Low-Rank Transformations of the Syntactic-Semantic Space in Multiple Natural Language Tasks

Yves Emmanuel[1], **Silvio Melo**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
50732-970 – Recife – PE – Brasil

{yefo,sbm}@cin.ufpe.br

Abstract. *This project evaluates the LoRA-multi approach for parameter-efficient fine-tuning of Transformer models across various natural language processing tasks. By applying Low-Rank Adaptation (LoRA) to BERT and DistilBERT models and combining them with a Mixture-of-Expert strategy on the X-LoRA framework, we assess their performance on established benchmarks, including SWAG, SQuAD, and WNUT17. Our experimental results validate that while full fine-tuning yields higher performance on specific tasks, the use of LoRA significantly reduces the number of trainable parameters, providing a practical balance between model performance and computational efficiency. This is particularly beneficial for multi-task learning scenarios, where minimizing resource consumption is crucial. Additionally, we explore the implications of these findings for deploying specialized models in real-world applications, highlighting the X-LoRA framework's capability to maintain versatility while adapting to various downstream tasks. Ultimately, our work aims to advance the understanding of how low-rank adaptations can enhance the efficiency of Transformer models, paving the way to understanding how these models encode syntactic-semantic information.*

1. Introduction

Transformer-based generative language models have been widely used in the industry to address problems in various fields, such as structured query language (SQL) generation from informal text [Qin et al. 2022], solving mathematical problems [Liu et al. 2023] and legal consulting [Cui et al. 2023]. Companies adapt these models for specific tasks through different approaches: Prompt Engineering - techniques such as Few-shot Prompting [Brown 2020] and Chain-of-Thought Prompting [Wei et al. 2022] aim to improve the quality of responses by enabling in-context learning [Work 2022] providing task examples or the step-by-step reasoning for the task; Retrieval-Augmented Generation (RAG) [Lewis et al. 2020] - the process of optimizing the model's response about unseen data introducing context with similar data retrieval, which offers a good balance of performance and efficiency; and Fine-Tuning [Howard and Ruder 2018] - specializing the model in downstream tasks with subtle adjustments to its pre-trained parameters, enabling transfer-learning [Zhuang et al. 2019].

Prompt engineering provides clear instructions to the model but performs poorly compared to fine-tuning [Liu et al. 2022]. While this approach is practical for simple tasks, it needs outcome control for more complex tasks. In parameter fine-tuning, the model

is trained as a specialist in a task with precise control over the outcome requiring computational resources and data. RAG combines the generative power of language models with the ability to retrieve relevant information from external sources, providing accurate responses with less data needed than fine-tuning. Inference latency is a limitation of RAG architectures, particularly as the input size increases linearly with the number of retrieved documents, as noted by [Zhu et al. 2024].

The fine-tuning approach excels in scenarios demanding high performance across multiple tasks with minimal latency [Zhao et al. 2024]. State-of-the-art models like GPT-4 [Achiam et al. 2023], PaLM2 [Anil et al. 2023], Claude 3.5 [Anthropic 2024], Llama3 [Touvron et al. 2023] and Gemini 1.5 [Reid et al. 2024] are defined with billions of parameters and large-scale, domain-diverse training data to facilitate effective transfer learning across multiple knowledge domains. Consequently, fine-tuning offers a compelling balance of high performance in various tasks with no additional inference latency, as the training datasets complement the pre-trained knowledge.

Full fine-tuning large language models for multiple tasks is unfeasible due to the large dimensionality of its parameters. Researchers have proposed parameter-efficient fine-tuning (PEFT) optimizations to reduce the computational resources associated with the number of parameters to be adapted. Adapter layers [Houlsby et al. 2019] reduce the number of parameters to be adapted but add inference latency with sequential processing. Low-Rank Adaptation (LoRA) [Hu et al. 2021] is an innovative approach based on the proposal by [Aghajanyan et al. 2020] that overparameterized language models have low intrinsic dimensions. Thus, it conjectures that adjustments also have low intrinsic dimensions, which can be decomposed into low-rank matrices, drastically reducing the number of parameters to be adapted. In many tasks, adapting 1% of the parameters can achieve 90% of the full fine-tuning performance.

The framework "Meteor-of-LoRA" (MeteoRA), based on a Mixture of Experts (MoE) architecture, is proposed by [Xu et al. 2024] as a method to articulate LoRA adapters for multiple tasks. This framework keeps the low inference overhead while regulating the contribution of different LoRA adapters with a gating network and provides a feasible way to adapt models for multiple tasks. X-LoRA [Buehler and Buehler 2024] is a MoE strategy that introduces a scaling mechanism for LoRA adapters via a feed-forward neural network.

This project presents an experimental study aimed at transforming and understanding the syntactic-semantic space of Transformer-based models across multiple tasks. Using parameter-efficient fine-tuning techniques to adapt the model's intrinsic dimensions, we significantly reduce computational costs, performing these adaptations with a single GPU. Through these efficient strategies, we provide a comprehensive analysis of how modern language models represent information and adapt to the complexity of diverse tasks with minimal overhead.

2. Related Work

Transformer-based Language Model. The Transformer architecture utilizes multiple layers of encoders and decoders with multi-head attention mechanisms that simultaneously focus on different parts of the input sequence, capturing complex relationships between word embeddings and understanding extended contexts and distant dependencies

[Vaswani 2017].

The Transformer block illustrated in Figure 1 consists of two fundamental modules: the attention mechanism and the feedforward neural network. In the multi-head attention mechanism, the key components are the query (Q), key (K), and value (V) matrices. In section 3.2 we will explore how the query, key, and value matrices relate while adapting Transformer weights to downstream tasks. Given an input sequence $\vec{x} \in \mathbb{R}^{n \times d}$, each head in the multi-head attention mechanism has its own set of weights:

$$Q_i = \vec{x}W_i^Q, \quad K_i = \vec{x}W_i^K, \quad V_i = \vec{x}W_i^V \quad \text{for } i = 1, \ldots, h$$

$W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ are the weight matrices for the i^{th} attention head.

During pre-training, the Transformer learns vector representations of words, mapping them to a high-dimensional space [Mikolov 2013]. The attention mechanism calculates the similarity between words using the inner products of these vectors, organizing them into matrices that indicate the strength of semantic, syntactic relationships, and coreference [Clark 2019]. The scaled dot-product attention for a single head is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where d_k is the dimension of the key matrix.

The outputs of the attention heads are concatenated and linearly transformed:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ is the weight matrix for the output linear transformation to aggregate multi-head outputs. Layer normalization is applied before and after the attention mechanism is performed to stabilize the training process and boost model convergence by addressing the *internal covariate shift problem* (ICS) [Zhang and Sennrich 2019]. ICS arises when the distribution of layer activations changes during training.

The feedforward neural network (FNN) module plays a critical role in encoding information into high-dimensional vectors, enabling language models to efficiently store facts [Nanda et al. 2023]. Two key concepts for understanding this process are *polysemanticity* and *distributed representations*. Polysemanticity refers to individual neurons encoding multiple features, activating in a range of contexts [Cunningham et al. 2023], while distributed representations involve many neurons firing for the same feature [Geiger et al. 2024]. Features are properties of the input; for example, in a language model, the input sequence *"LeBron James is the GOAT"* might have properties such as athlete, basketball, greatness, and opinion, each of which is encoded into specific vector directions. The *superposition* hypothesis [Elhage et al. 2022] provides a mechanistic explanation, proposing that features are stored as nearly orthogonal vectors in the activation space (see Equation 2). Although this enables efficient encoding, it can introduce slight interference between features, adding a degree of noise.

This behavior is closely related to the Johnson–Lindenstrauss lemma [Nunes and Antunes 2018], which asserts that the number of nearly orthogonal features that can be packed into a space grows exponentially with the number of dimensions. For large language models, this means that independent concepts can be associated with nearly orthogonal directions in a high-dimensional space [Engels et al. 2024], enabling efficient and complex representation of language.

The feedforward neural network in Transformers consists of linear transformations applied to the input sequence. Between these transformations, an activation function captures the complex relationships between words. This process can be represented as:

$$\text{FNN}(\vec{x}) = \text{Activation}(\vec{x}W^{up\_proj} + b_{up\_proj})W^{down\_proj} + b_{down\_proj} \tag{2}$$

where $W^{up\_proj} \in \mathbb{R}^{d \times d_{model}}$, $W^{down\_proj} \in \mathbb{R}^{d_{model} \times d}$ and $d_{model} > d$.

This structure allows the FNN to scale effectively and is a key reason why Transformer-based models perform so well in various natural language processing tasks, such as machine translation, text summarization, question answering, and text generation [Bubeck et al. 2023]. In the larger GPT-3, which has 175 billion parameters, about one-third of the parameters are dedicated to the attention mechanism, a small proportion is used for bias and normalization layers, while nearly two-thirds of the parameters are allocated to the feedforward network.

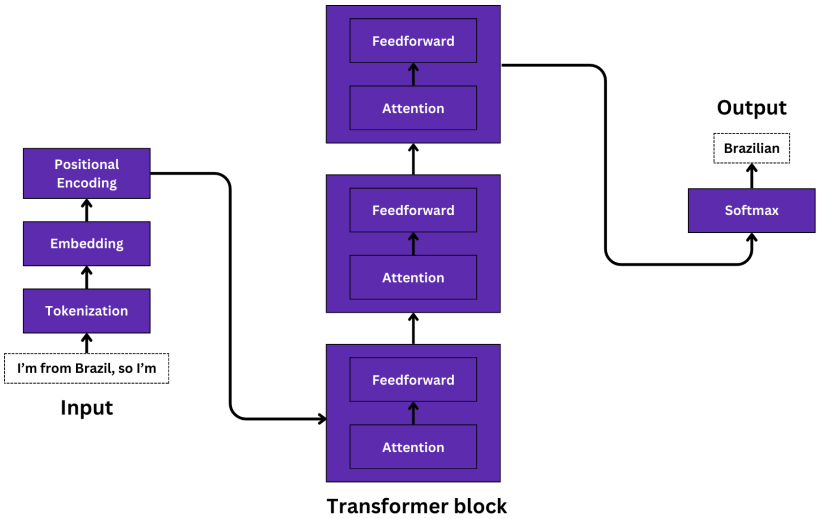

Figure 1. Transformer-based architecture.

Prompt Engineering. Prompt engineering is a discipline for developing and optimizing prompts to effectively perform tasks across various domains. It involves designing prompts to guide language models' behavior in tasks that require complex reasoning.

The zero shot prompt is the basic instruction format given to a pre-trained model:

Prompt:

> **Task:** Classify the text into negative or positive.
> **Text:** I love Brazil!
> **Sentiment:**

Output:

> Positive

Introducing minimal examples - a few shot prompts - enables in-context learning [Dong et al. 2022] for tasks where the model may lack knowledge. Consider the following example presented in [Brown 2020]:

Prompt:

> A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
> We were traveling in Africa and we saw these very cute whatpus.
>
> To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

Output:

> One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.

This structured approach enhances the model's ability to generate contextually relevant outputs. For tasks that require multiple reasoning steps to perform correctly, few shot prompting shows limitations. Consider the following example with a wrong output:

Prompt:

> **Q:** Roger has 6 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
> **A:** The answer is 11.
> **Q:** The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Output:

> **A:** The answer is 27.

As demonstrated by [Wei et al. 2022], Chain-of-Thought prompting facilitates complex

reasoning tasks by breaking them down into structured steps. It guides the model through a sequence of logical connections, ensuring that each step builds upon the previous one. This method is particularly effective for tasks involving arithmetic, commonsense reasoning, and symbolic logic, where more than simple prompt-answer pairs may be needed. By encouraging a chain of reasoning, the model can achieve more accurate and context-relevant responses, as shown by the referenced paper in the following example:

Prompt:

> **Q:** Roger has 6 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
> **A:** Roger started with 6 tennis balls. He bought 2 more cans of 3 tennis balls each, totaling 6 tennis balls. Thus, 6 + 6 = 12. The answer is 12.
> **Q:** The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Output:

> **A:** The cafeteria originally had 23 apples. They used 20 to make lunch, leaving them with 3 apples. They bought 6 more apples, so now they have 3 + 6 = 9 apples. The answer is 9.

OpenAI's o1 model [El-Kishky et al. 2024] utilizes the Chain-of-Thought paradigm, enhanced by Reinforcement Learning from Human Feedback (RLHF) [Lightman et al. 2023], to improve complex reasoning. By breaking tasks into structured steps, o1 excels in areas like competitive programming, mathematics, and scientific problem-solving. While this boosts performance, it also increases inference time due to the model's more deliberate, step-by-step processing.

Retrieval-Augmented Generation. RAG enhances the capabilities of language models by integrating information retrieval systems [Lewis et al. 2020]. This approach allows models to retrieve relevant documents from external databases and incorporate knowledge into their prompts, as illustrated in Figure 2. By dynamically accessing up-to-date information during inference, models can provide more informed and contextually rich outputs. This challenges the fine-tuning approach, as the model can complement its pre-trained knowledge with external data without extra training effort. The evident side effect of RAGs is the increased inference latency due to the larger input size from the retrieved documents, as highlighted by [Zhu et al. 2024].

Low-Rank Adaptation. Low-rank structures are common in machine learning [Kulis et al. 2006] for problems with intrinsic lower-dimensional spaces [Li et al. 2018]. Overparameterized neural networks exhibit low-rank properties after training, which can be explicitly used during training to improve efficiency [Sainath et al. 2013]. Large language models, as demonstrated by [Aghajanyan et al. 2020], can be reduced to their intrinsic dimensionality with low-performance loss.

LoRA [Hu et al. 2021] is a fine-tuning method designed to reduce the number of trainable

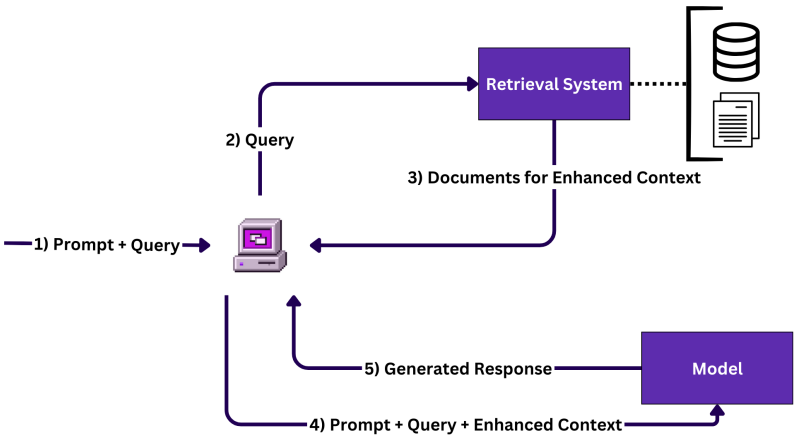

Figure 2. Retrieval-Augmented Generation architecture.

parameters in large language models. The key idea is to decompose the update matrices into two low-rank matrices, significantly reducing the number of parameters needed during the fine-tuning process. This method keeps the pre-trained model weights fixed and injects trainable low-rank matrices into any trainable layer of the architecture. The weight update in LoRA is represented as:

$$W = W_0 + \Delta W = W_0 + A_{d\times r} \times B_{r\times d}$$

where W_0 is the pre-trained weight matrix, ΔW is the update matrix, and A and B (initialization illustrated in Figure 3) are the low-rank matrices with dimensions $d \times r$ and $r \times d$, respectively. The rank r is a training hyperparameter typically much smaller than the dimensions of the weight matrix, making the update more parameter-efficient. The resulting model retains most of the performance benefits of full fine-tuning while adjusting significantly fewer trainable parameters, requiring less computational resources for task adaptation.

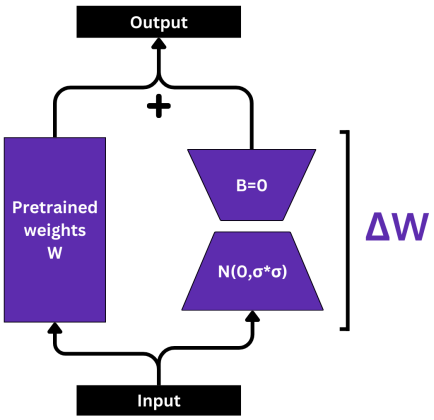

Figure 3. LoRA architecture.

Mixture of Experts. The Mixture of Experts (MoE) paradigm enhances prediction by dynamically combining specialized models, improving performance through majority vot-

ing [Opitz and Maclin 1999]. Each expert model handles a specific input subset, while a gating network assigns inputs to the most appropriate expert [Jacobs et al. 1991]. MoE architectures significantly increase model capacity in language models without proportional increases in computational resources during inference [Shazeer et al. 2017]. By activating only a subset of experts for each input, MoE models achieve high performance across tasks while maintaining efficiency. Notably, the Switch Transformers [Fedus et al. 2022] takes this concept further by scaling model size to trillions of parameters, achieving even greater sparsity by routing each input to a single expert.

Gated Neural Networks. Gated neural networks incorporate gating mechanisms to regulate the flow of information through a mixture of expert layers, whose experts make regression decisions with gating controls to weigh the decisions [Makkuva et al. 2020]. These mechanisms work as dynamic filters that control the passage of information, allowing the network to learn and model complex patterns and dependencies in data. The fundamental idea behind gating is to use a gate, typically defined by the equation:

$$g(\vec{x}) = \sigma(W \cdot \vec{x} + b),$$

where σ represents a non-linear activation function (generally some sigmoid function), W is the weight matrix, $\vec{x}$ is the input, and b is the bias. This gating function scales the input, effectively controlling how much information can influence the subsequent layers. The concept of gating has been pivotal in advancing natural language processing, with Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] and Gated Recurrent Unit (GRU) [Chung et al. 2014] networks playing a crucial role in overcoming the *vanishing gradient problem*.

Multi-task LoRA. While LoRA is primarily designed to specialize a model for a single downstream task, two approaches have been developed to handle multiple tasks. One method involves concatenating datasets from different tasks to create a single LoRA adaptation. However, [Ling et al. 2023] highlights significant challenges in adapting a single model to multiple tasks due to the heterogeneity of domain data, the complexity of domain-specific knowledge, the uniqueness of domain objectives, and the diversity of constraints (including social norms, cultural conformity, religious beliefs, and ethical standards) across different domains.

A more practical approach for exploiting LoRA adapters involves creating multiple adapters and dynamically switching between them within a single pre-trained language model, as illustrated in Figure 4. Frameworks such as HuggingFace's Parameter-Efficient Fine-Tuning (PEFT) [Mangrulkar et al. 2022], S-LoRA [Sheng et al. 2023], and LoRAHub [Huang et al. 2023] support this adapter-switching capability, but lack efficient mechanisms for dynamic and memory-conserving operations. MeteoRA [Xu et al. 2024] offers a more efficient alternative, but it is limited to specific datasets. By incorporating a Mixture of Experts (MoE) strategy within the HuggingFace framework, X-LoRA [Buehler and Buehler 2024] dynamically activates LoRA adapters based on the input task. It integrates a token- and layer-level scaling mechanism, enabling fine-grained adjustments across multiple layers. The X-LoRA scaling head, using hidden states from the base model, predicts scaling coefficients through linear fully-connected layers. This scaling head efficiently utilizes complex encodings, implemented as a shallow feed-forward

neural network. Moreover, LoRA Land [Zhao et al. 2024], which hosts 25 LoRA-adapted models governed by the X-LoRA approach, positions itself as a strong competitor to GPT-4 across various tasks. This underscores X-LoRA's capacity for flexible, scalable fine-tuning, effectively balancing task diversity and inference efficiency.

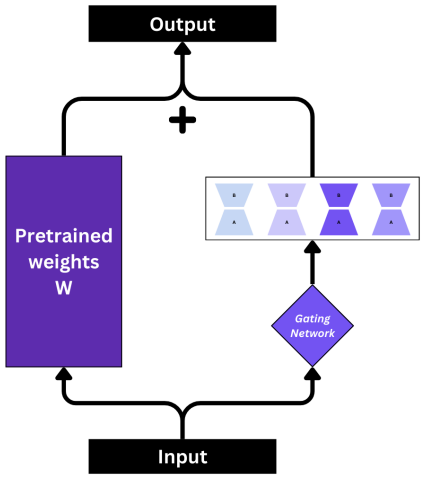

Figure 4. Multi-LoRA gated architecture.

3. Our Approach

This work presents a detailed experimental evaluation of the low-rank adaptation approach for parameter-efficient fine-tuning across multiple tasks within a single pre-trained Transformer model. By employing low-rank adapter matrices, this method is designed to enhance the efficiency and flexibility of language models, enabling them to handle a diverse set of tasks while minimizing inference time. The integration of X-LoRA within the Hugging Face framework facilitates its application to various natural language processing tasks, without requiring extensive computational resources or multiple large task-specific models. Through this investigation, we aim to provide a deeper understanding of how to effectively optimize language models for diverse applications, enhancing adaptability and performance, while shedding light on the underlying mechanisms of Transformer modules.

3.1. Multitasking Low-Rank Structures

For N-defined dataset tasks, the LoRA-multitasking approach involves performing separate LoRA processes, producing matrices A and B for each task:

$$S_{\text{multi}} = \{(A_i, B_i)\}_{i=1}^{N}$$

For the i-th task, the low-rank matrices A_i and B_i are combined to create task-specific updates:

$$\Delta W_i = A_i \times B_i$$

These updates are then applied to the shared pre-trained weights W_0, for a given task input $\vec{x}$, it yielding task-specific weights with a forward pass σ_i as follows:

$$\sigma_i = \vec{x}W_0 + \vec{x}\Delta W_i = \vec{x}W_0 + ((\vec{x} \times A_i) \times B_i)$$

A gating network determines the appropriate LoRA adapters for a given input task. The resulting weight update is then a linear combination of the matrices for multiple tasks, weighted by the gating results:

$$\Delta W_{\text{multi}} = \sum_{i=1}^{N} w_i \cdot ((\vec{x} \times A_i) \times B_i)$$

Consequently, the overall forward pass update considering the gating is:

$$\sigma_{multi} = \vec{x}W_0 + \vec{x}\Delta W_{\text{multi}} = \vec{x}W_0 + \sum_{i=1}^{N} w_i \cdot ((\vec{x} \times A_i) \times B_i)$$

3.2. Applying LoRA to the Transformer Weights

The core components of the attention mechanism in the Transformer architecture are the Query, Key, and Value matrices. These matrices are essential for calculating attention scores and determining how different parts of the input sequence contribute to the output by facilitating the exchange of information across the sequence. Initially, the Q, K, and V matrices are randomly initialized and learn to capture the relationships between different parts of the sequence during training [Dubey et al. 2024]. The Q matrix represents the semantic intent of each word in the context of the input sequence, while the K matrix acts as an identifier, providing a means to compare queries against all keys. The V matrix contains the actual values or representations of the words in the sequence.

The attention mechanism shown in Equation 1 computes the attention scores by taking the Q and K matrices' dot product, highlighting the syntactic and semantic relationships between words. Then, a softmax function is applied to create a probability distribution to weight the V matrix, effectively combining the contributions of different words based on their relevance to each other. The output of this attention mechanism is then transformed using an output projection matrix W^O, which integrates the attended information into the model's representation space.

Our approach explicitly focuses on adapting the Transformer weights' W_q (query) and W_v (value) in the attention module for parameter efficiency. As referenced in LoRA's paper, query and value matrices adapted with a rank much smaller than the full rank can successfully specialize the model with outstanding performance. The low-rank matrices A_i and B_i are applied to the i^{th} attention head in the multi-head attention mechanism as follows:

$$W_i^Q = W_0^Q + \Delta W_i^Q = W_0^Q + A_i^Q B_i^Q$$
$$W_i^V = W_0^V + \Delta W_i^V = W_0^V + A_i^V B_i^V$$

While we specifically exclude the feedforward layers from adaptation for parameter efficiency, LoRA can adapt any neural network. Previous works have shown that when fine-tuning Transformer-based models, the attention mechanism is the most essential for capturing task-specific information. However, the underlying mechanisms behind fine-tuning are still not fully understood. Future work could investigate how and what changes occur in the model's space through the subtle adjustments of LoRA adapters. Understanding these transformations in the model's intrinsic dimensional space could provide valuable insights into how language models represent and interpret language, offering a clearer view of their behavior. This investigation may also shed light on the trade-offs between increased complexity and potential performance gains when including feedforward layers.

3.3. Experimental Setup

To evaluate the effectiveness of our LoRA-multi approach, we conduct experiments using the BERT [Devlin 2018] and DistilBERT [Sanh et al. 2019] family of models pre-trained on the BookCorpus (800M words) [Zhu et al. 2015] and a filtered version of English Wikipedia (2,500M words). We develop a separate expert as a LoRA adapter for each proposed downstream task with the HuggingFace framework [Wolf 2019], which is then integrated into a mixture of experts with the X-LoRA framework. The performance of our LoRA-multi approach is compared against the fully tuned models.

The evaluation uses task-specific metrics and training time. This comprehensive analysis highlights the balance between performance and efficiency that our LoRA-multi approach achieves, offering insights into its potential for real-world applications in multi-task language modeling.

4. Experiments

In this section, we present the experimental evaluation of our LoRA-multi approach across a set of well-established natural language processing tasks. By focusing on these benchmarks, we aim to demonstrate the effectiveness of the proposed methods in adapting to different linguistic challenges while maintaining computational efficiency by minimizing the model to its intrinsic dimension through training parameter reduction.

4.1. Downstream Tasks

Situations With Adversarial Generations (SWAG) [Zellers et al. 2018]: SWAG is a large-scale dataset designed to evaluate grounded common sense inference by combining linguistic and visual knowledge. It contains 113,000 multiple-choice questions (73,000 for training, 20,000 for validation, and 20,000 for testing) based on video captions from the Large Scale Movie Description Challenge or ActivityNet Captions [Krishna et al. 2017]. Each question asks models to select the most plausible continuation of a given video caption, with incorrect answers being adversarially generated to challenge models. The dataset uses Adversarial Filtering [Le Bras et al. 2020] to ensure diverse and challenging problems. For example:

Prompt:

> **Context:** Members of the procession walk down the street holding small horn brass instruments.
> 0 - A drum line passes by walking down the street playing their instruments.
> 1 - A drum line has heard approaching them.
> 2 - A drum line arrives and they're outside dancing and asleep.
> 3 - A drum line turns the lead singer watches the performance.
> **Response:**

Output:

> 0

Stanford Question Answering Dataset (SQuAD) [Rajpurkar 2016]: SQuAD is a large-scale machine reading comprehension dataset designed to challenge models to extract precise answers from a given passage of text. It consists of 100,000+ question-answer pairs derived from over 500 Wikipedia articles. Crowdworkers pose questions based on a specific passage, with the answers being spans of text found directly in the context. The dataset is divided into training (87,000) and validation (10,000). Each passage typically has multiple questions, allowing the model to learn fine-grained text comprehension. The task emphasizes understanding specific details, factual recall, and precise extraction of information. For example:

Prompt:

> **Context:** The Apollo program was the third United States human spaceflight program carried out by NASA, which accomplished landing the first humans on the Moon from 1969 to 1972.
> **Question:** When did the Apollo program accomplish the first moon landing?
> **Answer:**

Output:

> 1969

WNUT17 Emerging Entities [Derczynski et al. 2017]: This task identifies unusual, previously unseen entities in emerging discussions, particularly in noisy text environments like Twitter, Reddit, YouTube, and StackExchange. Named Entity Recognition (NER) systems typically struggle with rare or novel entities, which this dataset aims to address. It consists of 5,690 texts from multiple sources, divided into training (3,394), validation (1,009), and test (1,287). The task challenges NER models to generalize beyond well-known entities, pushing them to handle the dynamic nature of language in real-world, user-generated content.

Prompt:

Context: @paulwalk It's the view from where I'm living for two weeks. Empire State Building=ESB. Pretty bad storm here last evening.
Answer:

Output:

('@paulwalk', 'O'), ('It', 'O'), ("'s", 'O'), ('the', 'O'), ('view', 'O'), ('from', 'O'), ('where', 'O'), ('I', 'O'), ("'m", 'O'), ('living', 'O'), ('for', 'O'), ('two', 'O'), ('weeks', 'O'), ('.', 'O'), ('Empire', 'B-location'), ('State', 'I-location'), ('Building', 'I-location'), ('=', 'O'), ('ESB', 'B-location'), ('.', 'O'), ('Pretty', 'O'), ('bad', 'O'), ('storm', 'O'), ('here', 'O'), ('last', 'O'), ('evening', 'O'), ('.', 'O')

4.2. Adaptations Procedure

Hyperparameters. The training hyperparameters were consistent across all experiments, except for the number of epochs, which varied by task-specific dataset to evaluate its impact on model performance. Key parameters included the learning rate, batch size for training and evaluation, and the number of epochs. A comprehensive summary of the hyperparameters for each task is presented in Table 1. For the LoRA adapters, we implemented low-rank adaptation with a fixed rank of 8. The rank choice is supported by the findings from the LoRA paper, which identified it as an effective starting point for adapting Transformer's query and value matrices across various tasks. The rank provides a balance between performance and parameter efficiency.

Dataset	Learning Rate	Batch Size	Epochs	LoRA Rank
SQuAD	5e-5	16	3	8
SWAG	5e-5	16	5	8
WNUT17	5e-5	16	10	8

Table 1. Training hyperparameters across multiple tasks for models adapted with full fine-tuning (FT) and low-rank adaptation (LoRA) approaches.

Machinery. Model adaptations were primarily conducted on a single Tesla L4 GPU with 22.5GB of RAM, accessed through Google Colab [Carneiro et al. 2018], chosen for its optimal balance of computational power and efficiency. To investigate performance under more resource-constrained conditions, we expanded our experiments to include a single Tesla T4 GPU with 16GB of RAM, also provided by Google Colab. This allowed for a comparative analysis of training times and adaptation performance across different hardware configurations. Specifically, we evaluated the performance of DistilBERT and BERT-base on the SQuAD dataset using the T4 GPU, as detailed in Table 2.

The analysis of training times reveals significant insights across different tasks and fine-tuning methods. The WNUT17 task (Table 4) implemented an early stopping strategy in all approaches with a maximum of 5 epochs to optimize training efficiency. Early stopping

is a regularization technique that halts training when the model's performance on the validation set ceases to improve, thereby preventing overfitting and reducing unnecessary computation while fully fine-tuning.

Model & Method	Hardware GPU	SQuAD Epochs	Training Time	Time per Epoch
DistilBERT (FT)	T4	3	13046.00	4348.67
DistilBERT (LoRA)	T4	3	10493.03	**3497.67**
BERT-base (FT)	T4	3	25524.57	8508.19
BERT-base (LoRA)	T4	3	19854.15	**6618.05**
BERT-large (FT)	L4	3	36168.27	12056.09
BERT-large (LoRA)	L4	3	28118.33	**9372.78**

Table 2. Training times (in seconds) for various models on the SQuAD task, comparing full fine-tuning (FT) and low-rank adaptation (LoRA) methods.

Model & Method	Hardware GPU	SWAG Epochs	Training Time	Time per Epoch
DistilBERT (FT)	L4	5	1547.32	309.464
DistilBERT (LoRA)	L4	5	1004.11	**200.82**
BERT-base (FT)	L4	5	2662.15	532.43
BERT-base (LoRA)	L4	5	3107.52	621.50
BERT-large (FT)	L4	5	8699.09	1739.82
BERT-large (LoRA)	L4	5	8522.51	**1704.50**

Table 3. Training times (in seconds) for various models on the SWAG task, comparing full fine-tuning (FT) and low-rank adaptation (LoRA) methods.

Model & Method	Hardware GPU	WNUT17 Epochs	Training Time	Time per Epoch
DistilBERT (FT)	L4	6	79.94	13.32
DistilBERT (LoRA)	L4	10	84.744	**8.47**
BERT-base (FT)	L4	6	120.19	20.03
BERT-base (LoRA)	L4	10	143.80	**14.38**
BERT-large (FT)	L4	7	439.05	62.72
BERT-large (LoRA)	L4	10	244.80	**24.48**

Table 4. Training times (in seconds) for various models on the WNUT17 task, comparing full fine-tuning (FT) and low-rank adaptation (LoRA) methods. All approaches employed *early stopping* with a maximum of 5 epochs.

Upon examining the time in seconds taken to iterate over the dataset while adapting the model, it becomes evident that using the LoRA approach substantially decreases the training duration. This consistent reduction highlights LoRA's ability to maintain efficient performance (presented in Table 5) while significantly lowering computational costs, demonstrating its effectiveness in resource-constrained environments.

4.3. Evaluation

We evaluate the models on downstream tasks using task-specific metrics on their respective validation sets. For SWAG, we use accuracy to quantify the model's proficiency in situational reasoning through multiple-choice scenarios.

For SQuAD, we employ two complementary metrics: F1 score and Exact Match (EM). The F1 score, calculated as the harmonic mean of precision and recall, offers a nuanced view of answer quality. It accounts for partial matches, balancing the proportion of correct words in the predicted answer (precision) against the proportion of ground truth words captured (recall). Each question's score is computed individually and then averaged across the dataset. Human performance on SQuAD, as reported by [Rajpurkar 2016], achieved an F1 score of 90.5% and an Exact Match score of 80.3%.

For WNUT17, we assess performance using accuracy and F1 score, providing insight into the system's effectiveness in recognizing rare entities in noisy text.

Higher scores in the metrics presented in Table 5 indicate superior performance in the respective tasks.

Model & Method	**# Trainable Parameters**	**SWAG Acc**	**SQuAD EM / F1**	**WNUT17 Acc / F1**
DistilBERT (FT)	67M	69.6	77.5 / 85.7	94.7 / 49.42
DistilBERT (LoRA)	0.15M	64.3	58.5 / 69.8	93.6 / 29.8
BERT-base (FT)	110M	77.6	80.5 / 88.2	94.8 / 44.5
BERT-base (LoRA)	0.3M	75.0	63.6 / 73.9	93.6 / 29.5
BERT-large (FT)	340M	80.2	83.1 / 90.5	94.9 / 48.6
BERT-large (LoRA)	0.8M	82.9	74.1 / 83.8	93.4 / 27.68

Table 5. Different models adapted with full fine-tuning (FT) and low-rank adaptation (LoRA) approaches across multiple tasks.

The performance analysis of WNUT17 and SWAG demonstrates that, for the majority of models, we achieve approximately 90% of the full fine-tuning performance while adapting less than 1% of the parameters. This efficiency highlights the advantage of LoRA in minimizing computational overhead without significantly compromising performance. Notably, for SWAG, the BERT-large surpassed the performance of full fine-tuning on the validation set. This result suggests that LoRA can leverage the capacity of larger models with billions of parameters in specific tasks, even with a small fraction of the parameters being adapted.

4.4. Mixturing Experts

This section demonstrates how integrating LoRA adapters enables high performance across multiple tasks while significantly reducing training time. The X-LoRA framework achieves a substantial reduction in the number of trainable parameters, as shown in Table 6, allowing for efficient *mixture-of-experts* with minimal computational overhead and reduced inference time. DistilBERT, however, was not suitable for the X-LoRA framework due to limitations in its HuggingFace implementation.

Through the X-LoRA framework, a single model can perform inference using different combinations of low-rank adapters, each contributing to the overall performance in spe-

Mixture of Experts	# All Parameters	# Trainable Parameters
BERT-base (X-LoRA)	164M	53M
BERT-large (X-LoRA)	392M	57M

Table 6. Different models mixturing the low-rank adapters. Less than 50% of the combined pre-trained, adapters and scaling mechanism parameters will be trainable.

cific tasks. This flexibility allows the model to adaptively adjust its decision weights, optimizing the influence of each adapter based on the context of the input. The ability to mix and match adapter contributions empowers organizations to fine-tune models for a range of use cases without extensive retraining, ultimately streamlining workflows and improving efficiency.

4.5. Conclusion

This study has demonstrated the efficacy of the LoRA framework in fine-tuning Transformer models across multiple NLP tasks by adapting less than 1% of the model parameters. By focusing on adapting the Transformer's intrinsic dimension with a low rank, we showed that combining multiple adapters within a single pre-trained model can yield competitive results while drastically reducing computational costs. Despite the limitations in the Hugging Face implementation for DistilBERT, which hindered its suitability for X-LoRA, our primary goal of adapting to multiple tasks using LoRA has been successfully achieved.

Additionally, we highlight that the entire process, including fine-tuning and inference across multiple tasks, can be executed on affordable platforms such as Google Colab for less than $30. This reinforces the practical feasibility of adopting LoRA-based multi-task models in real-world applications without requiring high-end infrastructure. By reducing costs and maintaining strong performance, X-LoRA offers a compelling approach for deploying specialized language models in diverse settings.

5. Future Works

The underlying mechanism behind fine-tuning in language models remains unclear. A significant question is how the semantic-syntactic space is transformed to perform effectively in downstream tasks. Future research can explore the insights gained from this work to deepen our understanding of how low-rank adapters influence the intrinsic dimensions of the syntactic-semantic space. This understanding could provide insights into how Transformer-based language models specialize in specific tasks while maintaining versatility.

The adaptation of value and key weights in the attention layers is an experimental choice, but the reasoning behind why these matrices enable the model to perform well on specific tasks has yet to be explained. Adapting only the value and key weights is a limitation of this project, aimed at maintaining low training costs while addressing multiple tasks. The inclusion of LoRA adapters in the feedforward neural networks of the Transformer architecture brings significant benefits to performance and paves the way for understanding how language models encode information.

Moreover, companies can adopt these adaptations strategy to deploy a single model capable of efficiently performing multiple productable tasks. LoRA Land hosts 25 fine-tuned Mistral-7B LLMs on a single NVIDIA A100 GPU, showcasing the cost-effectiveness and enhanced performance of multiple specialized models over a single general-purpose model. This integration streamlines workflows and optimizes resource utilization, ultimately driving productivity and innovation in AI-driven applications.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al. (2023). Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Anthropic (2024). Claude 3.5 sonnet model card addendum. *Techincal Report*.

Brown, T. B. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Buehler, E. L. and Buehler, M. J. (2024). X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2).

Carneiro, T., Medeiros Da NóBrega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., and Filho, P. P. R. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Clark, K. (2019). What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Cui, J., Ning, M., Li, Z., Chen, B., Yan, Y., Li, H., Ling, B., Tian, Y., and Yuan, L. (2023). Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model. *arXiv preprint arXiv:2306.16092*.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. (2023). Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Derczynski, L., Nichols, E., van Erp, M., and Limsopatham, N. (2017). Results of the WNUT2017 shared task on novel and emerging entity recognition. In Derczynski, L., Xu, W., Ritter, A., and Baldwin, T., editors, *Proceedings of the 3rd Workshop on*

Noisy User-generated Text, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. (2022). A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

El-Kishky, A., Selsam, D., Song, F., Parascandolo, G., Ren, H., Lightman, H., Chung, H. W., Akkaya, I., Sutskever, I., Wei, J., Gordon, J., Cobbe, K., Yu, K., Kondraciuk, L., Schwarzer, M., Rohaninejad, M., Brown, N., Zhao, S., Bansal, T., Kosaraju, V., and Zhou, W. (2024). Learning to reason with llms. *Techincal Report*.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. (2022). Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Engels, J., Liao, I., Michaud, E. J., Gurnee, W., and Tegmark, M. (2024). Not all language model features are linear. *arXiv preprint arXiv:2405.14860*.

Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Geiger, A., Wu, Z., Potts, C., Icard, T., and Goodman, N. (2024). Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pages 160–187. PMLR.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. (2023). Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Niebles, J. C. (2017). Dense-captioning events in videos. In *International Conference on Computer Vision (ICCV)*.

Kulis, B., Sustik, M. A., and Dhillon, I. S. (2006). Learning low-rank kernel matrices. *Proceedings of the 23rd international conference on Machine learning*.

Le Bras, R., Swayamdipta, S., Bhagavatula, C., Zellers, R., Peters, M., Sabharwal, A., and Choi, Y. (2020). Adversarial filters of dataset biases. In *International conference on machine learning*, pages 1078–1088. Pmlr.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023). Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Ling, C., Zhao, X., Lu, J., Deng, C., Zheng, C., Wang, J., Chowdhury, T., Li, Y., Cui, H., Zhang, X., et al. (2023). Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Liu, Y., Singh, A., Freeman, C. D., Co-Reyes, J. D., and Liu, P. J. (2023). Improving large language model fine-tuning for solving math problems. *arXiv preprint arXiv:2310.10047*.

Makkuva, A., Oh, S., Kannan, S., and Viswanath, P. (2020). Learning in gated neural networks. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3338–3348. PMLR.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. (2022). Peft: State-of-the-art parameter-efficient fine-tuning methods. *URL: https://github.com/huggingface/peft*.

Mikolov, T. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nanda, N., Rajamanoharan, S., Kramár, J., and Shah, R. (2023). Fact finding: Attempting to reverse-engineer factual recall on the neuron level. In *AI Alignment Forum, 2023c. URL https://www. alignmentforum. org/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall*, page 19.

Nunes, D. and Antunes, L. (2018). Neural random projections for language modelling. *arXiv preprint arXiv:1807.00930*.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Qin, B., Hui, B., Wang, L., Yang, M., Li, J., Li, B., Geng, R., Cao, R., Sun, J., Si, L., et al. (2022). A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629*.

Rajpurkar, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.-b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E., and Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arxiv 2019. *arXiv preprint arXiv:1910.01108*.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sheng, Y., Cao, S., Li, D., Hooper, C., Lee, N., Yang, S., Chou, C., Zhu, B., Zheng, L., Keutzer, K., et al. (2023). S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wolf, T. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Work, W. M. I.-C. L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? *ACL anthropology*.

Xu, J., Lai, J., and Huang, Y. (2024). Meteora: Multiple-tasks embedded lora for large language models. *arXiv preprint arXiv:2405.13053*.

Zellers, R., Bisk, Y., Schwartz, R., and Choi, Y. (2018). Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhang, B. and Sennrich, R. (2019). Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Zhao, J., Wang, T., Abid, W., Angus, G., Garg, A., Kinnison, J., Sherstinsky, A., Molino, P., Addair, T., and Rishi, D. (2024). Lora land: 310 fine-tuned llms that rival gpt-4, a technical report. *arXiv preprint arXiv:2405.00732*.

Zhu, Y., Gu, J.-C., Sikora, C., Ko, H., Liu, Y., Lin, C.-C., Shu, L., Luo, L., Meng, L., Liu, B., et al. (2024). Accelerating inference of retrieval-augmented generation via sparse context selection. *arXiv preprint arXiv:2405.16178*.

Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2019). A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685.