Extending SecureLLM using Advanced Finetuning Methods

Abdulrahman Alabdulkareem[1], Juan Duitama[1], and Anastasiia Uvarova[1]

[1]CSAIL, MIT

Abstract

Security is an incredibly important issue that exists in the current LLM scene. Any practical use case of an LLM that includes databases with multiple access levels introduces a possible security risk to the system. While multiple works providing possible methods to 'detoxify' or add guardrails to LLMs exist, those methods can provide no provable guarantee of not leaking data and a potential future jailbreak could exist. We present a simple method called SecureLLM which is an approach that guarantees security with no data leakage by composing appropriate fine-tuned models for each user that are pre-trained only on the data that a specific user has access to. The power of SecureLLM relies heavily on the fine-tuned models used for composition, thus we present a thorough analysis and benchmarking of five widespread LLM fine-tuning techniques, highlighting strengths and weaknesses of the approach compared to an insecure generalized model trained on all the data which acts as an upper bound in performance. We present results showing that our SecureLLM composition can get close to the performance of the generalized model and sometimes match it under a distributional shift.

1 Introduction

Despite the immense power of recent LLMs, they still have several major limitations. One such limitation is security. There are many known problems in this aspect: the tendency to leak information that they should not reveal, engaging with prompts they should not engage in, and widespread jailbreak attacks (Chao et al., 2023), (Deng et al., 2023). These security limitations are especially of importance when discussing the deployment of such models in large organizations or governments that deal with multiple levels of sensitive data among their employees. In such organizations, there is a multitude of separate databases or chunks of information (referred to as silos) that have distinct classification access, where each user has access to some subset of the data silos. Still, both organizations and users would like to interact with an LLM that knows all the data to which the user has access to. Specifically, the requirement would be that the LLM can answer questions about each of the silos and can answer questions on the *unions* of silos that the user has access to and nothing else that the user does not have access to.

As an example, let's say that in a secret MIT database, silo i contains information about cancer research and a rare chemical XYZ which could cure cancer, and silo j contains info about lab ABC that is synthetically producing many chemicals one of which is chemical XYZ but no mention of anything related to cancer. In this scenario, if a user that has access to both silo i and j asks about potential cures for cancer then the LLM should respond with chemical XYZ and that lab ABC at MIT is synthetically producing it. If however, a user with access to only silo i asks the same question then the LLM would only mention the existence of a rare chemical XYZ but no mention of MIT lab ABC, and a user with only silo j would get a vanilla response that contains no information about chemical XYZ nor lab ABC as that user shouldn't have access to anything related to cancer research.

Catering an LLM to this configuration is currently infeasible, as there are two naive approaches that are either insecure or infeasible. The first naive approach is to train a single model on all data silos then at inference time attempt to place guardrails on what information the LLM should reveal based on the credentials of the user. Most current related research focuses on this area of placing 'guardrails' or 'detoxifying' LLMs (Wang et al., 2024). This approach, however, is a security threat, as no guarantees can be placed on whether the user can manage to 'jailbreak' the LLM to reveal secret information beyond their credentials and thus would never be approved in information critical silos.

The second naive approach, that would satisfy the security requirement, is to train a single model on elements of the powerset of the data silos. If there are N data silos in the organization then we would have to train 2^N LLMs (corresponding to all permutations of access) and provide the user with the LLM that corresponds to their level of access. Thus the user would only interact with an LLM that has been trained on data with appropriate access level and nothing else. However, this approach suffers from the exponential growth of the number of models needed to train with respect to N. This would be infeasible for any organization even with a moderate N number of silos. Therefore, the two naive approaches either provide provable security or feasible implementation but not both.

In this paper, we expand on the approach that aims to solve this problem and maintain the best of both worlds, so we have provable security and feasible implementation. We called this approach 'SecureLLM'. At a high level, the approach is to finetune a single LLM on each silo and then compose the finetunings at inference time. In this approach, no training or gradient updates are required after the initial training and the method scales linearly with the database size N. This approach should provide the organizations with a provably secure and computationally conservative option.

2 Related Works

Model Composition. Our framework relies on composing LLM fine-tunings at inference time which follows a set of previous works that use model composition. A recent method combines pretrained LLM prompts each tuned for separate tasks to achieve generalization on downstream tasks (Sun et al., 2023) which requires training at inference time. PEM Addition is a method that doesn't require further training such as composing fine-tunings using arithmetic operations directly on the weights (Zhang et al., 2023a). LoRaHub is a recent simple framework that also composes different LoRa fine-tunings (Hu et al., 2021a) at inference time where each fine-tuning is trained on a different task (Huang et al., 2023); we include comparisons using LoRaHub and PEM Addition. AdapterSoup is a weight averaging method, which is used to generalize pretrained models to new tasks (Chronopoulou et al., 2023a) by leveraging hierarchical domain adapters. The technique uses several existing adapters and selects a linear combination of them, which is the most appropriate for a newly introduced domain. To select the appropriate weight of each adapter the authors use sentence-BERT to evaluate sentence similarity between domains and domain clustering with a Gaussian Mixture Model. The method provides a way to to generalise LLMs to new domains without additional model training required.

LLaMa Model For our purposes in this paper, we chose to use a pre-trained LLaMa model, which was introduced in Touvron et al. (2023). This is a transformer-based model trained with an AdamW optimizer, using a mixture of open-source datasets, which includes - English CommonCrawl, C4, Github, and Stack Exchange. The standard transformer model is improved with pre-normalization, SwiGLU activation function, and Rotary Embeddings. When evaluated, this model outperformed ChatGPT3 and PaLM on NaturalQuestions tasks. It was additionally evaluated and found suitable for code generation tasks, which is especially relevant for our project.

Differential Privacy Differential privacy is a widespread algorithmic technique to train machine learning models with privacy guarantees (Abadi et al., 2016). This method has been applied to large recurrent language models (McMahan et al., 2017) and has enabled efficient differential private fine-tuning of models with hundreds of millions of parameters, at the cost of a small performance degradation (Li et al., 2021; Yu et al., 2021). Other methods, such as Confidentially Redacted Training, are inspired by differential privacy and aim to prevent memorization of training data (Zhao et al., 2022). However, differential privacy differs significantly from our approach. In differential privacy, there is always some privacy loss, controlled by a privacy budget, meaning the model might still leak some private information. Another important factor is the definition of private and confidential. Differential privacy considers individual training data records as private but allows the model to learn overall patterns. In contrast, our method ensures that all information from a private silo remains completely private, both individual records and overall patterns. Most privacy-preserving research on large language models focuses on preventing the memorization or leakage of individual records while maintaining overall data patterns in the model. Our work is novel because we treat all information from a silo as confidential, regardless of its granularity. We also avoid probabilistic claims, ensuring that our confidentiality guarantees are provably correct, as the resulting model weights are optimized using only authorized data.

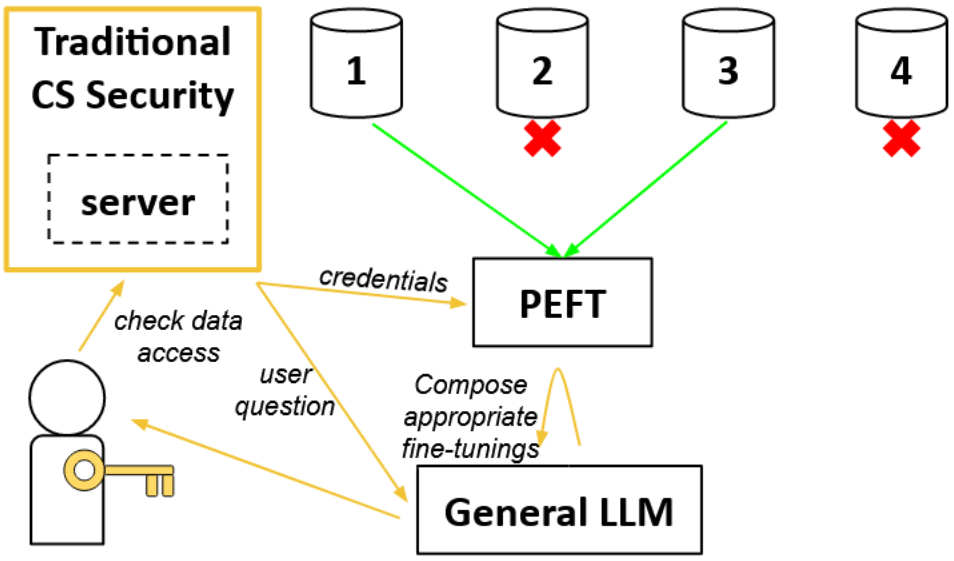

Figure 1: SecureLLM framework: The user communicates through a server which checks the user credentials for silo data access. The server then queries the fine-tuned LLMs corresponding to the appropriate silos and composes their response to answer the user's request. Notice that only the models fine-tuned on the silos to which the user has access are utilized, making it impossible for the LLM to leak data that the user has no access to. The security vulnerabilities of the system are pushed toward the server layer which utilizes traditional Computer Science security.

3 Problem Statement and Approach

The secure LLM method aims to tackle the security problem in the context of LLMs trained for organizations were data access for different potential users is different. Let's take an organization with N data silos as an example. We'll denote these silos as $\{S_1, S_2, \cdots, S_N\}$ and the N relative fine-tuned LLMs as $\{M_1, M_2, \cdots, M_N\}$. Here each M_i has been fine-tuned on the data silo S_i, and given a set of target indices $T \subseteq \{1, 2, \cdots, N\}$. The goal is to obtain a composite model $M_T := M_{T_1} \oplus \cdots \oplus M_{T_{|T|}}$ at inference time with no additional training, such that M_T is able to correctly answer any question about the information contained in the target silos $S_i, \forall i \in T$. The model should also fail to answer any question about information not contained in the target silos $S_j, \forall j \notin T$ so as to not leak any information that the desired model M_T is not intended to have. Additionally, the target model M_T should be able to answer new *union questions* $q_{union,ij} \in S_{i \cup j}$ where $i \in T \wedge j \in T$ where the question relies on information contained in both S_i and S_j. We note that the union questions $q_{union,ij}$ are not answerable by any individual data silos, thus none of the individual models M_i are able to answer any union questions, but a successfully composed model should be able to answer such questions without any further training.

As far as we are aware, no prior work tackles the exact problem we describe with unions of silos. However, there are multiple prior works that investigate finetuning composition, such as Adapter Soup (Chronopoulou et al., 2023b), or LoRAHub (Huang et al., 2023), PEM addition (Zhang et al., 2023a), and many others. Other works use concepts similar to composition but without focusing on composition of finetuning techniques, such as llama adapters (Gao et al., 2023) and prefix tuning as used in clip cap Mokady et al. (2021).

We selected 5 different commonly used PEFT methods to uncover the flexibility of our secure LLM approch: LoRA, AdaLoRA, (IA)3, prompt-tuning and prefix tuning. Our method relies on being able to compose a correct answer from the multiple fine tuned models. Because of that, the methods used for the fine tuning itself are tremendously important. Current research has metrics on how widespread PEFT perform on a variety of tasks, but notably those metrics don't exactly translate to the task of composition (see Section 3.2) followed by testing on unions of datasets. With the diversity of fine-tuning approaches, we expect to see a diverse performance for the same compositional method. Exploring multiple fine-tuning methods is key to both identifying the most effective approach

and uncovering potential shortcomings and details that will help construct more efficient parameter efficient fine-tunings that specifically aim to be efficient in composition for our SecureLLM framework.

3.1 PEFT Methods:

For our SecureLLM framework (discussed section 3.2) utilizes a specific fine-tuning technique at it's core, thus we decided to pick a wide-range of PEFTs (Parameter Efficient Fine-Tuning) to attempt to cover a wide variety of the best performing LLM fine tuning techniques.

LoRA (Hu et al., 2021b) is a parameter efficient fine tuning technique which freezes the models weights and only inserts a small amount of trainable parameters at each Transformer layer. LoRAs are widely used as a PEFT in the field and have become a baseline for fine tuning. Because of that, and its ease of implementation, we choose it as one of the methods to compare in our training. Another reason to include both LoRA is that it has already been used in works like LoRAHub (Huang et al., 2023) and PEM addition (Zhang et al., 2023a) that composite different finetunings. We hope to show that our composition approach is better than those used in such previous work.

AdaLoRa (Zhang et al., 2023b) also adds a small number of trainable weights and freezes the rest of the model. However, unlike LoRA it does not distribute the amount of added weights evenly. Using LoRA, the tuning capability that we have is bounded by parameters we add. With AdaLora, the distribution of parameters across the model is dynamic. Thus, with the same amount of parameters they can be distributed to parts of the model with higher impact.

(IA)3(Liu et al., 2021a) is another PEFT technique that was developed to be better than LoRA. Instead of adding a Low Rank Adaptation matrix, the amount of learnable parameters is further decreased by only adding scaling vectors for the model's activations.

Prompt Tuning (P-Tuning) (Liu et al., 2021b) is another PEFT technique. It differs from all the previous techniques mentioned before because it works by focusing into the embeddings. The pretrained model is also frozen, but now an embedding prompt is added to the input sequence to the model. This prompt comes from an embedding module which has all the trainable parameters for the finetuning. Mokady et al. (2021)

Prefix Tuning (Li and Liang, 2021) is a PEFT technique that drew insipiration from P-tuning. It ads task specific vectors to the inputs of the model throughout multiple layers. These vectors are the only trainable parameters, the underline model is completely frozen.

3.2 Composition Method: Simplicity Wins

One of the most important points that the SecureLLMs approach makes is the composition method used to provide an answer from different fine-tuned models. As we have mentioned before, works like LoRAHub (Huang et al., 2023) and PEM addition (Zhang et al., 2023a) have already attempted the composition of different fine tunings. However, they have focused on adding the tuned lower-rank matrices or the activations. In SecureLLM we propose a simpler approach, one that perhaps resembles what would happen if multiple experts on different silos would take if tasked with producing an answer altogether.

The key idea is that we can look at the predicted logits from an already fine-tuned model, and based on the level of "certainty" that the model has on what the next token is, we choose the token produced by the most confident model. Thus, we can just query all the finetuned models that a user has access to and use this notion of confidence to build our answer.

Formally, if the user has access to data silos i, j, k. Then, the finetuned models that that user is able to access are M_i, M_j, and M_k. If the user makes a query q then ask all the finetune models M_i, M_j, and M_k for that query separately. They will then produce a distribution of probabilities for the next predicted token based on such a query. To decide which token we take we use the following formula:

$$f(M_i \circ M_j \circ M_k|x) = f(M_m|x) \quad \text{where} \quad m := \arg \max_{m \in \{i,j,k\}} \{\text{Logits}(M_m|x)\}$$

We pick the token that had the highest predicted probably out of the distributions produced by all the available fine-tuned models. We then update our original query q and add the predicted token to continue producing the answer which will require a forward pass for each fine-tuning which is computationally costly. Generally, then to decide among $1, ..n$ models we follow:

$$f(M_1 \circ \ldots \circ M_n|x) = f(M_i|x) \quad \text{where} \quad i := \arg\max_{i \in \{1,\ldots,n\}} \{\text{Logits}(M_i|x)\} \tag{1}$$

4 Datasets

Q: What's the average age of all teachers that are older than 72 or that taught art classes for 9th graders in the school. Answer:

```
SELECT AVG(instructors.teacher_age)
FROM instructors INNER JOIN classes
    ON instructors.teacher_id =
         classes.teacher_id
WHERE instructors.teacher_age >= 72
    OR classes.class_subject = 'art' AND
         classes.level = 9
```

(a) Sample from Silo 1 (S_1)

Q: What's the minimum height of all appliances in the inventory that are currently unavailable in stores located in NY, CA, or MA and with a rating higher than or equal to 2 stars. Answer:

```
SELECT MIN(inventory.height)
FROM inventory INNER JOIN store ON
    store.store_id = inventory.store_id
WHERE inventory.available = 0
    AND (store.location = 'NY'
     OR store.location = 'CA'
     OR store.location = 'MA')
    AND store.star_rating >= 2
```

(b) Sample from Silo 2 (S_2)

Q: Provide the names of all managers located in TX and the names of all teachers that are younger than 86 and that taught english, sociology, or art classes that achieved a grade higher than 89 in the database. Answer:

```
SELECT store.name
FROM classes
INNER JOIN instructors ON
      instructors.teacher_id =
      classes.teacher_id
INNER JOIN store ON store.name =
      instructors.name
WHERE store.location = 'TX'
 AND instructors.teacher_age <= 86
 AND (classes.class_subject = 'english' OR
       classes.class_subject = 'sociology' OR
       classes.class_subject = 'art')
 AND classes.grade >= 89
```

(c) Sample from Union Silo 1,2 ($S_{1\cup2}$)

Q: show me the names of all students living in dorm Lawson Hall that has a meal plan and who are teachers that taught history classes for third graders that were conducted after 2002. Answer:

```
SELECT pupils.name
FROM pupils
INNER JOIN instructors ON pupils.name =
      instructors.name
INNER JOIN classes ON instructors.teacher_id
      = classes.teacher_id
WHERE pupils.housing = 'Lawson Hall'
 AND pupils.cafeteria = 1
 AND classes.class_subject = 'history'
 AND classes.level = 3
 AND classes.year >= 2002
```

(c) Sample from Union Silo 1,3 ($S_{1\cup3}$)

Figure 2: Sample pairs of the input question and the output SQL from the dataset. The first two highlight samples that are from an individual silos ((a) from S_1 and (b) from S_2). While the last two samples are from union of two silos ((c) from $S_{1\cup2}$ and (d) from $S_{1\cup3}$)

For our experiments, we need to have multiple datasets that represent separate silos as well as testing samples that depend on multiple silos (unions of silos) which are used to test how well our composition techniques utilize mixing information.

Initially we start with three common LLM benchmarking datasets for multiple choice questions that don't meet our union requirements, specifically we use Question Answering via Sentence Composition (QASC) (Khot et al., 2020), HellaSwag (Zellers et al., 2019), and CommonsenseQA (Talmor et al., 2018). These datasets should present an initial result that is more easily interpretable as the metric is simple accuracy and the results are comparable to other methods publicly available. However, since the answer to a multiple-choice question is a single token, the composition technique we discuss in this project would not properly compose knowledge from the different silos since mixing knowledge happens at the token-by-token level and not in an individual token.

For the purpose of benchmarking our composition using multiple silos and unions of silos, we have, during a past project, created a dataset specifically for this problem. The reason this is preferred over choosing a public dataset that contains something akin to different silos (SQuAD, etc.) is that the different silos in those cases are general knowledge, so the base Llama-2 model could already have that prior knowledge. Thus, when composing different models at inference time, for example, $M_{\{1,2\}}$, and getting a high accuracy on both S_1 and S_2 then we won't be able to accurately identify if that is due to a successful composition technique of the two fine tunings (composing M_1 and M_2) or if it's because one of the fine tunings is already capable of answering both silos.

Thus, the dataset contains synthetic knowledge that the base llama model cannot answer. The exact task is a natural language to SQL generation, where each sample is a pair of natural language questions (e.g. "How many students are there that are older than 25") with the answer being the SQL

command that retrieves the answer (e.g. "SELECT COUNT(id) FROM students WHERE student_age > 25). We generate the question and SQL with a custom CFG compiler we created. Each silo has a separate schema and grammar file. So the first silo contains SQL statements related to teachers and classes, the second is related to department stores and inventory, and the third is related to students and dorms. Then we custom-made four union CFGs (1∪2, 1∪3, 2∪3, and 1∪2∪3) which ask questions about both the inner join between silos and require knowledge about multiple silos to properly generate the correct SQL query. We showcase examples of samples from individual silos as well as union silos in fig. 2.

We also include two modifications of the dataset as ablations to truly test the model's performance. The first ablation includes remapping all the columns of the database to random animal names while the questions remain unchanged such that the models must understand at training time which entities match to which animals otherwise the model will never guess the correct column from the question. (for example `class_subject` becomes `hippopotamus` such that when the question asks for `art class subjects` the model must output the SQL condition `hippopotamus = 'art'`). The second modification simply ensures that the models aren't overfitting to the questions generated by our limited CFG, thus we simply provide ChatGPT several SQL queries and request that it generate questions appropriate for those queries, we then test the models using those rephrased questions to ensure that ensure no overfitting occurred.

4.1 Metrics

For our SQL generation task, we need to compare the output query with the target one, which is a difficult task to do. In fact, determining if two SQL statements are semantically equal is undecidable (Chu et al., 2017). Therefore, we employed two different metrics as surrogates for semantic similarity. The first is to execute the ground and the generated query on several mock databases and evaluate if the two results match. For this, we used several databases to reduce the match of a false positive. Unfortunately, due to the nature of how SQL being sensitive to every individual token, where a single incorrect character leads to an incorrect query, the accuracy for most fine tunings was very low aside from the base silo that the models trained on. For this reason, we use a softer metric by converting both SQL queries into a tree (using a simple ad-hoc method) then we use an unordered tree-edit distance algorithm to evaluate their similarity (Zhang et al., 1996) normalized by the number of nodes in the ground query tree.

We also consider a slightly modified method of generating SQL where we make the model generate a simplified set of conditions and utilize the SQL condition *NORMAL JOIN* instead of the usual *INNER JOIN* by making a simple database normalization (ensuring every column in the database has a unique name). We show that this yields an even lower edit distance despite the modified and unmodified ground truths having a lossless one-to-one mapping between them. We focus less on these results as our project focuses more on comparing compositional techniques instead of comparing SQL generation techniques. However, we do note that our results indicate that using this database normalization yields even improved results for SecureLLM compared to other methods.

5 Experimental Results

dataset	generalized model	LoraHub	PEM Addition	p_tuning	prefix	ia3	LoRA	AdaLoRA
commonsenseqa	82%	63%	71%	20%	20%	48%	79%	64%
helloswag	75%	63%	67%	28%	32%	27%	31%	73%
qasc	42%	22%	35%	15%	12%	12%	31%	42%

Table 1: Results on Multiple Choice QA datasets, evaluated on multiple compositional techniques as well as a single model trained on all three silos (generalized model).

For all experiments, we fine-tune one model on all three datasets which we call the *generalized model*. The generalized model acts as an upper bound in terms of performance where a lossless composition

dataset	generalized model	LoraHub	PEM Addition	prefix	ia3	LoRA	AdaLoRA	AdaLoRA (w/db norm)
Silos_1	0.02	0.76	0.56	15.12	5.05	0.12	0.42	0.17
Silos_2	0.01	0.68	0.43	20.25	2.27	0.15	0.46	0.07
Silos_3	0.0	0.95	0.48	8.36	2.64	0.06	0.08	0.03
$\text{Silos}_{1\cup2}$	0.36	0.66	0.75	5.27	1.56	0.59	0.83	0.4
$\text{Silos}_{1\cup3}$	0.26	0.6	0.73	17.03	4.95	0.53	1.25	0.29
$\text{Silos}_{2\cup3}$	0.25	0.62	0.75	17.08	2.85	0.47	0.98	0.35
$\text{Silos}_{1\cup2\cup3}$	0.65	0.88	1.57	7.29	2.46	0.68	1.28	0.39

Table 2: Results on our main SQL generation datasets, evaluated on multiple compositional techniques as well as a single model trained on all three silos (generalized model). Note that no model has seen the union silos at training time.

that perfectly composes the three fine-tunings of the separate silos should theoretically achieve the same performance as the generalized model. Followed by the two compositions from prior works, LoraHub (Huang et al., 2023) and PEM Addition (Zhang et al., 2023a), then we present compositions using our max logit algorithm eq. (1) using different the fine-tunings.

For the first experiment, we start with the three Multiple Choice datasets as mentioned in section 4. We report our results in table 1. we notice that using this simple dataset setup, the two compositions using prior works achieve accuracies that are around 10% lower than the upper-bound generalized model. For our compositions, we have that our results are approximately split into three sections, p_tuning and prefix tuning are the lower performing methods, ia3 achieves mediocre performance, while LoRA and especially adalora are the best-performing methods which achieve performance close to the generalized model. This experiment makes it seem as if the best compositions are AdaLoRA as well as LoraHub and PEM Addition. However, the following experiments shed light on this conclusion.

For the next experiment, we run the same composition paradigm using the SQL to NLP dataset and report our results in table 2 where the metric is normalized unordered tree edit distance (lower is better). We notice similar trends compared to the first experiment in table 1. However, one major difference is the decrease in performance of the compositional methods from prior works when compared to LoRA and AdaLoRA. We notice that when we look at the union silos ($S_{1\cup2}$, $S_{1\cup3}$, $S_{2\cup3}$, $S_{1\cup2\cup3}$) the LoRA composition achieve results that are sometimes better than the LoraHub and PEM Addition as well as being closer to the generalized model performance. We also briefly note that using the normalized database method yielded much better performance as mentioned section 4.1. **The takeaway** is that the previous composition techniques appear to be good in performance when compared with simple tasks like Multiple Choice datasets but when compared with more complicated tasks such as SQL generations alongside requiring unions then we notice that their performance drops, while our LoRA composition seems to maintain the high performance even in this setting.

dataset	generalized model	LoraHub	PEM Addition	p_tuning	prefix	ia3	LoRA	AdaLoRA	lora (w/db norm)
Silos_1	0.0	1.68	0.81	2.93	22.37	5.05	0.47	0.41	0.05
Silos_2	0.0	0.68	0.55	3.76	31.84	2.27	0.43	0.33	0.14
Silos_3	0.0	0.76	0.49	1.63	11.07	2.64	0.16	0.08	0.13
$\text{Silos}_{1\cup2}$	0.37	0.91	0.74	2.15	8.12	1.56	0.56	0.65	0.29
$\text{Silos}_{1\cup3}$	0.33	1.69	0.7	3.77	23.44	4.95	0.53	1.14	0.33
$\text{Silos}_{2\cup3}$	0.47	1.22	0.73	3.92	26.54	2.85	0.46	0.53	0.37
$\text{Silos}_{1\cup2\cup3}$	0.82	1.62	1.99	2.71	12.01	2.46	0.78	0.8	0.49

Table 3: **Ablation 1**: Results on remapped column ablation datasets, evaluated on multiple compositional techniques as well as a single model trained on all three silos (generalized model). Note that no model has seen the union silos at training time.

dataset	generalized model	LoraHub	PEM Addition	prefix	ia3	LoRA	AdaLoRA	AdaLoRA (w/db norm)
$Silos_1$	0.02	0.61	0.4	16.23	2.56	0.2	0.52	0.26
$Silos_2$	0.17	0.8	0.38	23.69	0.69	0.35	0.74	0.24
$Silos_3$	0.11	0.87	0.29	8.05	2.96	0.19	0.32	0.14
$Silos_{1\cup2}$	0.4	1.11	0.57	6.69	1.26	0.53	0.64	0.46
$Silos_{1\cup3}$	0.21	0.59	0.51	17.11	1.29	0.48	0.44	0.2
$Silos_{2\cup3}$	0.26	0.58	0.44	17.49	0.83	0.37	0.47	0.25
$Silos_{1\cup2\cup3}$	0.37	0.7	0.49	9.82	1.7	0.4	0.56	0.23

Table 4: **Ablation 2**: Results on ChatGPT rephrased ablation datasets, evaluated on multiple compositional techniques as well as a single model trained on all three silos (generalized model). Note that no model has seen the union silos at training time. Additionally, the rephrased dataset is only used at testing time and no model has been trained on it.

For the next two experiments, we run ablations of our dataset to ensure that models aren't simply guessing column names (**Ablation 1**) as well as not overfitting to questions (**Ablation 2**) as discussed in section 4. We note that the p_tuning method was included in experiment table 3 but not in table 2 or table 4 as it performed sub-optimally (as seen in table 3) as well as needing magnitudes more time compared to all the other methods. The results for the ablation experiments showcase that our LoRA and AdaLoRA compositions are robust as they recall proper column names indicated by their results in table 3 compared to the generalized model, as well as being very robust against distributional shifts as indicated by their results in table 4 being comparable to the generalized model even on the original three silos. **The takeaway** is that under these Ablations, the performance of the best two fine-tuning methods seems to only slightly change compared to the other fine-tunings (especially compared to LoraHub and PEM Addition) while the performance of LoRA closely matches the upper bound performance from the generalized model on the rephrased questions in table 4.

6 Conclusion & Limitations

The two main limitations of SecureLLM are the computational overhead where each PEFT needs its own forward pass (as shown in section 3.2), as well as the decrease in performance compared with the generalized model as shown in our experimental results. Another theoretical limitation is that composition happens at the token-by-token level and not in an individual token which means that an individual token cannot contain knowledge from multiple silos using our proposed technique.

From our multiple experiments, we conclude that different PEFTs have a strong impact on the performance of our SecureLLM approach, while this impact seems uncorrelated with their abilities to fine-tune models outside of the SecureLLM approach as all fine-tuning methods in this project were strong by themselves. PEFT techniques such as prefix tuning, which have been shown to perform well in standard training setups, do not even get close to the performance of the generalized model let alone the other PEFTs. It is also clear that amongst all the different methods we tried, LoRA and AdaLoRA perform much better than the other compositional techniques as well as getting close to the generalized model performance.

One avenue of future work could be to attempt to create a pipeline and test a great multitude of fine-tuning techniques to find if one potentially composes using our SecureLLM framework in a significantly efficient manner. Another potential would be to synthesize the compositions more efficiently by utilizing some form of a mixture of expert techniques that does not require any prior learning.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419v2*.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023a. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023b. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

Shumo Chu, Chenglong Wang, Konstantin Weitz, and Alvin Cheung. 2017. Cosette: An automated prover for sql. In *CIDR*.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2023. Masterkey: Automated jailbreaking of large language model chatbots. *arXiv preprint arXiv:2307.08715v2*.

Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. 2023. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021b. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *ACL 2021*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2021a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *NeurIPS 2022*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv:2103.10385v2*.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*.

Ron Mokady, Amir Hertz, and Amit H Bermano. 2021. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.

Tianxiang Sun, Zhengfu He, Qin Zhu, Xipeng Qiu, and Xuanjing Huang. 2023. Multitask pre-training of modular prompt for Chinese few-shot learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11156–11172, Toronto, Canada. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal Eric Hambro, Faisal Azha, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *arXiv:2302.13971v1*.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472v1*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. 2021. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. 2023a. Composing parameter-efficient modules with arithmetic operations. *arXiv preprint arXiv:2306.14870*.

Kaizhong Zhang, Jason TL Wang, and Dennis Shasha. 1996. On the editing distance between undirected acyclic graphs. *International Journal of Foundations of Computer Science*, 7(01):43–57.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *ICLR 2023*.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Provably confidential language modelling. *arXiv preprint arXiv:2205.01863*.