Investigating Fine-Tuning of Language Models for Multiple-Choice Questions

by

Ivy A. Wang

B.S. in Computer Science and Engineering and Mathematics, MIT, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by:	Ivy A. Wang Department of Electrical Engineering and Computer Science August 30, 2024
Certified by:	Erik Hemberg Research Scientist, Thesis Supervisor
Certified by:	Una-May O'Reilly Principal Research Scientist, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair Master of Engineering Thesis Committee

Investigating Fine-Tuning of Language Models for Multiple-Choice Questions

by

Ivy A. Wang

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

This thesis investigates the positional and contextual bias of large language models (LLMs) when used to answer multiple-choice questions (MCQs). Given the increasing use of generative language models in fields ranging from cybersecurity to biomedical research, it is important to understand the causes of their behavior in order to mitigate biases and prevent errors. One known method of improving the performance of LLMs is fine-tuning, wherein a model is additionally trained on data from a specified distribution or subject area.

We specifically investigate training data properties related to positional bias in fine-tuned language model performance on correctly answering MCQs. To improve model efficiency, we used parameter-efficient fine-tuning, specifically LoRA (Low-Rank Adaptation), which reduces the dimensionality of weight matrices used in the model's layers. We verify that if the training data for the model possesses the same qualities and distributions as the test data, the LLM will achieve the best performance. In our experiments, we scaled and balanced our fine-tuning datasets and learned that both processes improve the accuracy on test sets of MCQs.

Thesis supervisor: Erik Hemberg
Title: Research Scientist

Thesis supervisor: Una-May O'Reilly
Title: Principal Research Scientist

Acknowledgments

I would like to express my immense gratitude to my advisor Erik Hemberg for his consistent guidance, support, and insights throughout this thesis. His mentorship has helped me grow exponentially as a student, researcher, and writer.

I am also extremely grateful to my supervisor, Una-May O'Reilly, for fostering an encouraging and collaborative environment in ALFA, and for always lending a helping hand. I would also like to thank Stephen Moskal, Ethan Garza, Miguel Tulla, Iris Kremer, and Stephen Jorgensen for their advice and assistance throughout my research and thesis-writing endeavors.

I also extend thanks to the entire ALFA group for providing the resources and facilities necessary for this work. Finally, I would like to acknowledge the support of my family and friends, whose encouragement and understanding have been a constant source of motivation.

Thank you all for your support and contribution to this project.

Ivy A. Wang
August 30, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large language models (LLMs) are used to perform text generation tasks such as answering questions or responding to prompts, and do so through probabilistic processes that predict the words needed given a specified context(s) [1]. They consist of millions of parameters in the form of weight matrices that store the bulk of their information. Even the simplest LLM architectures contain an embedding matrix layer that converts input tokens into numerical vectors, as shown in Figure 1.1. This embedding is then used to produce an output that helps predict or generate the next token.

Despite their increasingly widespread applications in fields from bioinformatics to cybersecurity, there is comparatively little understanding of their training and inference mechanisms, making it difficult to prevent errors and mitigate biases [2]. For example, LLMs may be better at recalling information presented at the start and end of the input, and worse at remembering the middle [3]. It is therefore important to further improve LLMs as well as make them safer and less biased in their applications.

One known issue with existing LLMs is the presence of positional bias, in which models' accuracy is affected by the order in which information is presented to it, such as in multiple-choice questions (MCQs) [4]. Figure 1.2 shows a True/False MCQ with answer options presented in different orders. This positional bias decreases LLMs' reliability for general use. Current solutions for mitigating positional bias often use black-box approaches that do not address the root causes of the issue, and are often computationally expensive, such as querying the model multiple times [5].

LLMs are storage-intensive and time-consuming to train, almost always requiring GPUs, as every instance of training includes the encoding, update, and decoding steps, and a considerable amount of storage is needed to store all the old and new parameters of the model. Additionally, many steps including in-context learning and post-processing are needed to achieve their generated responses.

Methods of improving LLM performance include in-weight learning, in which the model weights are changed through training and/or fine-tuning, and in-context learning, in which the prompts and inputs to the model are engineered to help it achieve better performance on specified tasks. With this in mind, we set a goal to minimize bias in answering multiple-

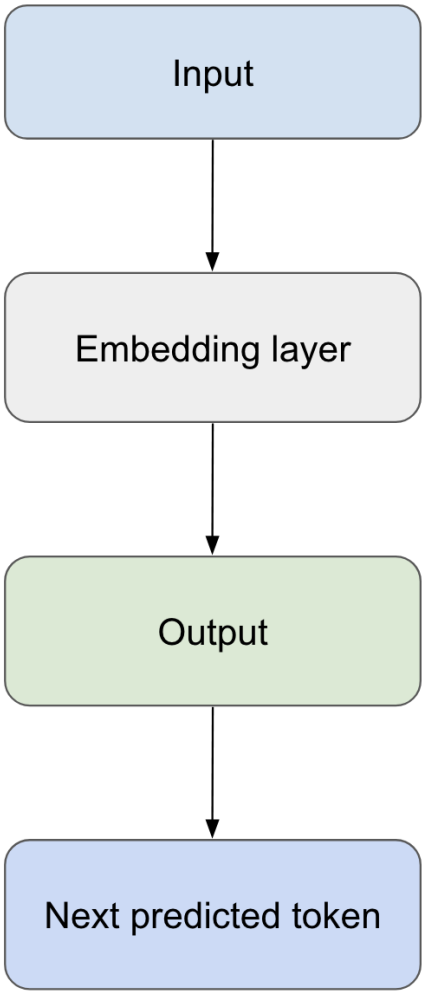

Figure 1.1: Diagram of the layers of a simple LLM. Most LLMs used in the present day contain many more layers, including transformer and attention layers.

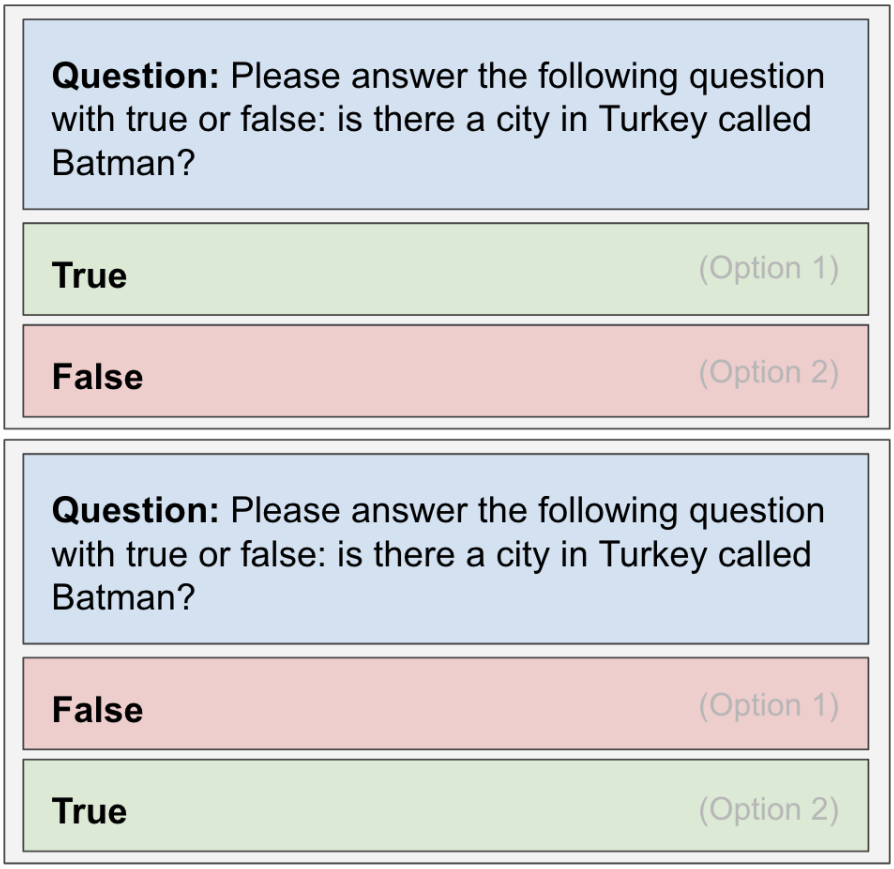

Figure 1.2: Example of a True/False multiple-choice question with answer options presented in two different orders. The green option represents the correct answer while the red option represents the incorrect answer. We want to present both of these question and answer sets to the model during fine-tuning so that it learns both answer positions equally.

choice questions through fine-tuning, as fine-tuning also produces a more robust model than prompt engineering. Improving both the model and the dataset saves time and resources that would be spent repeating the training steps. By improving our training data, we hope to discover methods that minimize inference costs and bias simultaneously.

In addition to investigating properties of the MCQ datasets used, we also focus on using low-resource methods to perform our experiments, including shorter training times and fewer model parameters, to better understand and reduce the costs of training LLMs. To improve model efficiency, we used Parameter-Efficient Fine-Tuning (PEFT), specifically LoRA (Low-Rank Adaptation), which reduces the dimensionality of matrices used in the model's weight layers. Instead of retraining all of the parameters of a pre-trained model during training, PEFT allows only a small subset of task-specific parameters to be updated, which greatly reduces the time and resource allocation needed for fine-tuning [6]. LoRA is one specific method of PEFT, in which some fully connected layers in a neural network are trained indirectly by optimizing rank decomposition matrices of these layers while keeping pre-trained weights frozen [7]. This also reduces the hardware barriers to entry for fine-tuning.

This project aims to investigate positional bias in language models by studying the effect of fine-tuning on accuracy in answering multiple-choice questions (MCQs). Relative to other uses of generative models, it is possible to immediately and definitively verify the accuracy of MCQs. Existing research in this area uses existing MCQ datasets to test the capabilities of LLMs [4, 5, 8]; however, little work is done in actually analyzing the properties of the datasets used, such as the ratios of correct answers and tokens used for answer options. In this thesis, we focus on constructing datasets that help perform optimal weight training in given language models. We take a more granular look at the datasets used in fine-tuning, and additionally check for bias in the actual tokens used in MCQs. In our thesis, we track the effects of different question types, dataset sizes, answer ratios, and answer tokens on language model accuracy after fine-tuning.

In this thesis, we ask two primary research questions:

- How can we reduce positional bias in language models without compromising their accuracy?
- Which characteristics of datasets result in the most meaningful changes and improvements?

Our contributions are listed below:

- We verified the ability of LoRA to fine-tune LLMs to reduce positional bias efficiently without compromising accuracy.
- We created curated datasets by focusing on properties such as doubling the MCQs used with different answer ordering, balancing the ratios of correct answers, and using different sets of answer tokens.
- We ran empirical experiments to mitigate bias in LLM performance on MCQs through fine-tuning on these datasets. Our experimental results showed us that having larger

and datasets with more balanced answer options maximizes the efficacy of fine-tuning to improve LLM performance.

Chapter 2 contains the background knowledge necessary to understand this project. Chapter 3 discusses previous works and research related to this thesis, as well as how this project builds on and differs from them. Afterwards, Chapter 4 details the methods used to conduct this research as well as the exact experiments conducted to answer the research questions. Chapter 5 conducts a discussion and analysis of the results of the experiments. Finally, Chapter 6 concludes the thesis and suggests related future works.

Chapter 2

Background

This chapter gives an overview of large language models and how they function (Section 2.1), as well as describing methods to improve their performance (Section 2.2 and Section 2.3).

2.1 Large Language Models

Large language models (LLMs) are used to perform text generation tasks such as answering questions or responding to prompts, and do so through probabilistic processes that predict the words needed given a specified context(s) [1]. Before models are trained, input information is broken into **tokens**, which can be characters, subwords, symbols, or words. For example, the True and False answer options in Figure 1.2 are tokens.

2.1.1 Transformer Architecture

Developments in language model research have introduced the **transformer architecture** (Figure 2.1), which allows **positional information** about tokens to be stored during the encoding process, typically in the form of position vectors [9], [3]. This information has been shown to affect the outputs of the decoding process, even in situations where positional bias is undesirable. As a result, various strategies and studies have been made in order to mitigate or even remove this bias, one of which is adding attention head layers within the model (Figure 2.1) [3]. Another method is to adapt the weights of the model, which will be discussed in Section 2.2.

2.1.2 Training, Data, and Bias

LLMs can be customized for a specific purpose by **training** the model on existing **labeled data** in order to update the model's weight layers and change the its behavior on its next interaction with both new and old inputs (Figure 2.2). This typically occurs by using a loss function to determine the LLM's performance on the output corresponding to each input, and continuously updating the model's weights to maximize its performance. However, this is an imperfect process; bias can occur during model training if data samples are disproportionate, labels are inconsistent, or information is ordered in some way [8]. **Positional bias** occurs

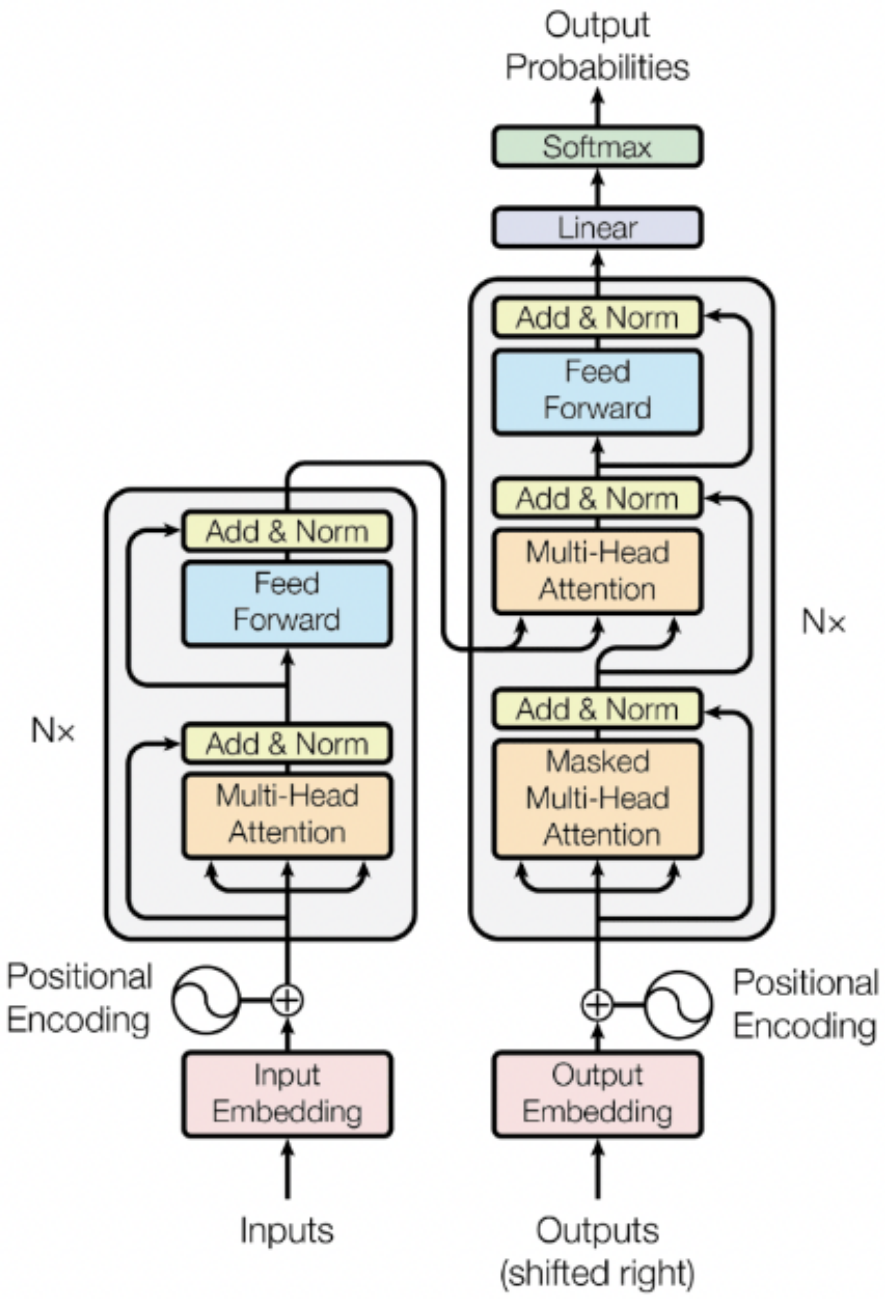

Figure 2.1: Basic structure of a transformer in an LLM with attention mechanisms to mitigate positional bias [3]. The positional encodings are vectors describing location of tokens in the inputs and outputs, and the multi-head attention layers allow the model to focus on different parts of the input sequence simultaneously.

when the model's outputs are influenced by the placement of information within the inputs, and can result in reduced accuracy when performing on new input data. This is an existing issue with LLMs as it can be exploited to produce skewed results and incorrect outputs [9].

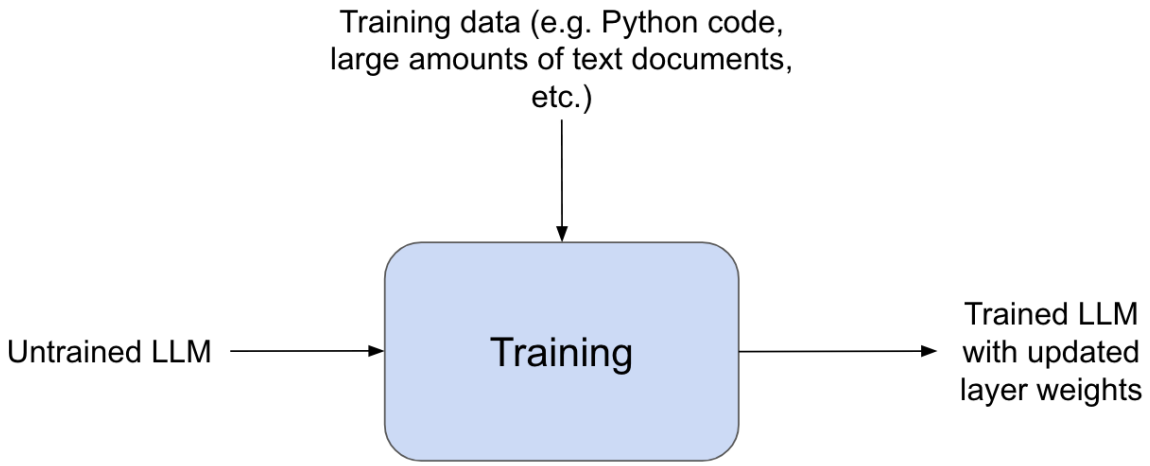

Figure 2.2: Training data is used to train a language model and update its parameters.

2.2 Mechanistic Interpretability

One way to investigate positional bias from LLMs is **mechanistic interpretability**, which aims to reverse engineer a neural network's computation process and obtain an exact piece-wise understanding of the model's behavior [10]. One of the main goals of mechanistic interpretability is to break down LLMs into human-readable pieces [11], and there are various methods for doing so, including patching.

Patching is a technique where a model's activations or weights are "patched", i.e. replaced, with other values during a run, with the goal of observing and quantifying behavioral changes caused by these modifications [2]. Since this technique targets specific, small components in the language model, it is a helpful tool for mechanistic interpretability and understanding how various parts of the model impact its performance.

2.3 Fine-Tuning

Fine-tuning takes a pre-trained model and adapts it to a specific task or dataset by further training it on task-specific data [7]. Fine-tuning updates weights throughout the model in order to improve its performance on a task or range of tasks. It is a slightly more complex approach to mechanistic interpretability than patching, but requires less overhead

than training the model from scratch, as only task-related weights will be updated. On the other hand, fine-tuning can also degrade the model's performance on other tasks, so it is important to consider the task domain while fine-tuning. Figure 2.3 shows how fine-tuning a pre-trained LLM on a MCQ dataset can adapt the LLM to better answer other MCQs. It is important to have the right dataset for fine-tuning, as this is the most impactful way to influence the model. For this reason, we focus on creating datasets with properties that best achieve our target performance on MCQs.

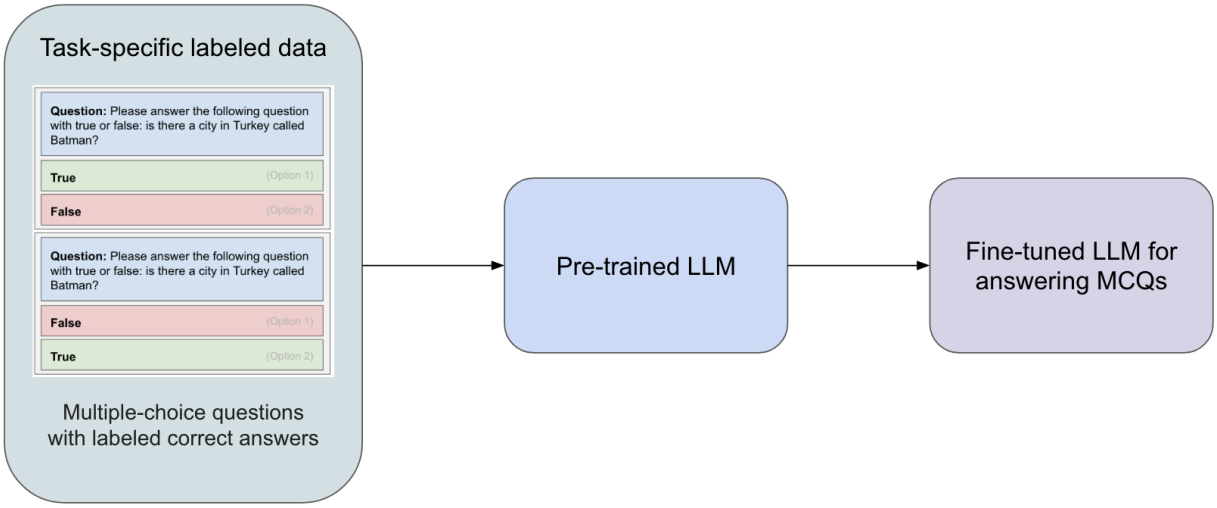

Figure 2.3: Example of fine-tuning on a MCQ dataset on a pre-trained LLM. The same question is duplicated, with the positions of the correct answers switched, to mitigate answer position bias.

2.3.1 Parameter-Efficient Fine-Tuning (PEFT)

The process of fine-tuning alone can often be costly both time-wise and resource-wise, as many parameters within the model are updated based on the input dataset. To mitigate this, researchers have created a broad category of methods labeled **Parameter-Efficient Fine-Tuning (PEFT)**. Existing PEFT methods include prompt-based learning (achieved through prefix tuning), reparametrization-based methods (including LoRA), series adapters, and parallel adapters [6]. In this case, **adapters** refer to specific layers that are changed during fine-tuning to customize the LLM for its intended purpose.

2.3.2 Low-Rank Adaptation (LoRA)

Low-rank adaptation, or LoRA, is one PEFT method that reduces the amount of weights needed to be updated during fine-tuning. In [7], Hu et al. detail the mathematics behind Low-Rank Adaptation and conduct experiments to show its benefits and limitations. Instead

of retraining all of the weights in a $\mathbb{R}^{d\times d}$ weight matrix, LoRA stores the weight matrix W as a linear combination $W_0 + BA$, where each of W_0, B, and A is $\mathbb{R}^{r\times d}$, $r \ll d$. During training, W_0 is frozen while A and B are updated with new parameters. Then for $h = W_0 x$, the forward pass results in

$$h = W_0 x + \Delta W x = W_0 x + BAx.$$

This process is illustrated in Figure 2.4.

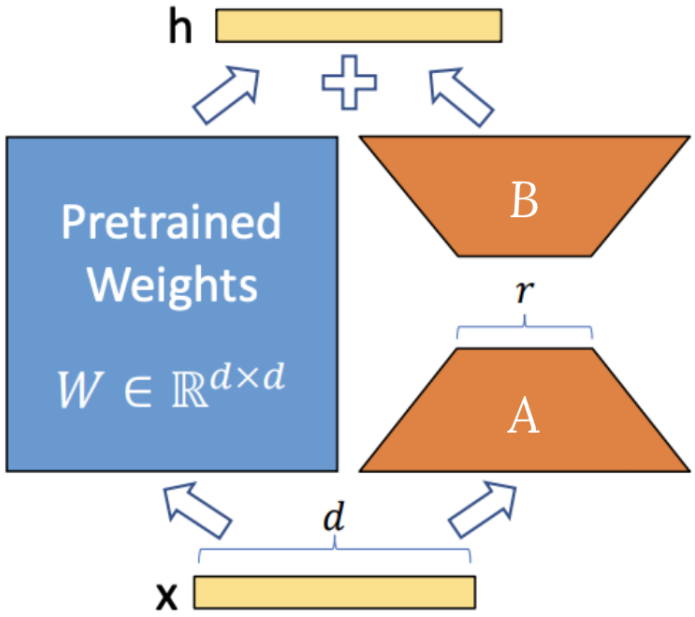

Figure 2.4: LoRA fine-tuning process: the pre-trained weights are reparameterized, and only matrices A and B are retrained [7].

Using a rank r that is significantly smaller than the full rank d is crucial to performing LoRA successfully. Empirically, in GPT-3 175B, a rank (r in Figure 2.4) as low as 1 suffices even when the full rank (d) is as high as 12,288 [7]. In addition, matrices A and B could easily be replaced to perform a different specified task, making LoRA both storage and compute efficient.

Chapter 3

Related Works

This chapter describes some work that has been previously done with PEFT and LoRA (Section 3.1), as well as previous studies of LLM performance on MCQs (Section 3.2). We then analyze the datasets and methods used in the previous works and describe how we plan to improve upon these experiments (Table 3.1).

3.1 Studies on PEFT and LoRA

3.1.1 PEFT

In [6], Hu et al. perform a series of experiments using several LLMs, four PEFT adapter structures, and several math and commonsense reasoning datasets. Their results show that smaller language models achieve comparatively better performance using PEFT adapters compared to larger language models, and that using multiple PEFT adapters simultaneously can further improve performance. In this study, the researchers sampled multiple datasets of math and commonsense reasoning questions, filtered out data that answered the questions incorrectly, and then compiled them to create the Math10k and Commonsense170k datasets [6]. Our thesis will use similar sampling methods to create datasets for fine-tuning LLMs, with the added steps of duplicating multiple-choice questions and swapping their answer choice positions to mitigate positional bias, as well as curating multiple datasets with different ratios of correct answer tokens to observe the impact of these variables on LLM performance after fine-tuning.

3.1.2 LoRA

The results of experiments from [6] suggest that out of the four PEFT methods studied, LoRA generally produces the highest accuracy when fine-tuning LLMs on math and commonsense questions. Additionally, experiments from [12] and [13] show that with LoRA, it is possible to achieve improved fine-tuning performance using a much smaller amount of time and storage compared to traditional fine-tuning. For these reasons, we chose to focus on LoRA in the experiments for this thesis.

3.2 Fine-Tuning LLMs for Reasoning Tasks

Previous studies in fine-tuning large language models to better solve reasoning tasks provide a solid foundation for this thesis, from providing datasets to introducing new fine-tuning methods. Reasoning tasks include both open-ended assignments such as reading comprehension questions, as well as multiple-choice questions, which are the focus of this study. Table 3.1 lists some of the most closely related works to this thesis, and compares their datasets and methods to ours. We studied their datasets, dataset modifiers, and other bias mitigation methods in order to construct our own datasets and experiments.

3.2.1 Multiple-Choice Questions (MCQs)

Out of the six related works detailed in Table 3.1, four of them specify using at least one dataset consisting of multiple-choice questions ([4], [5], [8], [6]). This type of dataset lends itself easily to sampling and filtering, as demonstrated by the work in LLM-Adapters [6]. This made using multiple-choice question datasets conducive to our goal of building our own custom datasets to maximize the efficacy of fine-tuning on LLMs. We were able to randomly sample questions from the MCQ datasets while keeping the distribution, and filtered out questions that either had incorrect answers or did not match our formatting criteria.

True/False Questions

Furthermore, we decided to focus on True/False or binary-answer MCQs as the core of this study, as there were clear methods of controlling and testing various answer tokens (i.e. True/False versus A/B), the ratio of correct answer tokens in the dataset (i.e. testing a dataset where most questions are True, versus a dataset where there is an equal number of True and False answers), and ensuring equal positionings of answer tokens were included in the dataset [8].

Table 3.1: Related works on fine-tuning language models for reasoning tasks. This table lists several papers whose studies are similar in purpose to this thesis, and notes the sizes of their datasets and models, as well as whether any measures were taken to mitigate bias in the datasets or models. This thesis is included in the last row of the table as a comparison.

Related Works on LLM Fine-Tuning for Reasoning Tasks				
Study	**Dataset size(s) & type(s)**	**Dataset modifier(s)**	**Other bias mitigation method(s)**	**Model size(s)**
Anchored Answers: Unraveling Positional Bias [8]	10k-100k; MCQs	Not specified	Updated value vectors	124M-1.5B
LoRA [7]	16k-400k; open-ended text	Not specified	prefix-layer tuning, prefix-embedding tuning, adapter tuning, LoRA	1.5B-175B
Leveraging LLMs for MCQ Answering [5]	100; MC and open-ended text	Not specified	Fine-tuning on Python files	175B
LLMs are not Robust MC Selectors [4]	100; MCQs	Shuffling & removing option IDs	Not specified	7B-20B
LLM-Adapters [6]	1k-100k; MC and open-ended text and math	Sample & filter data for correctness	prefix tuning, LoRA, series adapter, parallel adapter	6B-175B
Mechanistically Analyzing Effects of Fine-Tuning [14]	2M; open-ended text	Not specified	Fine-tuning & reverse fine-tuning (reFT)	10B
Investigating Fine-Tuning for MCQs (this thesis)	40-1k; T/F MCQs	Answer token swapping, answer position swapping, answer ratio modification	Fine-tuning with LoRA	1.2B

Chapter 4

Methods and Experiments

This section discusses the LoRA version and data debiasing methods used (Section 4.1) as well as the experimental setup and resources used (Section 4.2).

4.1 Methods

4.1.1 LoRA with Conditional Generation

Section 3.1.2 and Figure 2.4 provide an overview and diagram of low-rank adaptation (LoRA). For our experiments, we used LoRA with **conditional generation**, which synthesizes the low-rank parameters (matrices A and B from Figure 2.4) based on task-specific examples and domain-specific conditions [13]. In conditional generation LoRA, a conditional diffusion model trained on a domain-specific condition (e.g. answering multiple-choice questions), called COND P-DIFF [13], generates outputs based on specified task conditions, which are then decoded into LoRA parameters specific to the task and domain. We chose this method as it is both efficient, as a variation of LoRA, and suited for adapting well to a given task domain (in our case, answering MCQs). Figure 4.1 illustrates the conditional generation process for LoRA.

4.1.2 Debiasing Data for Multiple-Choice Question Tasks

We tested different biases of LLMs when used to answer MCQs, including answer token bias, answer choice positioning bias, and answer choice ratio bias.

Answer token bias refers to the choice of tokens used to represent the answer options, for example, True/False, A/B, or 1/2. An initial concern when testing True/False questions was that the model may be biased towards one answer token or the other, so we tested different sets of answer tokens, as well as different ratios of positively labeled answer tokens, in order to investigate whether our model possesses any bias towards certain tokens.

Answer choice positioning bias refers to the positional bias that may be caused by the placement of the correct answer in multiple choice questions; for instance, presenting the correct answer as the first option may cause the model to learn a bias towards choosing the

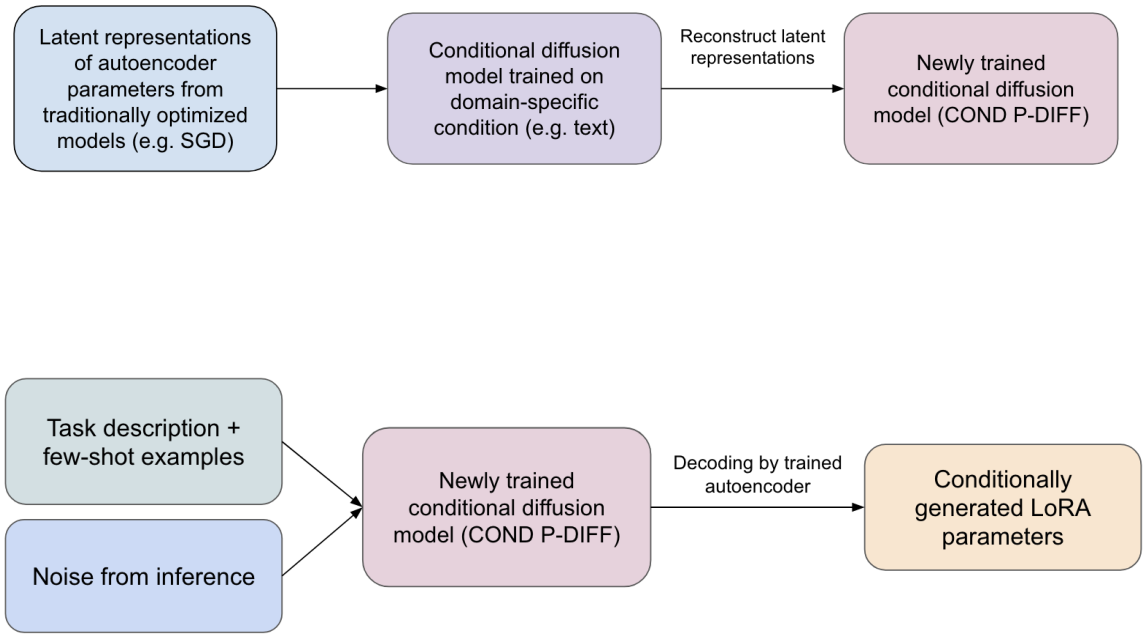

Figure 4.1: Diagram illustrating how LoRA parameters are obtained through conditional generation. A conditional diffusion model trained on a domain-specific condition, called COND P-DIFF [13], generates outputs based on specified task conditions, which are then decoded into LoRA parameters specific to the task and domain.

first answer option regardless of its correctness. To mitigate this bias, we duplicated the datasets for some of our experiments, keeping everything the same except for the order of the answer options, and observed the results.

Answer choice ratio bias refers to the ratio of answer options labeled as correct in the dataset. For example, an MCQ dataset where there is an equal number of True and False answers has a different answer choice ratio than an MCQ dataset where most of the answers are True. We experiment on multiple datasets with different answer choice ratios to determine if this parameter has any effect on the performance of the model after fine-tuning.

4.2 Experiments

4.2.1 Experimental Setup

For all of our experiments, we used the `bigscience/mT0-large` model [15], which has 1.2 billion trainable parameters and is achieved by applying a crosslingual task mixture (xP3) to the pre-trained mT5 multilanguage model. mT5 uses the same encoder-decoder architecture, pretraining objective (masked language modeling), and pretraining length (1 trillion tokens) as T5 [16]. Previous studies performed with this model have shown it to be biased towards shorter generated answers [15].

We studied the effects of various variables on the model's performance after fine-tuning on a training set of questions, including true/false answer ratio in the dataset, duplicity of original questions in the dataset, and the dataset size (number of questions in the dataset). We hypothesized that having an equal ratio of true to false answers in the dataset would lead to the best performance after fine-tuning, and that having duplicates of questions in the dataset as well as a larger dataset would both improve performance as well. Our main goal was to debias the dataset and model so that it would be equally likely to predict `True` or `False`, making it truly neutral when answering questions from our test data.

Our experiments used LoRA with conditional generation on the `bigscience/mT0-large` model, with a learning rate of 10^{-3}, 3 epochs, a batch size of 8. We used a LoRA α value of 32 and a droupout of 0.1.

Resources

Our experiments were carried out in a notebook on Google Colab, which allows roughly 13GB RAM on an Intel Xeon CPU, 15GB of RAM on an NVIDIA Tesla K80 GPU, 75GB of disk storage, and up to 3 hours of runtime for a singular experiment. We pre-processed our datasets using Python, and used Huggingface and Scikit-learn libraries for training and data analysis.

4.2.2 Datasets

We build our work off existing experiments performed with multiple-choice and open-ended questions on large language models. Work done in [6] and [8] uses various types of questions

to test the fine-tuning adaptability of LLMs, including true/false questions, reasoning-based multiple-choice questions, and both multiple-choice and open-ended math word questions.

We randomly sampled our datasets from the Commonsense170k dataset [6], which consists of 170,000 labeled True/False multiple-choice questions. After sampling, the questions and answer options for the datasets were modified to achieve the correct answer ratios desired for experimentation. Some of the smaller datasets had their True/False labels switched out for A/B labels to test token bias. Finally, two different approaches were used to create the Doubled datasets: the Doubled Swapped datasets were created by duplicating the entire base dataset and switching the order of the answer options in the duplicate, whereas the Doubled Sampled datasets were obtained by taking additional samples from Commonsense170k. We additionally created a holdout dataset with a 50:50 ratio of True/False answers to test the effects of fine-tuning our model on the all-True and all-False datasets, by randomly sampling another 250 questions from Commonsense170k. All of our datasets and their properties are listed in Table 4.1, and an example of a doubled swapped data input is shown in Figure 4.2.

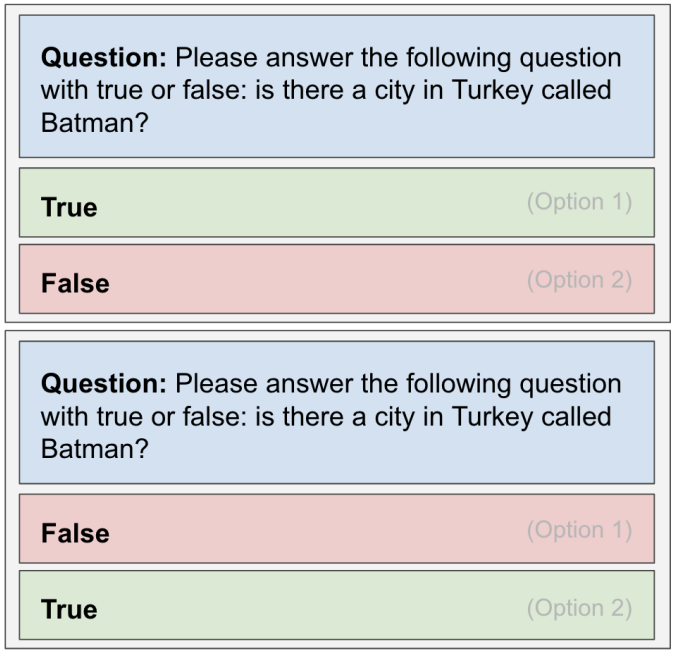

Figure 4.2: Example of True/False question from the dataset, duplicated with the answer positions swapped to mitigate positional bias.

Table 4.1: List of the datasets used in this thesis and their properties. All datasets were randomly sampled from the Commonsense170k dataset used in [6]. After sampling, the questions and answer options for the datasets were modified to achieve the correct answer ratios desired for experimentation. Some of the smaller datasets had their True/False labels switched out for A/B labels to test token bias. Finally, two different approaches were used to create the Doubled datasets: the Doubled Swapped datasets were created by duplicating the entire base dataset and switching the order of the answer options in the duplicate, whereas the Doubled Sampled datasets were obtained by taking additional samples from Commonsense170k. We additionally created a holdout dataset with a 50:50 ratio of True/False answers to test the effects of fine-tuning our model on the all-True and all-False datasets, by randomly sampling another 250 questions from Commonsense170k.

Dataset Setups			
Dataset	**Size**	**Labels**	**Ratio of Answers**
Base AB Mini	40	A/B	50:50
Base TF Mini	40	T/F	50:50
Doubled AB Mini Swapped	80	A/B	50:50
Doubled TF Mini Swapped	80	T/F	50:50
Doubled TF Mini Sampled	80	T/F	50:50
Base AB 50	250	A/B	50:50
Base TF 50	250	T/F	50:50
Base TF 0	250	T/F	0:100
Base TF 25	250	T/F	25:75
Base TF 75	250	T/F	75:25
Base TF 100	250	T/F	100:0
Doubled TF Swapped	500	T/F	50:50
Doubled TF Sampled	500	T/F	50:50
Holdout	250	T/F	50:50

Chapter 5

Experimental Analysis and Discussion

This chapter analyzes the results of our experiments and answers the research questions (Section 5.1), and then discusses the results in context of related works and ongoing research in LLM fine-tuning (Section 5.2).

We restate and answer our research questions:

- How can we reduce positional bias in language models without compromising their accuracy?
- Which characteristics of datasets result in the most meaningful changes and improvements?

To answer the first question, fine-tuning language models with datasets containing redundant data in differing positions, especially using LoRA, is a fast and efficient way to reduce positional bias without compromising accuracy. To answer the second question, we learned from our experiments that solely increasing the size of the dataset does not necessarily cause an improvement in performance, nor does this specific model react strongly to swapped answer choice tokens. The dataset characteristic that causes the most noticeable is duplicity of training data, particularly if it is presented to the model with different positional information.

5.1 Results

The results of our experiments are displayed in Table 5.1. In our experiments, each of the datasets listed in Table 4.1 underwent a randomly selected 80:20 train:test split. The pre-trained mT0-large model was loaded and fine-tuned on the training set, and then run on the test set. We recorded training and testing accuracies, as well as the training time, for each dataset. The first row of the table also displays the baseline performance of the model without fine-tuning, in comparison to its performance after fine-tuning.

Our experimental results show that Base TF 0 and Base TF 100, the datasets consisting of all False or all True answers, have the best test accuracy after fine-tuning, followed by the Doubled TF Swapped, the largest dataset using duplicated questions with their answer

Table 5.1: Table of results from performing the experiments specified in Chapter 4. Each dataset underwent a randomly selected 80:20 train:test split. The pre-trained mT0-large model was loaded and fine-tuned on the training set, and then run on the test set. We recorded training and testing accuracies, as well as the training time, for each dataset. The first row of the table also displays the baseline performance of the model without fine-tuning, in comparison to its performance after fine-tuning.

Dataset Setups				
Dataset	**Train Acc.**	**Test Acc. (in-dist.)**	**Test Acc. (holdout)**	**Train Time (min)**
Baseline	N/A	**0.00**	0.00	N/A
Base AB Mini	0.29	0.64	N/A	6
Base TF Mini	0.27	**0.67**	N/A	6
Doubled AB Mini Swapped	0.35	**0.71**	N/A	11
Doubled TF Mini Swapped	0.34	0.70	N/A	12
Doubled TF Mini Sampled	0.26	0.60	N/A	14
Base AB 50	0.28	0.70	N/A	41
Base TF 50	0.27	**0.71**	N/A	44
Base TF 0	0.42	0.84	0.45	40
Base TF 25	0.21	0.42	N/A	39
Base TF 75	0.14	0.40	N/A	37
Base TF 100	0.40	0.80	0.42	43
Doubled TF Swapped	0.25	**0.74**	N/A	85
Doubled TF Sampled	0.19	0.47	N/A	96

option positions swapped. Upon analysis, it makes sense that the datasets with only one answer token would perform "well", as the model learns to only predict one answer token and that same answer token is present in all the test data from the train:test split. The additional tests with the holdout dataset show that the fine-tuned model performs less well on a more general distribution.

The second result concerning Doubled TF Swapped also aligns with our hypothesis that duplicating the dataset samples and swapping the answer positions would result in a less biased model. Some of this increased accuracy may also stem from the train and test sets containing some of the same questions, making it more likely for the language model to predict correctly. Notably, we get similar performance from fine-tuning on Base TF 50 as the Double Mini Swapped datasets. This is significant as roughly a third of the amount of data is used in the latter compared to the former, meaning that using data wisely can help reduce training time and resources.

5.1.1 Impact of Dataset Contents

Observing the Baseline row of Table 5.1, we see that having some sort of MCQ fine-tuning data that has the same distribution and characteristics as the test data does improve per-

formance. There is no clear impact from changing dataset size alone; the method used to increase dataset size matters much more, as evidenced by the results of datasets Doubled TF Swapped and Doubled TF Sampled.

5.1.2 Impact of Dataset Bias Mitigation Measures

From our experiments testing different ratios of True and False correct answers, we can infer that balancing the True/False ratios in a dataset of True/False MCQs does indeed improve test accuracy. The datasets consisting of purely True or False answers can be seen as outliers, as the language model learns to predict the exact same thing and ends up being correct as there is absolutely no variation in the dataset.

5.1.3 Statistical Significance

The experiments using smaller datasets, namely Baseline, Base AB Mini, Base TF Mini, Doubled AB Mini Swapped, Doubled TF Mini Swapped, and Doubled TF Mini Sampled, were each run 6 times to reduce variance. The test accuracy variances for each of these experiments, respectively, were 0, 0.001, 0.003, 0.003, 0.002, and 0.004. These values suggest that the results for the smaller experiments are not too noisy, and future work gathering variances for the larger experiments could further solidify our results.

There were several sources of randomness in our experiments, including randomly sampling smaller datasets from Commonsense170k and generating the train:test splits. Fine-tuning the model and generating responses after fine-tuning further amplify any dataset randomness. Because of this, we would need to repeat these experiments a much larger number of times in order to gain a measure of statistical significance.

5.2 Discussion

5.2.1 Comparison to Related Works

The results of our experiments show that our bias mitigation methods are on par with other mitigation methods used in related works (Table 3.1). This is perhaps to be expected, as we applied a combination of several techniques used in previous research, such as LoRA and answer position swapping.

5.2.2 Limitations

Other Datasets and Models

We were also interested in learning the impact of other reasoning tasks when used for fine-tuning LLMs, such as the open-ended arithmetic reasoning questions from [6] or the reading comprehension tasks from [5]. However, these types of questions did not lend themselves easily to the conditional generation LoRA approach of our experiments. Additionally, we would have liked to fine-tune on larger datasets and models, but these did not fit into the

constraints set by us to run in Google Colab, as mentioned in Section 4.2.1. Additionally, the smaller size of our datasets meant that any inconsistencies in sampling were magnified and possibly impacted our results.

Other Mitigation Methods

There are also other bias mitigation methods that we did not pursue in this thesis. One example is in-context learning, in which the model itself is unchanged, but the prompts are engineered to maximize test accuracy. We believe that combining our dataset modification approaches with prompt engineering may lead to even better performance after fine-tuning.

Chapter 6

Conclusion

In this thesis, we studied the effect of fine-tuning on LLM accuracy in answering multiple-choice questions (MCQs). We achieved this by analyzing the properties of the datasets used, including the ratios of correct answers, tokens used for answer options, and duplicity of samples in datasets. We constructed datasets to test the impact of each of these properties on the final language model performance after fine-tuning, tracking the effects of different question types, dataset sizes, answer ratios.

At the start of this thesis, we asked two primary research questions:

- How can we reduce positional bias in language models without compromising their accuracy?
- Which characteristics of datasets result in the most meaningful changes and improvements?

We were able to answer these research questions with the following discoveries:

- Fine-tuning language models with datasets containing redundant data in differing positions, especially using LoRA, is a fast and efficient way to reduce positional bias without compromising accuracy.
- The dataset characteristic that causes the most noticeable is duplicity of training data, particularly if it is presented to the model with different positional information. Solely increasing the size of the dataset does not necessarily cause a significant improvement in performance, nor does this specific model react strongly to swapped answer choice tokens.

Throughout the course of our research, we also made the following contributions:

- We verified the ability of LoRA to fine-tune LLMs to reduce positional bias efficiently without compromising accuracy.
- We created curated datasets by focusing on properties such as doubling the MCQs used with different answer ordering, balancing the ratios of correct answers, and using different sets of answer tokens.

- We ran empirical experiments to mitigate bias in LLM performance on MCQs through fine-tuning on these datasets. Our experimental results showed us that having datasets with higher sample duplicity, with more balanced answer options maximizes the efficacy of fine-tuning to improve LLM performance.

6.1 Future Work

This thesis pays attention to and actively studies properties of datasets used for fine-tuning LLMs, such as MCQ answer token bias, answer positioning bias, and answer frequency bias. We hope to conduct similar studies to this thesis, that build off the related works detailed in Chapter 3, in order to either solidify those results or make new discoveries.

Possible future studies stemming from this thesis could include using these datasets to perform fine-tuning with more than one of the adapters mentioned in [6], as using multiple adapters simultaneously empirically improves LLM performance more than using just one (in our case, LoRA). We could also observe the effects of using more epochs during training.

Additionally, we could repeat this experimental process with other types of datasets and models to better understand our current results in a larger context, as well as creating and experimenting on more granular variants of our own MCQ datasets. There is also work to be done to reduce possible confounding factors affecting test accuracy in Doubled Swapped datasets, as suggested in Section 5.1. Furthermore, we could test additional dataset parameters and splits, experiment on more granular ratios of True/False answers, or try out other answer tokens in addition to True/False and A/B. Finally, we could combine the bias mitigation methods used in this thesis with other mitigation approaches, such as prompt engineering, to achieve even better accuracies for LLM performance on MCQs.

References

[1] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian. *A Comprehensive Overview of Large Language Models*. 2024. arXiv: 2307.06435 [cs.CL].

[2] I. Kremer. *Towards Better Understanding of Positional Bias in Transformer-based Language Models*. 2024.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[4] C. Zheng, H. Zhou, F. Meng, J. Zhou, and M. Huang. *Large Language Models Are Not Robust Multiple Choice Selectors*. 2024. arXiv: 2309.03882 [cs.CL].

[5] J. Robinson, C. M. Rytting, and D. Wingate. *Leveraging Large Language Models for Multiple Choice Question Answering*. 2023. arXiv: 2210.12353 [cs.CL].

[6] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee. *LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models*. 2023. arXiv: 2304.01933 [cs.CL]. URL: https://arxiv.org/abs/2304.01933.

[7] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[8] R. Li and Y. Gao. "Anchored Answers: Unravelling Positional Bias in GPT-2's Multiple-Choice Questions". In: *ArXiv* abs/2405.03205 (2024). URL: https://api.semanticscholar.org/CorpusID:269605110.

[9] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[10] L. Bereska and E. Gavves. *Mechanistic Interpretability for AI Safety – A Review*. 2024. arXiv: 2404.14082 [cs.AI].

[11] N. Elhage et al. "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread* (2021). https://transformer-circuits.pub/2021/framework/index.html.

[12] J. Zhu, J. Lin, X. Dai, B. Chen, R. Shan, J. Zhu, R. Tang, Y. Yu, and W. Zhang. *Lifelong Personalized Low-Rank Adaptation of Large Language Models for Recommendation*. 2024. arXiv: 2408.03533 [cs.IR]. URL: https://arxiv.org/abs/2408.03533.

[13] X. Jin, K. Wang, D. Tang, W. Zhao, Y. Zhou, J. Tang, and Y. You. *Conditional LoRA Parameter Generation*. 2024. arXiv: 2408.01415 [cs.AI]. URL: https://arxiv.org/abs/2408.01415.

[14] S. Jain, R. Kirk, E. S. Lubana, R. P. Dick, H. Tanaka, E. Grefenstette, T. Rocktäschel, and D. S. Krueger. *Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks*. 2023. arXiv: 2311.12786 [cs.LG]. URL: https://arxiv.org/abs/2311.12786.

[15] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. Le Scao, M. S. Bari, S. Shen, Z. X. Yong, H. Schoelkopf, et al. "Crosslingual Generalization through Multitask Finetuning". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2023, pp. 15991–16111.

[16] 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". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.