuring how well each passage can reconstruct the original query
4. Calculate KL divergence between the “ideal” distribution (based on reconstruction probabilities) and the current similarity-based retrieval distribution
5. Optimise the adapter to align retrieval scores with reconstruction quality

We compute reconstruction scores using the generator’s encoder-decoder architecture, feeding the passage to the encoder and evaluating how well the decoder predicts the original question tokens. Temperature-scaled softmax normalisation (set to 0.1 after experimentation) controls the sharpness of the probability distributions before computing KL divergence, balancing exploration and exploitation during training.

5.4 Theoretical Foundation

Our document adapter approach is grounded in several theoretical principles that collectively enable efficient domain adaptation:

5.4.1 Embedding Space Transformation

The core assumption of our approach is that domain adaptation can be achieved through a relatively low-dimensional transformation of the embedding space rather than requiring a complete re-encoding of the KB; this aligns with recent research showing that domain shifts often manifest as systematic transformations in the embedding space [10].

The adapter effectively bridges the gap between the pre-trained and target domain embedding space without modifying the original encoders by learning this transformation.

5.4.2 Gradient Flow Preservation

A critical aspect of our implementation is preserving gradient flow through the adapter during training and inference. Our enhanced retrieval process ensures that gradients propagate correctly through query and document adaptation paths, enabling effective learning despite the two-stage retrieval process.

5.4.3 Index Staleness Management

Our periodic re-indexing approach addresses the challenge of index staleness, where the index represents embeddings transformed by an outdated adapter version. We maintain alignment between the training and retrieval processes by periodically updating the index with embeddings transformed by the current adapter.

The frequency of re-indexing represents a trade-off between computational efficiency and index freshness. We decided on 750 steps as this was the average re-index frequency observed by Siriwardhana et al. [7] in their experiments.

5.5 Framework Integration

Our document adapter implementation integrates with existing RAG architectures with minimal disruption while maintaining training efficiency.

The `EncoderWithAdapter` class provides a wrapper around the original DPR question encoder, maintaining a clean computation graph without requiring direct gradient manipulation. This approach ensures compatibility with DPR’s API, allowing seamless integration into existing systems.

Our implementation extends the existing RAG training framework through the `AdaptivePEFTGenerativeQAModule` class, which inherits from our `PEFTGenerativeQAModule`, overriding only the necessary methods while preserving most of the original training logic. This inheritance-based design ensures compatibility with existing checkpointing, evaluation metrics, and training workflows.

A mix of object composition and monkey patching intercepts the necessary method calls without changing the core RAG architecture. This approach maintains compatibility with the broader ecosystem while enabling our specific adaptations.

Custom checkpointing logic saves both the model and adapter state independently, allowing flexible loading of either component. This separation enables mixing different adapters with various pre-trained models, supporting continued experimentation without full retraining.

5.6 Limitations

While our document adapter approach offers significant computational advantages, it does have several theoretical and practical limitations:

- **Expressiveness Ceiling:** The adapter’s architecture and capacity bounds its expressivity, which may be insufficient for dramatic domain shifts.
- **Initial Embedding Quality:** The effectiveness of adaptation depends partly on the quality of the original embeddings—if particular domain-specific distinctions are absent in the original embedding space, the adapter may struggle to recover them.
- **Index Staleness During Training:** Despite periodic updates, there is some inherent staleness in the index between updates, which may affect training stability.
- **Asymmetric Representation Learning:** Since we train the query encoder and document adapter separately, there is a risk of developing asymmetric representation spaces for queries and documents.

5.7 Summary

This chapter presented our novel document adapter approach for parameter-efficient RAG domain adaptation. By introducing a trainable transformation between the original and domain-specific embedding spaces and an efficient periodic re-indexing strategy, our approach eliminates the need to re-encode the entire knowledge base during training repeatedly.

The described architecture and methodology directly address our research question of enabling end-to-end retrieval adaptation without prohibitive computational requirements. In the next chapter, we evaluate this approach empirically, comparing it against conventional fine-tuning and other parameter-efficient techniques.

Chapter 6

Evaluation

6.1 Model Selection Experiments

As described in Chapter 3, due to our constrained resources, we conduct initial validation experiments on a single dataset (Conversation QA) to validate design elements before conducting the final evaluation. These experiments allowed us to iterate on architectural decisions quickly and identify potential stability issues before committing our limited GPU time to complete runs for all datasets.

6.1.1 RAG-token-base versus RAG-token-nq Initialisations

The first set of experiments compares two initial checkpoints offered by the original Hugging Face RAG work: The default **rag-token-base** checkpoint is pre-trained on general corpora (chiefly English Wikipedia), whereas **rag-token-nq** is further adapted for open-domain QA on the Natural Questions (NQ) dataset. We applied prefix tuning and p-tuning v2 in both instances to the generator and query encoder, using a learning rate of 5e-5 as recommended by Li and Liang [8].

From our comparison (detailed in Appendix B), two important observations emerged:

- With the **rag-token-base** checkpoint, the model struggled to exceed near-zero exact match scores even after several epochs.
- In contrast, the **rag-token-nq** initialisation rapidly improved on the same data, achieving an exact match (EM) of around 5–6% by step 30.

These results confirm our hypothesis that **rag-token-nq**, given its prior exposure to QA-style tasks, yields more effective representations for new QA domains. Hence, for all subsequent experiments, we adopt **rag-token-nq** as the starting checkpoint for both the retriever and generator components.

6.1.2 Shallow versus Deep Prompting Techniques

Following our choice of **rag-token-nq** initialisation, we investigated whether shallow or deep prompt tuning would be more effective for our parameter-efficient domain adaptation approach. Our literature review findings motivated this comparison, which suggests that deep prompting performs better on smaller models than shallow prompting methods.

Our comparison of deep prompting (prefix tuning) against shallow prompted alternatives (P-tuning v1) revealed significant performance differences (see Appendix B for detailed results):

- Deep prompt tuning achieved substantially higher exact match scores (peaking at 5.6%) than shallow prompt tuning (remaining at or below 1.2%).
- Deep prompting showed consistent improvement patterns with gradual gains, while shallow prompting appeared unstable with fluctuating metrics.

These results align with findings from the literature: deep prompting techniques substantially outperform shallow approaches, especially for smaller pre-trained models [8, 40]. Modifying representations at every layer of the transformer appears crucial for effectively steering the model’s behaviour during domain adaptation.

6.1.3 Assessment of LoRA adaptations

Following our evaluation of deep prompting techniques, we explored LoRA as an alternative parameter-efficient approach. LoRA modifies existing weight matrices through low-rank decomposition rather than introducing new tokens or embeddings.

We applied LoRA to both the generator and question encoder components using configurations targeting query and value projection matrices, with a learning rate 2e-04 based on Hu et al. [10]’s findings for transformer models of comparable size.

Compared to deep prompt tuning, our LoRA experiments (detailed in Appendix B) demonstrated:

- Significantly higher exact match scores (peaking at 8.6% compared to 5.6% for deep prompting), representing a 54% relative improvement.
- Faster convergence, reaching comparable performance levels in fewer training steps.
- More stable performance with minimal fluctuation in later training steps.

The superior performance of LoRA aligns with findings suggesting that LoRA’s approach to parameter adaptation may be more effective for specific model architectures. Both methods tune a tiny fraction of the model’s parameters (approximately 0.1-0.3%), but LoRA’s ability to directly modify weights rather than introducing additional tokens appears advantageous for the RAG architecture.

6.1.4 Document Adapter Architecture Experiments

Having established LoRA as our primary generator and question encoder adaptation method, we investigated the impact of adapter architecture complexity on our novel document adaptation approach. We compared three adapter architectures: standard MLP, deeper MLP, and transformer-based adapters.

Key findings from these experiments (see Appendix B for detailed results):

- The transformer-based adapter significantly outperformed both MLP variants, achieving exact match scores 150-300% higher than the standard MLP adapter.
- Surprisingly, the deeper MLP adapter performed worse than the standard MLP, suggesting that adding depth without more sophisticated transformation mechanisms may hinder adaptation.
- Despite the transformer adapter’s superior performance among tested architectures, it still fell short of the baseline performance achieved without any document adapter.

We also explored Autoencoding-based Retriever Training (ART) with Kullback-Leibler (KL) divergence loss as an alternative training objective. While this approach showed modest improvements in peak performance compared to standard cross-entropy loss, it did not fundamentally overcome the limitations of our document adaptation approach.

Based on the collective results from our model selection experiments, we decided to proceed with the LoRA adaptation of the generator and question encoder components for our final evaluation across all datasets without including the document adaptation

approach. While our document adapter explorations provide valuable insights into the challenges of parameter-efficient KB adaptation, the LoRA approach offers substantially better performance (8.6% EM) and training stability.

6.2 Final Evaluation

Having established LoRA as our most effective parameter-efficient adaptation technique through the model selection experiments, we present a comprehensive evaluation across all three datasets: COVID-19 QA, News QA, and Conversation QA. This section provides a detailed analysis of our final results, comparing the performance of our LoRA-based approach against the baseline `rag-token-nq` model.

6.2.1 Experimental Setup

For our final evaluation, we applied LoRA to both the generator and question encoder components using the configurations established in Section 6.1.3:

- **Generator LoRA Configuration:** Rank 4, alpha 16, dropout 0.1, targeting query and value projection matrices (`q_proj` and `v_proj`)
- **Question Encoder LoRA Configuration:** Rank 4, alpha 8, dropout 0.1, targeting query and value weights (`query` and `value`)

We conducted three complete training runs with different random seeds (21, 42, and 84) for each dataset to ensure robust evaluation and statistical significance. The learning rate was 2e-04. All models were initialised from the `rag-token-nq` checkpoint, demonstrating superior performance in our model selection experiments.

6.2.2 Evaluation Metrics

We assessed the performance using four key metrics:

- **Exact Match (EM):** Measures the percentage of generated answers that exactly match the reference answer after normalisation.
- **F1 Score:** Computes the token-level overlap between generated and reference answers, providing a more nuanced measure of answer correctness than EM.

- **Retrieval Precision@5:** Evaluates the percentage of questions for which at least one of the top 5 retrieved passages contains the answer.
- **Retrieval Precision@20:** Extends the retrieval evaluation to the top 20 passages, offering insight into the model’s ability to retrieve relevant information over a broader search space.

Statistical significance was determined through bootstrap testing with 1,000 samples and bias-corrected accelerated (BCa) 95% confidence intervals, as detailed in Chapter 3.

6.2.3 Results and Analysis

Table 6.1 presents the comprehensive results of our evaluation across all three datasets, comparing the parameter-efficient LoRA approach against the baseline `rag-token-nq` model:

Metric	COVID-19 QA		News QA		Conversation QA	
	Baseline	LoRA	Baseline	LoRA	Baseline	LoRA
Exact Match (%)	1.36	2.38*	4.45	7.27*	2.09	6.39*
F1 Score (%)	4.76	9.97*	8.39	14.18*	4.36	13.17*
Precision@5 (%)	47.59	52.26*	30.86	33.96*	14.69	18.31*
Precision@20 (%)	59.66	65.21*	44.25	47.67*	25.67	29.32*

* Statistically significant improvement

Table 6.1: Performance comparison between `rag-token-nq` and LoRA adaptation across three datasets. Values for LoRA represent the average across three seeds.

6.2.3.1 COVID-19 QA Results

On the COVID-19 QA dataset, our LoRA approach achieved significant improvements across all metrics:

- **Answer Generation:** LoRA improved the EM score from 1.36% to 2.38% (relative improvement of 75%) and F1 score from 4.76% to 9.97% (relative improvement of 109%).
- **Retrieval Quality:** We observed substantial gains in retrieval precision, with Precision@5 increasing from 47.59% to 52.26% (relative improvement of 9.8%) and Precision@20 improving from 59.66% to 65.21% (relative improvement of 9.3%).
- **Statistical Significance:** Bootstrap testing confirmed that all improvements were statistically significant, with 95% confidence intervals of (0.28%, 1.76%) for

EM, (4.21%, 6.24%) for F1 score, (2.55%, 7.14%) for Precision@5, and (3.40%, 7.65%) for Precision@20.

The substantial improvement in F1 score despite a more modest gain in EM suggests that our approach generates answers that are semantically closer to the reference answers even when they do not match exactly; this is particularly valuable in the biomedical domain, where terminology variations are common.

6.2.3.2 News QA Results

For the News QA dataset, LoRA adaptation demonstrated even more pronounced improvements in generation metrics:

- **Answer Generation:** EM score improved from 4.45% to 7.27% (relative improvement of 63%) and F1 score from 8.39% to 14.18% (relative improvement of 69%).
- **Retrieval Quality:** Retrieval precision also showed consistent gains, with Precision@5 increasing from 30.86% to 33.96% (relative improvement of 10%) and Precision@20 from 44.25% to 47.67% (relative improvement of 7.7%).
- **Statistical Significance:** All improvements were statistically significant, with 95% confidence intervals of (2.11%, 3.49%) for EM, (5.00%, 6.51%) for F1 score, (1.91%, 4.21%) for Precision@5, and (2.26%, 4.48%) for Precision@20.

The strong performance on News QA indicates that our parameter-efficient approach effectively adapts to news articles' different linguistic styles and content structures. These articles often contain more complex narratives and indirect answers than the more structured scientific content in COVID-19 QA.

6.2.3.3 Conversation QA Results

The Conversation QA dataset showed the most dramatic relative improvements:

- **Answer Generation:** LoRA adaptation increased the EM score from 2.09% to 6.39% (relative improvement of 206%) and F1 score from 4.36% to 13.17% (relative improvement of 202%).

- **Retrieval Quality:** Retrieval precision also improved significantly, with Precision@5 increasing from 14.69% to 18.31% (relative improvement of 24.6%) and Precision@20 from 25.67% to 29.32% (relative improvement of 14.2%).
- **Statistical Significance:** All improvements were statistically significant, with 95% confidence intervals of (3.50%, 5.18%) for EM, (7.86%, 9.82%) for F1 score, (2.23%, 4.97%) for Precision@5, and (2.24%, 5.33%) for Precision@20.

The substantial gains in Conversation QA are particularly noteworthy given the challenging nature of conversational content, which often contains informal language, references to previous turns, and context-dependent information. The significant improvement in both EM and F1 scores suggests that our parameter-efficient approach efficiently adapts to the conversational domain.

6.2.4 Cross-Dataset Comparison

Analysing the results across datasets reveals several interesting patterns:

- **Relative Improvement Pattern:** The relative improvement in generation metrics (EM and F1) is notably higher for Conversation QA (>200%) compared to COVID-19 QA (75-109%) and News QA (63-69%); this suggests that the baseline model struggled most with the conversational domain, providing more room for improvement through domain adaptation.
- **Retrieval vs. Generation Gains:** Across all datasets, the relative improvement in generation metrics (EM and F1) substantially exceeds the gains in retrieval precision; this indicates that our LoRA adaptation particularly enhances the generator’s ability to utilise retrieved information effectively, even when retrieval quality improves more modestly.
- **Domain-Specific Patterns:** The baseline performance varied significantly across domains (with News QA showing the highest baseline EM and F1 scores). Despite these differences, LoRA adaptation consistently delivered statistically significant improvements across all metrics and datasets, demonstrating its robustness for domain adaptation tasks.

6.2.5 Parameter Efficiency Analysis

A key aspect of our approach is parameter efficiency. Our LoRA adaptation modifies only approximately 737K parameters (0.12% of the total model parameters, including the

context encoder) yet delivers substantial performance improvements across all datasets. This extreme parameter efficiency has significant practical implications:

- **Memory Efficiency:** The memory footprint during training is dramatically reduced compared to full fine-tuning, allowing our approach to run on a single 6GB GPU while full fine-tuning of the original RAG model (as performed by Siriwardhana et al. [7]) required multiple high-memory GPUs.
- **Storage Efficiency:** The checkpoint size for each domain-adapted model is reduced from hundreds of megabytes to just a few megabytes, as only the LoRA adapters need to be stored.
- **Training Speed:** The reduced parameter count translates to faster training times, making domain adaptation more accessible for resource-constrained environments.

6.2.6 Comparative Analysis with Prior Work

We compare our relative improvements to prior approaches from Siriwardhana et al. [7] to contextualise our parameter-efficient approach. Tables 6.2 and 6.3 present this comparison (it is important to note that there are limitations to comparing retrieval metrics as we did not use golden passages), with three distinct adaptation approaches using `rag-token-nq` as the baseline:

1. **Our LoRA approach:** Parameter-efficient adaptation (0.1% trainable parameters) with frozen context encoder.
2. **RAG-original-QA:** Standard fine-tuning approach with question encoder adaptation but frozen context encoder.
3. **RAG-end2end-QA+R:** Full fine-tuning with question and context encoder adaptation and auxiliary reconstruction signal.

Dataset	EM			F1		
	LoRA	Orig	End2end	LoRA	Orig	End2end
COVID-19	1.75×	+2.95%*	+8.32%*	2.09×	2.54×	4.14×
News	1.63×	1.68×	3.25×	1.69×	1.80×	2.99×
Conv	3.06×	2.20×	4.73×	3.02×	2.16×	4.09×

Table 6.2: Performance improvements of adaptation methods. LoRA: 0.12% params, frozen context encoder; Orig: ~83% params, frozen context encoder; End2end: 100% params, trained context encoder. *Absolute improvement shown where the baseline was 0%.

Several key observations emerge from this three-way comparison:

Dataset	Prec@5			Prec@20		
	LoRA	Orig	End2end	LoRA	Orig	End2end
COVID-19	1.10×	1.16×	2.18×	1.09×	1.18×	1.99×
News	1.10×	1.17×	2.04×	1.08×	1.14×	1.68×
Conv	1.25×	1.87×	4.05×	1.14×	1.60×	2.93×

Table 6.3: Relative improvements (Precision metrics) of adaptation methods. LoRA: 0.12% params, frozen context encoder; Orig: $\sim 83\%$ params, frozen context encoder; End2end: 100% params, trained context encoder.

- **Parameter Efficiency Advantage:** Our LoRA approach achieves competitive improvements with only 0.12% of trainable parameters compared to standard fine-tuning (83% parameters) and full end-to-end fine-tuning (100% parameters); this demonstrates a highly favourable efficiency-performance trade-off.
- **Context Encoder Impact:** The most significant performance gap is between approaches with a frozen context encoder (LoRA and RAG-original) versus those with a trainable context encoder (RAG-end2end); this underscores that context encoder adaptation is crucial in maximising performance, particularly for retrieval precision.
- **LoRA vs. Standard Fine-tuning:** Despite using only 0.12% of trainable parameters, our LoRA approach achieves comparable and sometimes superior improvements to standard fine-tuning (RAG-original-QA). For the Conversation QA dataset, LoRA delivers substantially better results ($3.06\times$ vs. $2.20\times$ for EM), highlighting its effectiveness for significant domain shifts.
- **Domain-Specific Patterns:** The COVID-19 domain shows the most significant gap between approaches, with full fine-tuning achieving an $8.32\times$ improvement in EM compared to $2.95\times$ for standard fine-tuning and $1.75\times$ for LoRA; this reflects the specialised nature of biomedical literature, which benefits more from comprehensive adaptation, including the context encoder.

This comparison reveals important trade-offs in RAG domain adaptation. While full end-to-end fine-tuning with context encoder adaptation delivers the best absolute performance, our parameter-efficient LoRA approach offers competitive and sometimes superior improvements compared to standard fine-tuning, requiring dramatically fewer resources; this makes our approach particularly valuable for resource-constrained environments where full fine-tuning would be prohibitively expensive.

Chapter 7

Discussion and Conclusions

This chapter critically evaluates our research on parameter-efficient domain adaptation for Retrieval Augmented Generation (RAG) models. We first revisit our research goals and questions, discussing how we addressed them, followed by a thorough analysis of our findings. We summarise the key contributions and outline potential future work directions.

7.1 Discussion

Our primary goal was to reduce the computational overhead required for adapting RAG systems to new domains. Specifically, we aimed to address three research questions:

1. How can Parameter-Efficient Fine-Tuning (PEFT) methods be effectively integrated into retrieval-generation architectures like RAG?
2. Which existing PEFT methods realise the best performance gains regarding accuracy, F1-score, and top-k retrieval?
3. How can PEFT methods be applied to eliminate the need for Knowledge Base (KB) re-encoding when adapting RAG retrieval end-to-end?

Our evaluation results demonstrate significant progress toward answering these questions while revealing important insights about the relative efficacy of different parameter-efficient techniques for domain adaptation in hybrid retrieval-generation systems.

7.1.1 Integration of PEFT Methods with RAG

Regarding our first research question, we successfully integrated multiple PEFT methods into the RAG architecture. Our implementation accommodated both prompt-based methods (shallow prompt-tuning, deep prefix-tuning, and P-tuning v2) and low-rank adaptation (LoRA) for the generator and question encoder components.

The integration process revealed several technical challenges that we effectively addressed:

- We resolved the incompatibility between PEFT methods and the autoregressive generation process by implementing a custom forward method that handles None inputs gracefully during inference.
- We developed a specialised implementation of P-tuning v2 for the Dense Passage Retrieval (DPR) question encoder, adapting this PEFT technique from its original span prediction task to an embedding generation context.
- We integrated the PEFT-modified components seamlessly with the existing RAG pipeline, maintaining compatibility with established training and evaluation workflows.

These implementation details are important, as they provide a technical blueprint for applying parameter-efficient adaptation techniques to hybrid architectures combining retrieval and generation components. The success of this integration demonstrates that PEFT methods can effectively tune more complex architectures beyond single-component language models.

7.1.2 Comparative Performance of PEFT Methods

Our second research question sought to identify which PEFT methods deliver the best performance when adapting RAG models to new domains. Our findings in this area were particularly enlightening:

- LoRA consistently outperformed prompt-based methods across all datasets, with relative improvements of 54% in exact match performance compared to deep prompt tuning in our model selection experiments.
- Deep prompting techniques (prefix tuning, P-tuning v2) substantially outperformed shallow prompting, aligning with existing literature that suggests deeper modifications are necessary for smaller models.

- Both approaches demonstrated minimal parameter overhead, but LoRA achieved faster convergence and greater stability during training.

Given our resource constraints, Lora’s superior performance is particularly noteworthy. With only a 6GB GPU, LoRA adaptation achieved significant improvements over the baseline ($1.75\text{--}3.06\times$ for exact match and $1.69\text{--}3.02\times$ for F1 across datasets) while requiring only a fraction of the computational resources needed for full fine-tuning. This efficiency-performance trade-off represents a compelling case for adopting LoRA in resource-constrained environments.

The consistent improvement across all three datasets—COVID-19 QA, News QA, and Conversation QA—demonstrates the robustness of our approach to different domain characteristics. The robust results on Conversation QA ($3.06\times$ improvement in EM) suggest that our parameter-efficient approach is especially valuable for domains that differ substantially from the pre-training distribution.

7.1.3 Document Adaptation Without Re-encoding

Our third research question explored whether PEFT methods could eliminate the need for KB re-encoding during end-to-end adaptation; this represented the most ambitious and technically challenging aspect of our research, and our results were underwhelming:

- We successfully implemented a novel document adapter approach that enables adaptation of the passage representations without re-encoding the entire KB.
- Our approach introduced a trainable transformation between the original and domain-specific embedding spaces, combined with an efficient periodic re-indexing strategy.
- We evaluated three adapter architectures (shallow MLP, deep MLP, and transformer-based), finding that the transformer-based adapter provided the best performance.
- Despite these innovations, our document adaptation approach underperformed compared to the baseline in our experiments, indicating limitations.

The underperformance of our document adaptation approach, despite its theoretical soundness, suggests several potential limitations:

- The adapter’s capacity may be insufficient to model complex domain shifts effectively.

- The indirect optimisation signal from periodic re-indexing may impede effective learning.
- Adapting pre-computed embeddings rather than computing domain-specific embeddings from scratch may result in information loss.

Our exploration of alternative loss functions (KL divergence with Autoencoding-based Retriever Training) showed modest improvements but did not fully overcome these limitations; this suggests that while our approach offers significant computational advantages, further research is needed to achieve competitive performance without re-encoding.

7.1.4 Practical Implications

Considering our results holistically, several important practical implications emerge:

- Parameter-efficient approaches, particularly LoRA, offer a viable path to domain adaptation for organisations with limited computational resources.
- The trade-off between end-to-end fine-tuning (which achieved the best absolute performance in prior work) and our parameter-efficient approach presents a spectrum of options depending on available resources.
- The strong performance of LoRA, even without context encoder adaptation, suggests that we can realise much of the benefit of domain adaptation through efficient modification of the generator and question encoder alone.
- The challenges encountered with our document adaptation approach highlight the importance of context encoder adaptation for maximising performance, pointing to a need for more efficient approaches to this component.

These findings align with our original motivation of making RAG models more accessible and easier to customise for smaller players without substantial hardware investments. While complete end-to-end adaptation with context encoder training may remain the gold standard for performance, our LoRA approach offers a compelling alternative that balances performance and resource efficiency.

7.1.5 Limitations and Challenges

Several limitations and challenges affected the progress and scope of our work:

- **Hardware Constraints:** Our modest GPU (6GB) limited the batch sizes, model components, and validation set sizes we could use, necessitating compromises such as validation set sampling and limiting the number of retrieved documents.
- **Baseline Comparability:** Due to hardware constraints and limited documentation of hyperparameters in prior work, our absolute performance numbers may not be directly comparable to published results, though our relative improvements remain valid.
- **Document Adapter Limitations:** The underperformance of our document adaptation approach despite significant implementation effort points to fundamental challenges in adapting retrieval context without re-encoding.

Despite these limitations, our work significantly contributes to the field, particularly in demonstrating the viability of parameter-efficient adaptation for RAG models in resource-constrained environments.

7.2 Conclusion

This research has demonstrated that parameter-efficient fine-tuning methods can significantly reduce the computational resources required for domain adaptation of pre-trained RAG models. Our work bridges the gap between the growing interest in RAG systems for mitigating hallucinations in language models and the practical challenges of deploying these systems in resource-constrained environments.

Our primary conclusions are:

1. **Parameter-efficient domain adaptation is viable:** LoRA adaptation of the generator and question encoder components achieves substantial improvements over the baseline ($1.75\text{--}3.06\times$ for exact match and $1.69\text{--}3.02\times$ for F1 across datasets) while modifying only 0.12% of the model’s parameters; this makes domain adaptation accessible even to organisations with limited computational resources.
2. **LoRA outperforms prompt-based methods for RAG adaptation:** Among the parameter-efficient techniques evaluated, LoRA consistently delivered superior performance, faster convergence, and greater stability than shallow and deep prompting approaches; this aligns with recent findings in the literature and extends them to the hybrid retrieval-generation context of RAG.

3. **Context encoder adaptation remains challenging:** Our novel document adapter approach, while theoretically sound, underperformed the baseline in our experiments; this highlights the difficulty of adapting retrieval components without re-encoding and suggests that more research is needed.
4. **Domain characteristics influence adaptation gains:** The relative improvement from parameter-efficient adaptation varied across domains, with the most significant gains observed in Conversation QA ($3.06\times$ improvement in EM); this suggests that domains that differ substantially from the pre-training distribution benefit most from adaptation, even with parameter-efficient approaches.
5. **Resource-performance trade-offs are domain-dependent:** The comparison with prior work reveals that while complete end-to-end adaptation with context encoder training achieves the best absolute performance, the gap between our parameter-efficient approach and standard fine-tuning varies by domain. Our approach offers comparable or superior performance to standard fine-tuning for some domains despite using orders of magnitude fewer parameters.

Secondary conclusions include:

- The initialisation point significantly impacts adaptation performance, with the rag-token-nq checkpoint (pre-trained on question-answering tasks) outperforming the base RAG checkpoint.
- Deep prompting techniques are necessary to effectively adapt moderate-sized models, with shallow prompting showing minimal improvement in our experiments.
- Attention mechanisms are crucial for document adaptation, as evidenced by the superior performance of our transformer-based adapter compared to MLP variants.
- Document transformations and periodic re-indexing provide a computationally efficient alternative to continuous re-encoding, though further research is needed to make such an approach viable.

Together, these findings advance the state of the art in parameter-efficient domain adaptation for retrieval-augmented language models, providing theoretical insights and practical approaches for deploying these systems in resource-constrained environments.

7.3 Future Work

Our research opens several promising avenues for future investigation:

7.3.1 Improved Document Adaptation Techniques

While our document adapter approach did not outperform the baseline, adapting retrieval components without re-encoding remains valuable. Future work could explore:

- **More expressive adapter architectures:** Deeper transformer-based adapters or alternative architectures that better capture domain-specific transformations of the embedding space.
- **Contrastive learning objectives:** Loss functions that optimise for similarity between related query-document pairs in the domain-specific space.
- **Joint query-document adaptation:** Approaches that ensure alignment between query and document representations through shared parameters or regularisation techniques.
- **Hybrid re-encoding approaches:** Methods that combine partial re-encoding of the most relevant documents with the adaptation of the remainder, potentially offering a better balance between performance and efficiency.

7.3.2 Technical Enhancements

Several technical improvements could enhance the practicality and performance of our approach:

- **Optimisation of adapter inference:** Techniques for reducing the latency overhead of adapters through weight merging, quantisation, or other optimisations.
- **Automated adapter design:** Methods for automatically determining the optimal adapter architecture and hyperparameters for a given domain adaptation task.
- **Hybrid CPU-GPU processing:** Approaches that leverage CPU resources for retrieval and indexing tasks while reserving GPU resources for model adaptation, potentially enabling more efficient training with limited hardware.
- **Distributed training support:** Extensions to enable efficient distributed training of parameter-efficient RAG models across multiple lower-end GPUs rather than requiring high-memory devices.

By addressing these future directions, researchers can build upon our work to further enhance the accessibility, efficiency, and performance of RAG models for domain-specific applications; this will ultimately make these robust systems more widely available to organisations with limited computational resources.

Bibliography

- [1] A. Radford and K. Narasimhan, “Improving language understanding by generative pre-training,” 2018.
- [2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. 2019 Conf. North American Chapter Assoc. Comput. Linguistics: Human Language Technologies*, vol. 1. Assoc. for Comput. Linguistics, Jun. 2019, pp. 4171–4186.
- [3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 1–67, Jan. 2020.
- [4] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, “Survey of hallucination in natural language generation,” *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023.
- [5] A. Hadi, E. Tran, B. Nagarajan, and A. Kirpalani, “Evaluation of chatgpt as a diagnostic tool for medical learners and clinicians,” *PLOS ONE*, vol. 19, no. 7, pp. 1–20, Jul. 2024.
- [6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” in *Proc. 34th Int. Conf. Neural Information Processing Systems*, ser. NIPS ’20. Curran Associates Inc., 2020.
- [7] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, “Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering,” *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1–17, Jan. 2023.
- [8] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics and 11th Int. Joint*

-
- Conf. Natural Language Processing.* Assoc. for Comput. Linguistics, Aug. 2021, pp. 4582–4597.
- [9] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proc. 2021 Conf. Empirical Methods Natural Language Processing.* Assoc. for Comput. Linguistics, Nov. 2021, pp. 3045–3059.
 - [10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” in *Proc. Int. Conf. Learning Representations*, 2022.
 - [11] I. Bousquette, “Ai doesn’t know much about golf. or farming. or mortgages. or ...” *The Wall Street J.*, Oct. 2024, article discusses limitations of AI models in specific domain knowledge.
 - [12] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. t. Yih, “Dense passage retrieval for open-domain question answering,” in *Proc. 2020 Conf. Empirical Methods Natural Language Processing.* Assoc. for Comput. Linguistics, Nov. 2020, pp. 6769–6781.
 - [13] R. K. Luu and M. J. Buehler, “Bioinspiredllm: Conversational large language model for the mechanics of biological and bio-inspired materials,” *Adv. Sci.*, vol. 11, no. 10, p. 2306724, 2024.
 - [14] Y. Guo, W. Qiu, G. Leroy, S. Wang, and T. Cohen, “Retrieval augmentation of large language models for lay language generation,” *J. Biomed. Informatics*, vol. 149, p. 104580, 2024.
 - [15] S. Kresevic, M. Giuffrè, M. Ajcevic, A. Accardo, L. S. Crocè, and D. L. Shung, “Optimization of hepatological clinical guidelines interpretation by large language models: A retrieval augmented generation-based framework,” *npj Digit. Med.*, vol. 7, no. 1, p. 102, 2024.
 - [16] A. Ficek, J. Zeng, and O. Kuchaiev, “GPT vs RETRO: Exploring the intersection of retrieval and parameter-efficient fine-tuning,” in *Proc. 2024 Conf. Empirical Methods Natural Language Processing.* Assoc. for Comput. Linguistics, Nov. 2024, pp. 19 425–19 432.
 - [17] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, “Realm: Retrieval-augmented language model pre-training,” in *Proc. 37th Int. Conf. Machine Learning*, ser. ICML’20. JMLR.org, 2020.
 - [18] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. V. D. Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. D. L. Casas, A. Guy,

- J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, and L. Sifre, “Improving language models by retrieving from trillions of tokens,” in *Proc. 39th Int. Conf. Machine Learning*, vol. 162. PMLR, Jul. 2022, pp. 2206–2240.
- [19] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, “The faiss library,” 2024.
- [20] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*. Assoc. for Comput. Linguistics, Jul. 2020, pp. 7871–7880.
- [21] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.
- [22] B. J. Chan, C.-T. Chen, J.-H. Cheng, and H.-H. Huang, “Don’t do rag: When cache-augmented generation is all you need for knowledge tasks,” in *Proc. Web Conference 2025 (WWW ’25)*, 2025, p. 5.
- [23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. t. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” in *Proc. Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474.
- [24] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, “Query rewriting in retrieval-augmented large language models,” in *Proc. 2023 Conf. Empirical Methods Natural Language Processing*. Assoc. for Comput. Linguistics, Dec. 2023, pp. 5303–5315.
- [25] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, “Knowledge graph prompting for multi-document question answering,” in *Proc. Thirty-Eighth AAAI Conf. Artificial Intelligence*. AAAI Press, 2024, p. 2141.
- [26] K. Sawarkar, A. Mangal, and S. R. Solanki, “Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers,” in *2024 IEEE 7th Int. Conf. Multimedia Information Processing and Retrieval*, 2024, pp. 155–161.
- [27] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlga, A. Shashua, K. Leyton-Brown, and Y. Shoham, “In-context retrieval-augmented language models,” *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1316–1331, Nov. 2023.

-
- [28] Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig, “Learning to filter context for retrieval-augmented generation,” 2023.
 - [29] H. Yang, Z. Li, Y. Zhang, J. Wang, N. Cheng, M. Li, and J. Xiao, “Prca: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter,” in *Proc. 2023 Conf. Empirical Methods Natural Language Processing*. Assoc. for Comput. Linguistics, Dec. 2023, pp. 5364–5375.
 - [30] Z. Yu, C. Xiong, S. Yu, and Z. Liu, “Augmentation-adapted retriever improves generalization of language models as generic plug-in,” in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*. Assoc. for Comput. Linguistics, Jul. 2023, pp. 2421–2436.
 - [31] Z. Ke, W. Kong, C. Li, M. Zhang, Q. Mei, and M. Bendersky, “Bridging the preference gap between retrievers and llms,” in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics*. Assoc. for Comput. Linguistics, Aug. 2024, pp. 10 438–10 451.
 - [32] J. Huang, L. Cui, A. Wang, C. Yang, X. Liao, L. Song, J. Yao, and J. Su, “Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal,” in *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 1. Assoc. for Comput. Linguistics, Aug. 2024, pp. 1416–1428.
 - [33] Z. Hu, C. Wang, Y. Shu, H.-Y. Paik, and L. Zhu, “Prompt perturbation in retrieval-augmented generation based large language models,” in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining*. New York, NY, USA: Assoc. for Comput. Machinery, 2024, pp. 1119–1130.
 - [34] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. D. Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for nlp,” in *Proc. 36th Int. Conf. Machine Learning*, ser. Proc. Machine Learning Research, vol. 97. PMLR, Jun. 2019, pp. 2790–2799.
 - [35] M. Q. Pham, J. M. Crego, F. Yvon, and J. Senellart, “A study of residual adapters for multi-domain neural machine translation,” in *Proc. Fifth Conf. Machine Translation*. Online: Assoc. for Comput. Linguistics, Nov. 2020, pp. 617–628.
 - [36] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, “Mad-x: An adapter-based framework for multi-task cross-lingual transfer,” in *Proc. 2020 Conf. Empirical Methods Natural Language Processing*. Assoc. for Comput. Linguistics, Nov. 2020, pp. 7654–7673.
 - [37] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Nee-lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger,

- T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Proc. Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.
- [38] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proc. 2021 Conf. Empirical Methods Natural Language Processing*. Assoc. for Comput. Linguistics, Nov. 2021, pp. 3045–3059.
- [39] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, “Gpt understands, too,” *AI Open*, vol. 5, pp. 208–215, 2024.
- [40] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, “P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks,” in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics (Volume 2: Short Papers)*. Assoc. for Comput. Linguistics, May 2022, pp. 61–68.
- [41] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, “Natural questions: A benchmark for question answering research,” *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 453–466, Aug. 2019.
- [42] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension,” in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics (Volume 1: Long Papers)*. Assoc. for Comput. Linguistics, Jul. 2017, pp. 1601–1611.
- [43] T. Kočisk’y, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, “The NarrativeQA reading comprehension challenge,” *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 317–328, 2018.
- [44] P. Dasigi, K. Lo, I. Beltagy, A. Cohan, N. A. Smith, and M. Gardner, “A dataset of information-seeking questions and answers anchored in research papers,” in *Proc. 2021 Conf. North American Chapter Assoc. Comput. Linguistics: Human Language Technologies*. Assoc. for Comput. Linguistics, Jun. 2021, pp. 4599–4610.
- [45] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, and S. Bowman, “QuALITY: Question answering with long input texts, yes!” in *Proc. 2022 Conf. North American Chapter Assoc. Comput. Linguistics: Human Language Technologies*. Assoc. for Comput. Linguistics, Jul. 2022, pp. 5336–5358.

-
- [46] M. Zhong, D. Yin, T. Yu, A. Zaidi, M. Mutuma, R. Jha, A. H. Awadallah, A. Celikyilmaz, Y. Liu, X. Qiu, and D. Radev, “QMSum: A new benchmark for query-based multi-domain meeting summarization,” in *Proc. 2021 Conf. North American Chapter Assoc. Comput. Linguistics: Human Language Technologies*. Assoc. for Comput. Linguistics, Jun. 2021, pp. 5905–5921.
 - [47] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017, pp. 2962–2971.
 - [48] Z. Li, Z. Zhong, P. Zuo, and H. Zhao, “A personalized federated learning method based on the residual multi-head attention mechanism,” *J. King Saud Univ. - Comput. and Inf. Sci.*, vol. 36, no. 4, p. 102043, 2024.
 - [49] R. Dror, G. Baumer, S. Shlomov, and R. Reichart, “The hitchhiker’s guide to testing statistical significance in natural language processing,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Volume 1: Long Papers)*. Assoc. for Comput. Linguistics, Jul. 2018, pp. 1383–1392.
 - [50] T. Berg-Kirkpatrick, D. Burkett, and D. Klein, “An empirical investigation of statistical significance in NLP,” in *Proc. 2012 Joint Conf. Empirical Methods Natural Language Processing and Computational Natural Language Learning*. Assoc. for Comput. Linguistics, Jul. 2012, pp. 995–1005.
 - [51] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*, ser. Chapman Hall/CRC monographs on statistics and applied probability. London: Chapman and Hall, 1993.
 - [52] Y. Bestgen, “Please, don’t forget the difference and the confidence interval when seeking for the state-of-the-art status,” in *Proc. Thirteenth Language Resources and Evaluation Conf.* European Language Resources Association, Jun. 2022, pp. 5956–5962.
 - [53] D. S. Sachan, M. Lewis, D. Yogatama, L. Zettlemoyer, J. Pineau, and M. Zaheer, “Questions are all you need to train a dense passage retriever,” *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 600–616, 2023.

Appendix A

Custom P-tuning v2 Implementation Details

This appendix details our custom P-tuning v2 implementation for Dense Passage Retrieval (DPR) question encoders.

A.1 Adaptation of P-tuning v2 for DPR Question Encoders

Adapting the reference implementation to work with DPR’s BERT-based architecture required addressing several fundamental architectural differences:

- **Shift from Span Prediction to Embedding Generation:** Unlike the reference implementation, which focuses on question-answering architectures that predict start/end token positions (like `BertForQuestionAnswering` and `RobertaPrefixModelForQuestionAnswering`), DPR requires generating embeddings for similarity matching. This fundamental shift required restructuring how the model processes and outputs information.
- **Bi-encoder Architecture Support:** DPR’s bi-encoder design, with separate question and passage encoders, required carefully applying prefix tuning to the question encoder while maintaining compatibility with the passage encoder; this contrasts with the reference implementation’s assumption of a single encoder for direct answer extraction.

- **Output Transformation:** We modified the output handling to focus on the pooled representation (typically from the [CLS] token) rather than the token-level outputs used in span-based QA models, ensuring compatibility with DPR’s similarity-based retrieval mechanism.
- **Embedding Projection Management:** We implemented optional projection dimension logic specifically for DPR’s embedding requirements, exposing the `embeddings_size` property to maintain compatibility with the retrieval pipeline’s dimensional expectations.

A.2 Key Components of the Implementation

Our implementation consists of three primary classes that work together to enable P-tuning v2 for DPR’s BERT-based architecture:

1. **PrefixEncoder:** This foundational class encodes prefix tokens into continuous prompts, adapted from the reference implementation but with enhanced dimensional handling for DPR. Unlike the reference implementation, our version includes comprehensive type hinting and more flexible configuration options, particularly in how it transforms embeddings through a two-layer MLP when `prefix_projection` is enabled.
2. **DPRPrefixEncoder:** This class extends BERT’s functionality specifically for DPR models, transforming the span prediction approach of the reference implementation to an embedding generation approach. It generates past key values for the attention mechanism while ensuring that only the prefix parameters are trainable, preserving the parameter-efficient nature of the technique.
3. **DPRPrefixQuestionEncoder:** The top-level class that fundamentally reorients the model from span prediction to embedding generation. It manages the forward pass to produce embeddings suitable for similarity matching rather than token classification, effectively bridging the architectural gap between standard QA and retrieval models.

A.3 Technical Challenges and Solutions

Converting the reference implementation to work with DPR’s retrieval-oriented architecture presented several technical challenges:

- **Attention Mask Extension:** DPR’s attention mechanism needed modification to properly account for prefix tokens. Our solution in the `forward` method explicitly prepends a prefix attention mask to the input mask, ensuring proper attention flow between prefix tokens and input tokens—a critical adaptation not addressed in the reference implementation.
- **Input Validation and Error Handling:** We implemented robust input validation to prevent runtime errors during prefix generation, particularly when handling edge cases like `None` inputs or absent embeddings; this contrasts with the reference implementation’s more basic error handling.
- **Pooler Output Management:** For retrieval to function correctly, we needed to carefully manage the pooler output (embedding from the [CLS] token) and implement an optional projection layer that properly transforms these embeddings to the expected dimensionality for the retrieval system—a component not needed in the span prediction context of the reference implementation.

A.4 Divergence from Reference Implementation

While our implementation builds upon the core concept of the reference P-tuning v2 code, several aspects represent significant architectural adaptations necessary for the retrieval context:

- **Output Structure Transformation:** The most fundamental change was reorienting the model from producing start/end logits for answer spans to generating dense embeddings for similarity search; this required a complete restructuring of the forward pass outputs and loss functions.
- **Embedding Dimensionality Control:** We added precise control over embedding dimensions through the `projection_dim` parameter and associated projection layer, ensuring compatibility with the embedding dimensions expected by the retrieval pipeline.
- **Forward Pass Refactoring:** While the reference implementation’s forward pass focuses on producing and processing token-level predictions, our implementation refactors this to emphasize the pooler output processing required for embedding-based retrieval.

Appendix B

Model Selection Experiments

This appendix provides detailed experimental results from our preliminary model selection experiments conducted on the Conversation QA dataset. Each validation step in the training log excerpts occurs every third of an epoch.

B.1 RAG-token-base versus RAG-token-nq Initialisations

Figure B.1 summarises training logs for both initialisations over 30 validation steps (on our 500-example stratified subset) with a learning rate of $5e-5$.

Base Checkpoint Logs (Selected Steps):

step	val_avg_em	val_avg_loss
1	0.0	54.21
2	0.0	49.55
3	0.0	49.35
5	0.0	47.84
... plateau at EM \sim 0.0		

NQ Checkpoint Logs (Selected Steps):

step	val_avg_em	val_avg_loss
1	0.016	61.19
6	0.046	52.86
11	0.050	51.23
20	0.052	50.19
27	0.056	50.02
30	0.054	49.90

Figure B.1: Training logs for **rag-token-base** (left) and **rag-token-nq** (right) on ConvQA. The NQ-initialized model achieves a modest but consistent improvement in EM by step 30, whereas the base checkpoint appears unable to exceed 0% EM.

With the **rag-token-base** checkpoint, early validation steps show $\text{val_avg_em} = 0.0$, indicating zero exact matches for the entire validation subset. Even after several epochs, the base model fails to exceed a near-zero exact match score.

In contrast, the **rag-token-nq** initialisation rapidly improves on the same data, achieving an exact match (EM) of around 0.05–0.06 (5–6%) by step 30. While the absolute

EM remains modest—likely due to limited fine-tuning steps and the complexity of conversational QA—this outperforms the base checkpoint, which appears to plateau near 0%.

B.2 Shallow versus Deep Prompting Techniques

We compared the deep prompting `rag-token-nq` run from the previous experiment with a shallow prompted alternative: P-tuning v1 applied to both the generator and question encoder components, with trainable continuous prompt vectors added only at the input embedding layer. The learning rate we used was 1e-05 (the most common value Liu et al. [39] used in their experiments).

Figure B.2 summarises the training logs for both prompting techniques.

Deep Prompt Tuning Logs (Selected):			Shallow Prompt Tuning Logs (Selected):		
step	val_avg_em	val_avg_loss	step	val_avg_em	val_avg_loss
1	0.016	61.19	1	0.012	86.22
6	0.046	52.86	2	0.010	84.31
11	0.050	51.23	3	0.010	84.01
20	0.052	50.19	4	0.012	85.84
27	0.056	50.02	5	0.008	84.90
30	0.054	49.90	6	0.010	87.35
			7	0.008	85.55

Figure B.2: Training logs for deep prompt tuning (left) and shallow prompt tuning (right) on ConvQA. Deep prompting shows consistent improvement and lower loss values, while shallow prompting struggles with higher loss and negligible EM improvement.

The key observations from this comparison include:

- **Superior EM Performance:** Deep prompt tuning achieves substantially higher exact match scores (peaking at 5.6%) compared to shallow prompt tuning (remaining at or below 1.2%).
- **Lower Loss Values:** The validation loss for deep prompting starts lower (61 vs. 86) and decreases consistently, eventually reaching 50. In contrast, shallow prompting exhibits higher loss values that fluctuate between 84-87 without showing a clear downward trend.
- **Stability Differences:** Deep prompting shows a steadier improvement pattern with consistent gradual gains, while shallow prompting appears unstable with fluctuating metrics that show no clear improvement trend.

- **Convergence Behavior:** The deep prompting approach converges toward a better solution, stabilising loss values around the 30-step mark. The shallow approach shows no signs of convergence within the available training steps.

B.3 Assessment of LoRA adaptations

We applied LoRA to both the generator and question encoder components using the following configurations:

The generator was adapted using a LoRA configuration targeting query and value projection matrices:

```
generator_lora_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM,
    r=4,
    lora_alpha=16,
    lora_dropout=0.1,
    target_modules=["q_proj", "v_proj"],
    bias="none",
)
```

Similarly, we adapt the question encoder with a slightly different configuration appropriate for its feature extraction role:

```
qenc_lora_config = LoraConfig(
    task_type=TaskType.FEATURE_EXTRACTION,
    r=4,
    lora_alpha=8,
    lora_dropout=0.1,
    target_modules=["query", "value"],
    bias="none",
)
```

Figure B.3 displays selected validation steps from our LoRA adaptation experiments.

Key observations from the LoRA experiments include:

- **Superior Performance:** LoRA adaptation achieves significantly higher exact match scores (peaking at 8.6%) compared to deep prompt tuning (peaking at

LoRA Adaptation Logs (Selected):

step	val_avg_em	val_avg_loss
2	0.044	53.48
4	0.058	51.12
6	0.074	50.23
8	0.066	49.94
13	0.078	49.03
17	0.080	48.85
22	0.086	48.84
27	0.084	48.87
30	0.086	48.91

Deep Prompt Tuning Logs (Reference):

step	val_avg_em	val_avg_loss
1	0.016	61.19
6	0.046	52.86
11	0.050	51.23
20	0.052	50.19
27	0.056	50.02
30	0.054	49.90

Figure B.3: Training logs for LoRA adaptation (left) and deep prompt tuning (right) on ConvQA. LoRA achieves substantially higher exact match scores and lower loss values, with consistent improvement throughout training.

5.6%); this represents a relative improvement of approximately 54% in exact match performance.

- **Faster Convergence:** LoRA shows more rapid improvement in early training steps, reaching 5.8% EM by step 4, whereas deep prompting is required until step 20 to reach a comparable level.
- **Lower Loss Trajectory:** Both approaches show similar patterns of loss reduction, but LoRA consistently achieves lower loss values at comparable training steps.
- **Continued Improvement:** LoRA shows steady improvement throughout training, with performance increasing even in later steps, suggesting that longer training might yield further gains.
- **Stability:** LoRA maintains stable performance with minimal fluctuation in later training steps, with EM scores consistently remaining above 8% from step 17 onward.

B.4 Exploration of Increasing Adapter Complexity

Having established LoRA as our primary adaptation method for the generator and question encoder, we next investigated the impact of adapter architecture complexity on our novel document adaptation approach. We used $1e-04$ as the learning rate for our adapter training based on empirical evidence from adapter-based parameter-efficient approaches in the literature. We re-indexed every 750 steps.

Figure B.4 presents validation logs for the three adapter architectures over multiple validation steps.

Standard MLP Adapter:

step	val_avg_em	val_avg_loss
1	0.016	89.83
2	0.012	90.88
4	0.016	91.16
6	0.010	89.79
8	0.014	90.00
10	0.008	90.09
12	0.010	90.24

Deeper MLP Adapter:

step	val_avg_em	val_avg_loss
1	0.016	89.38
2	0.012	89.13
3	0.006	92.01
4	0.004	91.55
5	0.002	91.35

Transformer Adapter:

step	val_avg_em	val_avg_loss
1	0.040	90.34
2	0.040	90.37
4	0.038	90.60
8	0.034	90.36
12	0.034	90.05
16	0.032	89.75
20	0.034	89.68

Figure B.4: Training logs for Standard MLP (top left), Deeper MLP (top right), and Transformer-based (bottom) document adapters on ConvQA. The transformer-based adapter demonstrates consistently higher EM scores and stable training dynamics, though all approaches fall short of the baseline performance.

The key observations from this comparison include:

- **Superior Performance of Transformer-based Adapter:** The transformer-based adapter achieves substantially higher exact match scores (peaking at 4.0% and consistently maintaining above 3.2%) compared to both MLP variants; this represents a relative improvement of approximately 150-300% over the standard MLP adapter, which fluctuates between 0.8-1.6% EM.
- **Diminishing Returns from Depth Alone:** Surprisingly, the deeper MLP adapter performs worse than the standard MLP, with EM scores declining to as low as 0.2% by step 5; this suggests that simply adding depth without introducing more sophisticated transformation mechanisms may hinder rather than help adaptation.
- **Stability Differences:** The transformer-based adapter exhibits remarkably stable training dynamics, consistently maintaining similar EM performance throughout training. In contrast, the MLP-based adapters show erratic performance with substantial fluctuations in both EM scores and loss values.
- **Limited Loss Improvement:** Despite significant differences in EM performance between adapter architectures, all three variants show relatively minor changes in loss values. The transformer adapter’s loss decreases marginally from 90.34

to 89.68 over 20 steps, suggesting that the loss function may not fully capture improved retrieval quality.

- **Performance Gap vs. Baseline:** When compared to the validation EM of 4.40% achieved without any document adapter (using just the `rag-token-nq` checkpoint), even our best transformer-based adapter (4.0% EM) falls short; this represents an unexpected finding—our document adaptation approach degrades performance rather than improving it.

B.5 Assessment of KL Divergence Loss

We used the standard MLP adapter architecture for this experiment with our novel document adaptation approach. We compared two training setups:

- **Standard Loss:** The base configuration using cross-entropy loss for generation with the standard RAG training objective.
- **KL Divergence Loss:** The ART approach with KL divergence between the “ideal” distribution (based on reconstruction probabilities) and similarity-based retrieval distribution.

The KL divergence implementation used a temperature scaling parameter of 0.1 to control the sharpness of the probability distributions. Figure B.5 displays selected validation steps from both training approaches.

Standard Loss (Selected Steps):

step	val_avg_em	val_avg_loss
1	0.016	89.83
2	0.012	90.88
4	0.016	91.16
6	0.010	89.79
8	0.014	90.00
10	0.008	90.09
12	0.010	90.24

KL Divergence Loss (Selected Steps):

step	val_avg_em	val_avg_loss
1	0.018	90.01
2	0.008	90.25
3	0.010	91.08
4	0.024	90.53
5	0.008	91.14
6	0.010	89.69
7	0.016	90.93

Figure B.5: Training logs for standard loss (left) and KL divergence loss (right) with the standard MLP document adapter on ConvQA. The KL divergence approach shows a slightly higher peak EM score but similar overall performance patterns.

Key observations from this comparison include:

- **Modest Improvement in Peak Performance:** The KL divergence approach achieves a slightly higher peak exact match score (2.4% at step 4) compared to the

standard loss approach (1% at step 6). However, this improvement is relatively small and may not be statistically significant, given the limited number of training steps.

- **Similar Loss Trajectories:** Both approaches show comparable validation loss patterns, fluctuating within similar ranges (89.69–91.14 for KL divergence and 89.79–91.16 for standard loss) without a clear downward trend.
- **Continued Performance Gap:** Even with the KL divergence loss, the document adaptation approach still falls significantly short of the 4.40% EM baseline achieved without any document adapter, indicating that the loss function alone does not address the fundamental limitations identified in our adapter architecture experiments.
- **Performance Instability:** Both approaches exhibit fluctuating EM scores across training steps, suggesting neither method provides stable, consistent improvement during the training process.