Assessing the Vulnerability of LoRA-Based Fine-Tuning to Data Poisoning Attacks

William Burriss
Virginia Tech

Alex Klee
Virginia Tech

Rajith Pandeti
Virginia Tech

Abstract

Our goal is to evaluate which LoRA based Parameter Efficient Fine-tuning methods are the most vulnerable to data poisoning attacks. Using the BeaverTails QA data set we will fine-tune Meta's Llama Large Language Model using LoRA, QLoRA, Delta LoRA, and Prompt Tuning. To simulate a realistic data poisoning attack we will reduce the amount of harmful data from the BeaverTails data set such that the it will make up around 10% of the final training data. Each of these models will be evaluated using a standardized evaluation dataset along with methods to quantify the results such as the ROUGE score.

1 Motivation and Problem Statement

In recent years the field of AI has seen many breakthroughs especially in regards to Large Language Models (LLMs). Today, we have seen numerous companies integrating LLMs into their products. We have also witnessed powerful open source models being released as well, such as Meta's Llama [6]. These open source models make it easy for small companies or hobbyist developers to fine tune Large Language Models for specific purposes. One very popular method for fine tuning is Parameter Efficient Fine-tuning. This allows for the fine-tuning of LLMs without the computationally expensive task of altering every parameter.

Often times when fine-tuning a Large Language Model, an openly sourced data set is used in the training process. Websites like HuggingFace [2] and Kaggle [5] make it very easy for anyone to share and use these datasets. However, given how large these datasets are the problem of possible data poisoning arises. Data poisoning is the malicious act of intentionally altering the data within a dataset to contain false, biased, or harmful information [1]. Many data sets contain hundreds of thousands of entries. This makes it almost impossible for small teams to verify the integrity of datasets which they might use. Thus, it is important to evaluate how susceptible these commonly used LoRA based methods for fine-tuning are.

2 Related Work

During our research we were able to locate two papers, [4] and [8], which both test the robustness of Natural Language Generation models against data poisoning attacks. These papers both provided insight on possible evaluation methods and possible challenges we might encounter. Despite the similarities between these papers and our project there are some important differences. For one, our project will be solely evaluating the effectiveness of LoRA based fine tuning to withstand data poisoning attacks. Secondly, we plan on using an evaluation model such as the BeaverDam-7B evaluation model to quantify the toxicity of our models responses. This is unlike any of the methodologies described these two papers.

In [4] the authors analyze the effect of data poisoning attacks on LLMs using both full fine-tuning and PEFT-based fine-tuning. The primary PEFT method used is prefix tuning and the fine-tuning effectiveness is tested on two natural language generation(NLG) tasks: text summarization and text completion.

They evaluate effectiveness through stealthiness, or difficulty in the attack's influence being detected while evaluating, and success of the attack. For text summarization, the Clean ROUGE score is used and for text completion Clean Perplexity is used. A high rouge score and a low perplexity score are signs of a stealthy attack.

The results displayed that increasing the percent of poisoned training data improves success of attack, where the trigger is inserted can change the effectiveness of an attack, and for text summarization PEFT methods are more resilient against attacks, while for text completion full fine-tuning is more effective.

This paper provides valuable information on the process of poisoning LLMs, as well as methods to poison, such as different trigger positioning. Although it differs from

our project because of our focus on the LoRa family specifically, the paper displays useful results in evaluating the effectiveness of the PEFT method in NLG tasks. We could incorporate the findings into our project by expanding our ways of attack on the LoRa based PEFT methods to further test their effectiveness.

In [8] the authors address how backdoor attacks, specifically syntactic and triggerless backdoor attacks, affect NLG tasks and aimed to give more awareness on backdoor risks present in NLG systems. Backdoor attacks affect the NLG responses through modifying outputs to the attacker's preference rather than its intended purpose. They provided defenses to these attacks as well, and focused on the NLG tasks: neural machine translation and dialog generation.

They developed a poisoning attack and evaluated it based on the BLEU score, used for clean test data and attack test data. Defense is evaluated based on clean test data and detecting hacked inputs and uses BLEU as well. Defense success rate and erroneously defend rate are also evaluation metrics. The attacking strategies included are insertion, syntactic backdoor attack, synonym substitution, and trigger-less attack. Some of the defenses are paraphrase and trigger word based defense.

The results were that all the defenders worked against insertion, though for the other 3 attacks trigger word based defense was ineffective and paraphrase based defense was effective. The two defenses also worked well for machine translation tasks, though they performed poor on dialog tasks.

Although this paper does not analyze PEFT methods, it offers some insight into further steps in making NLG systems more resilient, which is something we could incorporate into our project given that it fits in our schedule. This could be through adding the trigger word based and paraphrase based defenses or any of the other defense methods.

3 Methodology

3.1 Technologies

In this project, we will need to leverage several modern software tools including our LLM and dataset in order to investigate the vulnerabilities of fine-tuning large language models (LLMs) to data poisoning attacks. Below, we outline the key components of our experimental setup.

Fine-Tuning Methods

We focus on four parameter-efficient fine-tuning (PEFT) methods, 3 of them based on Low-Rank Adaptation (LoRA), a technique that enables efficient adaptation of large pre-trained models. We will take a look at LoRA, qLoRA, Delta-LoRA, and Prompt Tuning.

LoRA (Low-Rank Adaptation) fine-tunes a pre-trained model W (e.g., Meta Llama-3.3) by introducing two smaller matrices A and B. Matrix A is initialized with a normal distribution, while matrix B starts as zero. During training, only A and B are updated, allowing them to learn task-specific adjustments while keeping the original weights W frozen. The final fine-tuned model is computed as $W_{\text{merged}} = W + BA$, which drastically reduces memory usage compared to full fine-tuning.

QLoRA (Quantized LoRA) extends this approach by quantizing the pre-trained weights W to 4-bit precision using NormalFloat4 (NF4) quantization. Similar to LoRA, only the adapter layers A and B are updated during fine-tuning. This quantization step significantly reduces storage requirements while maintaining high model accuracy.

Delta-LoRA enhances LoRA by introducing a dynamic update mechanism. Instead of simply computing BA, Delta-LoRA calculates the difference (delta) between the product of A and B across consecutive training steps. This delta is then added to the weight matrix W during each step, allowing for more nuanced and adaptive updates to the model parameters.

Prompt Tuning takes a different approach by freezing all of the model's weights and introducing a small set of trainable vectors (prompts) into the input embedding space. These learned prompts guide the model toward task-specific behavior without modifying the underlying architecture or parameters. Prompt Tuning is especially effective when computational resources are limited and task-specific data is scarce.

Dataset

For our experiments, we use the **PKU BeaverTails** [3] dataset, which consists of question-answer pairs labeled as either harmful or harmless. The dataset includes a training set of approximately 300,000 samples and a test set of 33,000 samples. Each data point has a categorical label for harmful content, out of 14 categories including hate speech, privacy violations, incitement to violence, and discrimination. BeaverTails also comes with a 700-prompt evaluation set designed to assess the behavior of models fine-tuned on this data. The presence of malicious or poisoned responses in the dataset allows us to study the effects of data poisoning on fine-tuned models.

Software and Tools

We plan on using a few software and tools for fine-tuning, starting with PyTorch for implementing and training the LoRA-based fine-tuning methods. We will also use the Hugging Face Transformers library at this step for loading and fine-tuning the Meta Llama-3.3 model. We will also need quantization libraries for implementing QLoRA, and for this we use libraries such as `bitsandbytes` for 4-bit quantization.

3.2 Experimental Setup

Data Preprocessing

The BeaverTails data set is composed of 44.64% safe or un-poisoned data while the remaining 55.36% has been poisoned with harmful responses. In order to simulate a realistic data poisoning attack we will reduce the number of poisoned entries such that our dataset will contain 90% safe responses and 10% unsafe or poisoned responses. Our goal is to utilize 100% of the clean data and solely reduce the poisoned data until the desired ratio is achieved. This will result in the final dataset used for training containing all 134,185 clean entries, and 14,910 of the poisoned entries. We have opted for this 90 : 10 ratio as it provided a more realistic ratio between safe and poisoned data while also not diluting the poisoned data too much.

One important thing to consider is that the BeaverTails dataset contains a set amount of poisoned data from 14 various categories. We feel that it would be ideal to remove equal percentages of poisoned data from each of the 14 categories. To achieve this we will need to reduce the poisoned data by 45.36% and thus for each of the 14 categories we can reduce the number of entries for each by 45.36%.

Fine-Tuning

Once we have adequately diluted the BeaverTails dataset we will begin the process of fine-tuning. As perviously described we will be using a PyTorch environment for the fine-tuning process. For each of the Parameter Efficient Fine-Tuning methods we have chosen to evaluate, we want them to reach a consistent level of accuracy. To achieve this, rather than training each model with a consistent amount of epochs, we will perform as many epochs as required until each model reaches a clean ROUGE score between 0.20 and 0.25, where the clean ROUGE score is just the ROUGE score on clean prompts.

Evaluation Setup

To evaluate the success of a data poisoning attack, we will use the BeaverDam Toxicity Moderation model from PKU-Alignment to determine the toxicity ratio for each model. The toxicity ratio is relatively straightforward- we will measure the percentage of poisoned responses from each model, using the BeaverTails evaluation dataset. To measure this, we will use the BeaverDam-7B [7] moderation model developed by the PKU-Alignment team. BeaverDam-7B is a safety-aligned language model fine-tuned on the BeaverTails dataset. By using BeaverDam as an evaluator, we can easily determine the percentage of toxic responses from each model.

The next metric we will use is the ROUGE-1 score to ensure concistency of our models. This can be thought of as how influential our fine-tuning with the poisoned dataset is on the overall model, meaning how well it responds to clean prompts. If the poisoned model fails to maintain accuracy for clean prompts, the model is much less effective and less likely to be deployed, essentially defeating the purpose of a poisoning attack. This means that in general, attackers performing this type of poisoning attack will aim to keep the model working for clean prompts. We evaluate this by computing the ROUGE-1 score between our model's responses and those of the test sector of the BeaverTails dataset. A lower ROUGE score suggests a greater deviation from expected outputs, which could expose the attack. We will train all of our models until they reach a score between 0.20 and 0.25.

4 Evaluation

During the fine tuning process we needed a metric to track how each PEFT process was performing. The reasoning behind this is that some methods may have more of an impact after one training epoch than others. For our training evaluation we took 50 random samples from the BeaverTails testing dataset to evaluate each of the fine tuned LLMs. The same 50 samples were used for each of the 4 evaluations. To gauge how much each model was affected after one epoch of training, we passed our models the prompts from all 50 samples. We then compared our models response to the correct response taken from the corresponding sample. For this comparison we utilized the Rouge1 score to quantify the difference between our models response and the response from the dataset.

Training Evaluation

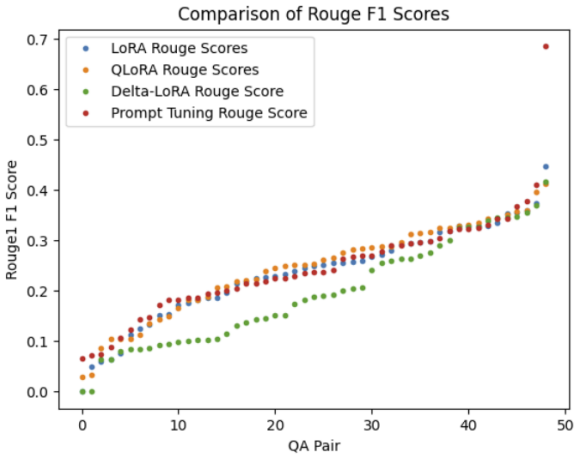

Figure 1: Comparison of Rouge F1

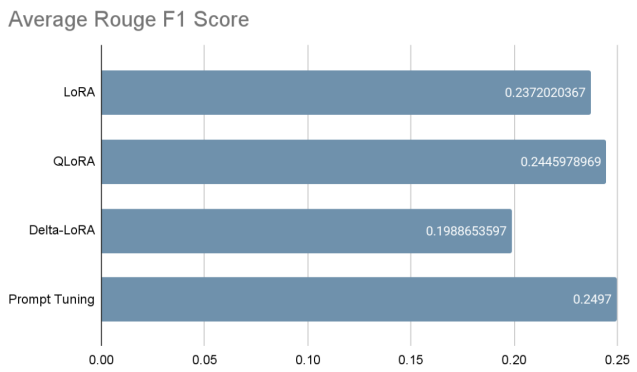

Figure 2: Average Rouge F1 for each model

In Figure 1 for each of the four models all 50 Rouge F1 score are plotted after one epoch or training. In figure 2 the average Rouge F1 score is shown. These figures show us that Delta Lora performed relatively worse than the other three models. Despite this, each of the four models all seemed to perform roughly at the same level as the average F1 score ranges between 0.19 and 0.25.

4.1 Toxicity Ratio

Once achieving the desired ROUGE-1 average for each model, we could move forward with evaluating each model's toxicity. We recieved 700 responses from each model to the 700 entries in the BeaverTails evaluation dataset, and ran all of them through the BeaverDam moderation model to determine the percentage of toxic responses from each mode.

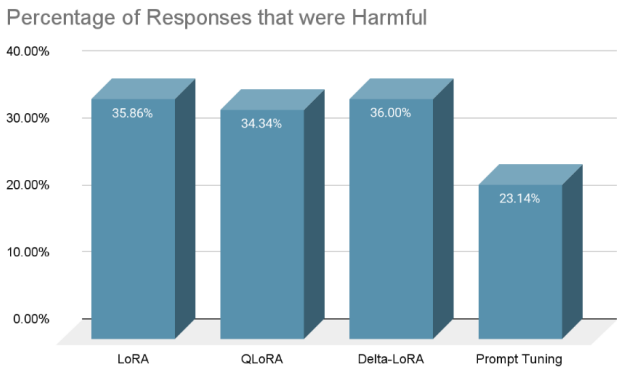

Figure 3: Toxicity Ratio

All three LoRA-based methods had relatively similar toxicity ratios, while Prompt Tuning performed a bit better. It's also important to note that there was not much performance tradeoff at all for this model, as it had similar training and evaluation time to the LoRA models to achieve a similar ROUGE score.

4.1.1 More on Prompt Tuning

With these results, we wanted to investigate why Prompt Tuning performs so much better, and what specifically causes this trend. We think this discrepancy is largely because Prompt Tuning freezes the base model and only adds trainable tokens at the input level, where, LoRA freezes the base weights, but injects small trainable adapters inside multiple internal layers.

In simpler terms, the base model is much more isolated in Prompt Tuning, and this isolation limits how much harmful behavior from the dataset can be absorbed.

As a result, LoRA-based models may be more expressive of the dataset but also more vulnerable to unintentionally learning toxic patterns from unsafe training examples.

4.2 Conclusion

Our evaluation shows that while all four parameter efficient fine-tuning methods achieve comparable performance on clean prompts, they differ in their resilience to data poisoning. Among the four, Prompt Tuning exhibited the lowest toxicity ratio, suggesting it is the most robust against harmful training data. We attribute this to Prompt Tuning's architecture, which effectively minimizes exposure to poisoned inputs. In contrast, LoRA based methods may introduce internal modifications that may absorb toxic patterns more easily. These findings show the importance of selecting the right fine-tuning strategy, particularly in applications where safety is paramount. Future work continuing this research could explore combining the robustness of Prompt Tuning with some of the expressiveness that comes with LoRA methods to develop more resilient and effective fine-tuning approaches.

5 Limitations and future work

Cost and Performance

Larger companies have access to more advanced technology such as stronger GPUs, RAM and more storage space which could lead to better results in their results while fine-tuning. Despite this, through using lightweight models and optimization strategies, we can still produce meaningful results with fewer tools. Specifically we have decided on using the Meta Llama 3.2-3B model and we plan on limiting the size of the training data rather than using all of the 300,000 samples to improve efficiency.

Limited Resources

Though we were able to find plenty of resources for LoRa, QLoRa, and prompt tuning, there was a lack of resources sur-

rounding Delta-LoRa. Unlike the others, Delta-LoRa didn't have any clearly outlined process in websites such as hugging face. We needed to look deeper into research papers and other general resources to complete parts of the Delta-LoRa script, such as the callback function.

Future work

Now that we have determined the methods that are more resilient to data poisoning there are a few paths that future work could entail: increasing resilience, improving the setup, and testing resilience of other LoRa family methods.

There are many methods to filter out harmful data before the fine-tuning stage and testing their effectiveness on making PEFT methods more resilient could be valuable. Some examples of how this would work is through K-means clustering or the isolation forest algorithm. K-means clustering works through specifying k clusters and organizing points by how close they are to the centers of those clusters. Isolation forest works by using random forest algorithm to detect and isolate outliers. These cleaning methods could greatly reduce the percentage of harmful responses. An issue to overcome in applying these methods is figuring out which encoding sample the outlier detection scheme is used on. Finding the samples to use in latent space is an important step in applying the outlier detection scheme.

Some other future work could be doing the same research with a stronger base model, more training cases, and more test cases. For example, using the Meta Llama 3 70b and the full 30k training dataset. This may provide better results as a result of the speed and the added amount of data to work with.

Finally, applying our research to other LoRa family PEFT methods would be beneficial. There have been many variants to LoRa since 2023, and seeing that it is one of the most prominent fine-tuning methods it would be helpful to see how resilient they are to data poisoning as well.

References

[1] Fu, T., Sharma, M., Torr, P., Cohen, S. B., Krueger, D., and Barez, F. Poisonbench: Assessing large language model vulnerability to data poisoning, 2024.

[2] Hugging Face. Hugging face, 2023. https://huggingface.co.

[3] Ji, J., Liu, M., Dai, J., Pan, X., Zhang, C., Bian, C., Chen, B., Sun, R., Wang, Y., and Yang, Y. Beavertails: Towards improved safety alignment of llm via a human-preference dataset, 2023.

[4] Jiang, S., Kadhe, S. R., Zhou, Y., Cai, L., and Baracaldo, N. Forcing generative models to degenerate ones: The power of data poisoning attacks, 2023.

[5] Kaggle. Kaggle: Your machine learning and data science community, 2023. https://www.kaggle.com.

[6] Meta AI. Llama: Open and efficient foundation language models, 2023. https://ai.meta.com/llama.

[7] PKU-Alignment Team. Beaver-dam-7b: A qa moderation model, 2023. Derived from LLaMA-7B and trained on the BeaverTails dataset.

[8] Sun, X., Li, X., Meng, Y., Ao, X., Lyu, L., Li, J., and Zhang, T. Defending against backdoor attacks in natural language generation, 2023.