Layer Select LoRA: Enhancing LLM Factual Accuracy through Selective Layer Adaptation

Siavash Nejadi
snejadi3@gatech.edu

Shima Rajaei Dehkordi
sdehkordi6@gatech.edu

Ryan J. Stonick
rstonick3@gatech.edu

Abstract

Large language models (LLMs) often generate hallucinated or factually incorrect responses, limiting their reliability in real-world applications. This project investigates the use of parameter-efficient layer select fine-tuning via Low_Rank Adaptation (LoRA) on the Mistral-7B language model, with the goal of imrproving factual question answering performance. Based on the idea that different layers specialize in different functions. We trained the model on the TriviaQA dataset using LoRA and evaluated the resulting model on both in-domain TriviaQA and out-of-domain TruthfulQA datasets. We will adapt the model to provide more truthful, verifiable responses. This lightweight fine-tuning approach makes it a practical and accessible method to improve the truthfulness and factual reliability of LLMs.

1. Introduction, Background and Motivation

In this project, the objective is fine-tuning Mistral 7B [7] language model using different layer selective techniques and Low-Rank Adaptation or LoRA [6] to increase its ability to generate factual information. Full fine-tuning of an LLM is challenging, and fine-tuning will become less feasible without access to huge amounts of computing resources. Mistral 7B has 7.3 billion parameters and fine-tuning of all parameters is computationally expensive and beyound the scope of this project. Instead we use LoRA, which freezes the pretrained model weights and injects trainable rank decomposition into each layer of the transformer, and significantly reduces the number of trainable parameters. Here, we further improve the LoRA implementation by focusing the adaptation on the attention heads within the model and specific layers within the feedforward multilayer perceptrons in an attempt to focus on fine-tuning just the location where factual information is represented in the model. This reduces the need for large computation resources. We sought to achieve this by training the model on the TriviaQA dataset [8]. To better understand why Minstal 7b and other attention-based models sometimes hallucinated factually inaccurate information we aimed to evaluate how much the trained model is prioritizing producing fluent and convincing language over factual accuracy. This is done by testing the trained model on TruthfulQA dataset [11] which will be explained in more details in Subsection 2.1.

Based on the current state-of-the-art in LLM fine-tuning, which progressed from the original LoRA's rank decomposition matrices [6] to QLoRA's memory-efficient quantization techniques [2] and LongLoRA's context length extensions [1], our proposed "Layer Select LoRA" project addresses a novel and unexplored direction in the field. While existing approaches focus on memory efficiency (QLoRA) and context length (LongLoRA), our project uniquely targets the enhancement of factual accuracy through selective layer adaptation. In our approach, we identify and fine-tune specific "fact-processing" layers within transformer architectures to develop efficient language models. This targeted strategy enables us to maintain strong performance while remaining computationally feasible on limited hardware resources.

2. Approach

In this project, we fine-tuned the Mistral-7B language model using LoRA (Low-Rank Adaptation) on the TriviaQA dataset to explore how lightweight fine-tuning methods can efficiently adapt large language models. We formatted the dataset into question-answer pairs and applied tokenization suitable for Mistral-7B. Using **PyTorch** [13] as our deep learning framework, we fine-tuned a small subset of Mistral's 7 billion parameters through LoRA, significantly reducing computational and memory requirements compared to full-model fine-tuning. The number of trainable parameters in LoRA depends on the number of target modules—specifically W_q, W_k, W_v, and W_o, which are the query, key, value, and output projection matrices in the self-attention module—as well as the rank of the adaptation and the depth of the model. In some configurations, the number of trainable parameters can be as low as 0.01% of the full pre-trained Mistral-7B model. According to the LoRA paper [6], the number of trainable parameters can be estimated

as:

$$|\Theta| = 2 \cdot L_{\text{lora}} \cdot d_{\text{model}} \cdot r$$

As part of this study, we aimed to understand low-rank updates and how they work. To do so, we conducted a series of experiments to explore the following questions:

- Which subset of weight matrices in a pre-trained transformer should be adapted to maximize model performance?
- What is the optimal rank value that avoids overfitting while maintaining strong performance?
- How does this fine-tuning method affect the model's performance on the TruthfulQA dataset?
- Does strong performance on TriviaQA necessarily indicate good performance on the TruthfulQA dataset?
- How does tuning different layers such as transformer or MLP affect the performance of the model on these datasets?

In addition to the aforementioned research, our investigation systematically varied LoRA configurations to assess their impact. Primarily here we targeted the MLP (Feed-Forward Network) layers (`gate_proj`, `up_proj`, `down_proj`) for adaptation, exploring different ranks (r) such as 8, 16, 32, and 64, often adjusting the LoRA scaling factor (α) accordingly.

To better understand the contributions of different components to the overall performance of the model, we performed a series of ablation studies [14, 16, 9]. The methodology involves systematically applying LoRA to specific subsets of these MLP layers and observing the resulting changes in the model's performance on evaluation datasets. By analyzing the impact, we can identify which modules are critical for maintaining high factual accuracy and truthfulness, and which have a negligible effect. Furthermore, we looked at the effect applying LoRA adaptations selectively across different layer ranges within the model: early layers (0-9), middle layers (10-21), and late layers (22-31). Additional variations included adjustments to dropout rates, learning rates, dataset subset sizes, and training epochs. Most of the hardware used to facilitate the hundreds of runs we iterated through was accessed through the PACE-ICE cluster provided to us by Georgia Institute of Technology. The high performance cluster (HPC) is run using SLURM so we could submit our models for training in job array batches of up to 28. This resource was fantastic for our use case but also required quite a steep learning curve that was an important part of our experiment to push limited resource access to it absolute limit.This challenge underscored the realities of modern deep learning implementation, where significant compute resources are often required. It pushed us to refine our model building and training techniques under these constraints, forcing careful experimentation with parameters like batch sizes, maximum token sequence lengths during training, and dataset subset sizes to balance performance with computational feasibility. The GPUs used to train consisted mostly of a Nvidia H200, Nvidia RTX 6070, and RTX 6000. Our codebase underwent several iterations aimed at maximizing GPU efficiency. Since the primary goal of this project was to demonstrate that large language models can be optimized without access to extensive compute resources, we dedicated significant effort to performance tuning. To enable distributed training across multiple GPUs, we integrated the Hugging Face accelerate library [3], allowing us to leverage the capabilities of the H200 GPU to their fullest. While this setup worked reliably on a single GPU, approximately 50% of our multi-GPU training runs failed due to a known issue with HPC communication in the NVIDIA Collective Communications Library (NCCL), which remained unresolved despite extensive debugging and optimization efforts. We also utilized portions of existing code from the TriviaQA and TruthfulQA GitHub repositories for running benchmarks and evaluations. Given our focus on fine-grained hyperparameter tuning and its effect on model performance, we did not implement our own adapters. Instead, we used the LoRA-specific adapters provided by Hugging Face's Parameter-Efficient Fine-Tuning (PEFT) library [4].

2.1. Datasets

TriviaQA [8] is a large-scale reading comprehension dataset designed for training and evaluating language models. It contains over 650,000 question-answer-evidence triples, with questions crafted by trivia enthusiasts and evidence documents collected independently — averaging about six documents per question. The dataset also includes a broader set of 950,000 question-answer pairs linked to approximately 662,000 documents from Wikipedia and the open web. TriviaQA presents a realistic and challenging setting compared to traditional benchmarks, in which the answers are often not simple spans in the text and require reasoning over long, noisy contexts. It features both human-verified and automatically collected subsets, and is valuable for fine-tuning models to handle real-world question answering tasks.

TruthfulQA [11] is a benchmark designed to evaluate the truthfulness of language model outputs when answering questions. It consists of 817 carefully constructed questions in 38 categories, including health, law, finance, and politics, with many questions that target common human misconceptions or false beliefs. To succeed on TruthfulQA, models must not simply imitate human-written text but must actively avoid generating misleading or incorrect answers. Initial evaluations on models like GPT-3, GPT-Neo/J, GPT-

2, and T5 variants showed that the best model achieved only 58% truthfulness, compared to 94% for humans [11]. In particular, larger models tended to be less true, highlighting that scaling alone does not guarantee improved factual accuracy, suggesting that fine-tuning strategies focused on truthfulness are necessary for future improvements.

2.2. Data Preparation

As described in the previous section, we used the TriviaQA dataset for training purposes. TriviaQA contains approximately 90,000 training samples, 11,000 validation samples, and 11,000 test samples. In this study, we primarily selected a subset of 2,000 samples from the training set to fine-tune the model. In certain experiments, to assess the impact of training data size on model performance, we increased the subset size to between 15,000 and 20,000 samples.

The TriviaQA dataset was preprocessed by extracting question-answer pairs. Each sample was then reformatted into a prompt–completion structure suitable for instruction-style fine-tuning, where the question served as the prompt and the answer as the target output. All text inputs were tokenized using the Mistral-7B tokenizer, with appropriate truncation and padding applied to meet the model's maximum sequence length requirements

2.3. Evaluation Metrics

We employed a range of evaluation metrics at different stages of the project to assess model performance and guide training decisions.

During the fine-tuning phase on TriviaQA, we monitored validation perplexity to track learning progress. This helped us tune hyperparameters and detect signs of overfitting. We also observed both training and validation loss curves to evaluate model generalization and training stability.

For the final evaluation on the TriviaQA test set, we used the F1 score to assess the model's accuracy in generating correct answers. The F1 score was chosen because it balances precision and recall—two key measures in evaluating generation tasks. Precision indicates how much of the model's output is relevant to the reference answer, while recall measures how much of the correct answer was captured by the model's output.

To evaluate performance on the TruthfulQA dataset, we used BLEU [12], ROUGE [10], and multiple-choice (MC) accuracy. While BLEU was originally developed for machine translation, it is also applicable to question-answering tasks. It measures n-gram overlap between the generated and reference answers, with scores ranging from 0 to 1—higher scores indicate closer matches in word sequences. Similarly, ROUGE, commonly used for summarization, evaluates the overlap of lexical units between outputs and references. Like BLEU, its score ranges from 0 to 1, with higher values reflecting better performance.

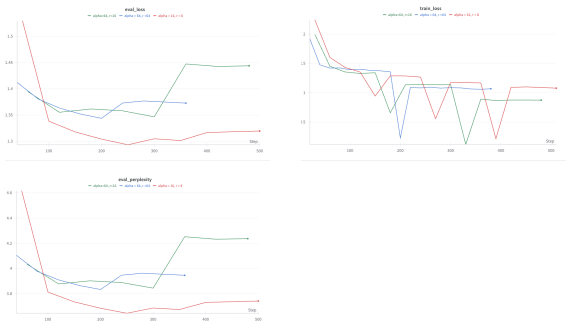

Figure 1: Evaluation Results

In addition, we used MC accuracy from the TruthfulQA benchmark, which evaluates whether the model selects the most truthful answer from multiple options. This metric directly measures the model's ability to distinguish factually correct information from plausible but incorrect alternatives.

The above metrics provide a comprehensive view of the model's performance, capturing both linguistic quality and factual accuracy.

2.4. Quantization

The original LoRA application does not inherently involve quantization, but it can be combined with quantization techniques to further reduce the computational and memory footprint of a model [2]. Using quantization, the model operates with lower precision, which complements the parameter efficiency achieved by LoRA. In QLoRA, quantization is directly integrated into the adaptation process. This means that the low-rank matrices introduced by LoRA are quantized, allowing for even more significant reductions in memory usage and computational requirements. This is particularly useful for deploying models on devices with limited resources. We implemented 4-bit quantization in our studies, and ran a few cases with 8-bit quantization, and without any quantization to compare the performance of the models and evaluate the effectiveness of quantization.

3. Experiments and Results

In this section, we present a series of experiments conducted to evaluate and improve the performance of our model on the TruthfulQA benchmark. We begin by exploring the effect of varying the LoRA rank parameter to identify an optimal configuration. We then investigate how adapting different weight metrics influences multiple-choice accuracy, followed by an analysis of fine-tuning MLP layers and their impact on both training and testing outcomes. We further examine the role of quantization in reducing memory usage while maintaining performance,

and analyze the behavior of training loss and gradient norms across these experiments to gain insights into the optimization dynamics.

3.1. Experiment 1: Optimal rank value

In this experiment, we focused on investigating the impact of the rank parameter r on model performance. We trained two models with identical hyperparameters but different values of r and the LoRA scaling factor α. The results for training loss, validation loss, and evaluation perplexity are presented in the figures below. The shared hyperparameters across the models were: batch size of 96, 3 training epochs, and a sequence length of 512 tokens. Both models were fine-tuned on the same target modules. As shown in the figures 1, increasing the rank from 16 to 64 led to a noticeable rise in validation loss and evaluation perplexity, indicating that the model was overfitting to the training data. Additionally, the model corresponding to the red plot—which used the lowest rank and alpha values—demonstrated the best generalization performance among all configurations.

Increasing the rank r in LoRA effectively increases the number of trainable parameters in the model. A higher rank makes the model more complex and that is why we see constant decrease in training error. Since the dataset size that we used in this experiment is relatively small, the capacity of the model is more than what our data could support, as a result, the model started to increase. When both r and α are increased, the model is experiencing overfitting even more. Therefore, in our study, having r values between 4 to 16 and maximum α value of 32 was sufficient.

3.2. Experiment 2: Impact of fine Tuning on self-attention modules

In this experiment, we evaluated the effect of fine-tuning different target weight modules within the attention mechanism on the model's performance on the TruthfulQA dataset. We measured performance using multiple-choice accuracy (MC acc), ROUGE, and BLEU scores. These results are displayed in Table 1. Our results show that applying LoRA to the **Wq** and **Wk** modules individually led to moderate improvements over the base model across all three evaluation metrics. Notably, adapting **Wq** alone yielded the highest multiple-choice (MC) accuracy (0.5116) and BLEU score (0.6712). In contrast, applying LoRA to the **Wv** module—either alone or in combination with others—consistently degraded performance across all metrics. This suggests that **Wv** may be less robust to parameter adaptation in this context and could introduce noise or instability during generation. In the transformer attention mechanism, the **Wq** (query) and **Wk** (key) matrices determine what the model attends to—that is, which parts of the input it considers important. Fine-tuning these weights helps the model

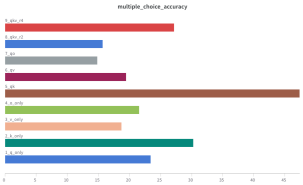

Figure 2: Multiple choice accuracy tested on Truthful QA after applying LoRA to different types of attention weights in Mistral 7B model, given the same number of trainable parameters

learn more effective attention patterns, which can enhance reasoning and fact retrieval. This likely explains the improved performance on the TruthfulQA dataset. In contrast, the **Wv** (value) matrix influences what content is extracted after attention is applied. Modifying these layers can significantly impact the semantics of the model's output.

3.3. Experiment 3: Evaluating the Impact of MLP Layer Fine-Tuning on Training and Test Performance in TruthfulQA

These experiments focused on investigating the impact of fine-tuning only the Multi-Layer Perceptron (MLP) components of the Mistral-7B-v0.1 model using Low-Rank Adaptation (LoRA).

The inspiration of this experiment was novel research and approaches [15] [5] seen in the state of the art where knowledge storage and retrieval is shown to have complex trade off between attends and MLPs with factual knowledge theorized to be stored in the MLP network within LLMs.

Most of the experiments done in this section followed the overall trend we implemented of making variations of the on rank and alpha but the primary goal was to assess how adapting these specific layers, responsible for feature transformation within the network, affected the model's performance, using TriviaQA as the fine-tuning dataset. Success during training was primarily monitored via training loss, and for later experiments (Runs 23-26), we implemented early stopping based on validation loss on a subset of the TriviaQA data.

The way we measured measure of success involved evaluating the fine-tuned adapters on downstream tasks, specifically TriviaQA (using EM/F1 scores) and TruthfulQA (using multiple-choice accuracy), comparing them against the base model's performance. Our experimental design involved systematically varying LoRA configurations: testing different ranks (r=8, 16, 32, 64) and alpha values across all MLP layers (`gate_proj`/`up_proj` or only `down_proj`), targeting MLP layers within specific block ranges (early, middle, late), and isolating subsets of the MLP modules (e.g., only `gate_proj`/`up_proj` or only `down_proj`).

However, the evaluation results 2 shows that none of our

Weight type	base	Wq	Wk	Wv	Wo	Wq, Wk	Wq, Wv
Rank (r)	-	8	8	8	8	4	4
MC acc	0.4186	0.5116	0.4871	0.1554	0.2693	0.4731	0.1946
ROUGE	0.6179	0.6717	0.6535	0.4585	0.4911	0.6321	0.4513
BLEU	0.5987	0.6712	0.6393	0.4065	0.4641	0.6514	0.3912

Table 1: Multiple-choice accuracy results on the TruthfulQA dataset

MLP-only fine-tuning configurations managed to outperform the base Mistral-7B model on the downstream tasks. This outcome confirmed our developing understanding, as we worked on this project, that simply targeting the hypothesized knowledge storage layers (MLPs) isn't sufficient for improving performance on complex tasks like TriviaQA and TruthfulQA. And that there is infact a need to have a balanced approach when fine tuning layer. There is certainly evidence that tuning those layers is valuable but doing so alone is insufficient.

So *only* the MLP layers, despite their theorized role in factual recall, is insufficient. The attention mechanism is paramount to actually retrieving that information by attending to the relative importance it has with the prompt. Thus to get overall performance gains requires coordinated adjustments across both MLP and attention mechanisms, suggesting that knowledge retrieval and reasoning are more distributed processes than initially targeted. We believe that the poor performance on TruthfulQA after fine-tuning on TriviaQA strongly supports the hypothesis that MLPs do, in fact, act as key knowledge repositories. By fine-tuning the MLPs specifically on TriviaQA data, we likely did store effectively reinforced the storage of factual information relevant to that dataset. However, this information, while "correct" in the context of trivia, proved unhelpful or even harmful when evaluated on TruthfulQA, a benchmark sensitive to common misconceptions and nuanced truth. In essence, we modified the model's knowledge base (the MLPs) with information that, while factual for trivia, degraded its ability to navigate the complexities of truthfulness required by TruthfulQA, confirming the critical role of MLPs in knowledge storage and the potential negative consequences of narrowly focused fine-tuning.

3.4. Experiment 4: Quantization

The baseline model without quantization ("None") achieves the best overall performance, exhibiting the lowest loss (1.45), lowest perplexity (3.53), and highest F1 score (56.26). When applying 8-bit quantization, there is a slight degradation in performance, indicating that 8-bit quantization preserves model quality reasonably well with only minimal trade-offs compared to the none-quantized model. In contrast, 4-bit quantization results in a more substantial performance decline, suggesting that it introduces noticeable degradation in the model's predictive capabilities, affecting both loss and F1 score.

3.5. Behavior of training loss, and gradient norm in different experiments

Figure 3 presents training loss, and gradient norm for two sample training runs, referred to as Case 1 and Case 2. The objective is to analyze and compare the fine-tuning behavior under different LoRA configurations. In Case 1, LoRA adaptation was applied to a limited subset of transformer layers—specifically a few modules from `q_proj`, `k_proj`, `v_proj`, and `o_proj`. In contrast, Case 2 targeted a wider range of modules, including `q_proj`, `k_proj`, `v_proj`, `o_proj`, as well as additional layers `gate_proj`, `up_proj`, and `down_proj`. This expanded coverage in Case 2 allowed the model to adjust a larger portion of the network, enabling more flexible and effective adaptation during fine-tuning. The effects of these configurations are reflected in the training dynamics discussed in the following.

In Case 1, the model shows good early learning, with a rapid initial decrease in training loss. However, after the first 20–50 steps, the loss plateaus and changes only gradually over the next 300 steps. Throughout this period, the gradient norm remains steady but relatively low, fluctuating between 2 and 2.5 — a sign that the model continues to update, but with smaller steps. This behavior typically indicates that the model has quickly learned the easier patterns in the data and is now struggling with more difficult examples. Although training remains overall healthy, convergence is slow. To achieve further improvements, it may be necessary to continue training longer, apply a more aggressive learning rate decay, or adjust the optimizer settings.

In Case 2, the significant drops in training loss observed at around 100 and 220 steps suggest that the optimizer discovered important new regions in the loss landscape or that the LoRA layers identified shortcuts to better minimize the loss, possibly by uncovering new patterns in the data. The gradual decay of the gradient norm throughout training indicates that the model is steadily converging. Compared to Case 1, the gradient norm in Case 2 starts much higher (7.5 versus 2–2.5), reflecting larger parameter updates early in training. Overall, this pattern is a strong sign of healthy training dynamics: the model is effectively learning, adapt-

Target Module	MC Score	Alpha	Rank
Wq+Wk	.4736	32	8
Minstral 7B Base	.399	-	-
Wk	.302	32	8
MLP all gates	.280	128	64
Wq+Wk+Wv	.271	32	4
MLP Middle Layers [10,21]	.264	32	32

Table 2: Top 5 module performance on TruthfulQA MC.

Quantization Type	Loss	Perplexity	F1 Score	Allocated GPU Memory
None	1.45	3.53	56.26	15 GB
4-bit	1.64	5.15	54.14	5 GB
8-bit	1.50	4.50	56.14	8 GB

Table 3: Performance metrics under different quantization configurations.

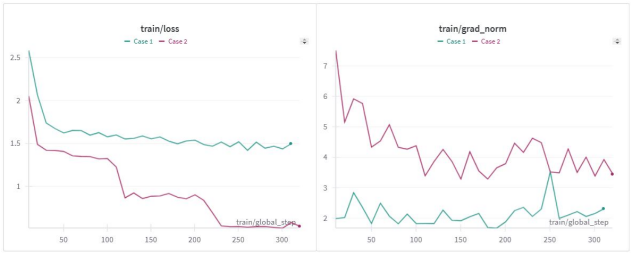

Figure 3: Training loss and gradient norm over the course of fine-tuning for two sample runs: Case 1 (limited LoRA modules) and Case 2 (expanded LoRA modules).

ing mid-training, and optimizing its internal representations in a meaningful way.

3.6. Conclusion

In this study, we utilized the PyTorch framework to investigate parameter-efficient fine-tuning of large language models using LoRA, with a focus on quantization and adaptation strategies.

- We demonstrated that LoRA can efficiently adapt large language models like Mistral-7B using minimal resources. By updating only 0.02% to 0.5% of parameters, we made fine-tuning feasible even on limited hardware, primarily applying 4-bit quantization for memory efficiency and faster inference.

- Our ablation studies showed that adapting a small subset of transformer layers leads to steady learning, while broader adaptations yield faster loss reduction and more dynamic learning behavior. In particular, adapting **Wq/Wk** improves attention control and generalization, whereas modifying **Wv** may increase the risk of hallucinations—especially with limited training data—by altering the extracted information.

- Fine-tuning on TriviaQA improved answer retrieval but did not translate to better performance on TruthfulQA. In fact, it made the model more fluent but also more confidently wrong, as TruthfulQA is designed to expose false beliefs. This highlights a trade-off between retrieval efficiency and robustness to adversarial questions, underscoring the importance of dataset alignment with evaluation goals.

- We performed comprehensive ablation studies to better understand the contributions of different LoRA components. In terms of learning behavior and model health, models with a limited subset of transformer layers demonstrate steady early learning with gradual convergence, while models targeting a broader range of modules exhibit more dynamic learning with significant loss reductions, indicating more effective adaptation through expanded LoRA configurations.

- We conducted preliminary experiments to evaluate the impact of quantization on overall model performance. However, further in-depth analysis is needed to fully understand its effects. Throughout the study, we applied 4-bit quantization primarily for memory optimization and faster inference, while balancing the associated performance trade-offs.

4. Work Division

The delegation of work among team members is provided in Table 4

References

[1] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-

Student Name	Contributed Aspects	Details
Siavash Nejadi	Code preparation, Hyperparameter tuning, Analysis, Report writing	Prepared the codebase and configurations, implemented and fine-tuned the model, analyzed results, contributed to report writing.
Shima Rajaei Dehkordi	Research, Coding, Implementation,Analysis, Report writing	Conducted Research- Implementation- code improvement- Training and fine ftuning- analysis and report preparation.
Ryan Stonick	PACE-ICE implementation and optimization, Code, Hyperparameter Tuning Research, Report	Worked with HPC cluster and GPU/resource management, wrote code to optimize performance and analysis, hyperparameter tuning and iterative refinement, and report writing

Table 4: Contributions of team members.

tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023. 1

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. 1, 3

[3] Hugging Face. Accelerate documentation. https://huggingface.co/docs/accelerate, 2024. 2

[4] Hugging Face. Peft documentation. https://huggingface.co/docs/peft, 2024. 2

[5] Constanza Fierro, Negar Foroutan, Desmond Elliott, and Anders Søgaard. How do multilingual language models remember facts?, 2025. 4

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1

[7] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. 1

[8] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, page arXiv:1705.03551, 2017. 1, 2

[9] Chethan Kadavath, Jared Casper, Roger Grosse, and et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022. 2

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. 3

[11] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022. 1, 2, 3

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002. 3

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019. 1

[14] Ethan Perez, Sam Ringer, Marco Tulio Ribeiro, Sharan Narang, Jason Phang, Samuel Bowman, Colin Raffel, and Deep Ganguli. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022. 2

[15] Yiqun Wang, Sile Hu, Chaoqun Wan, Yonggang Zhang, Xiang Tian, Yaowu Chen, Xu Shen, and Jieping Ye. Deciphering and enhancing commonsense reasoning in LLMs from the perspective of intrinsic factual knowledge retrieval, 2025. 4

[16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. 2