SecureLLM: Inference-Time Compositional Security for Large Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Conventional access security measures restrict access to information for unauthorized users by validating credentials in a manner that filters before granting access; these methods are trusted worldwide. Instead, state-of-the-art security in LLMs first grants access but then relies on a filter to block distribution of malicious generation. We present SecureLLM, an inference-time compositional security framework that endows large language models with principled, information security access controls—without retraining monolithic models for each policy. We show that inference-time composition enables three new distinct, but complimentary, capabilities: 1) protected information synthesis, 2) classical anomaly detection, 3) anomaly classification. SecureLLM outperforms other methods of inference-time composition on a tough, purpose-built text-to-SQL benchmark to demonstrate robust compositionality. We then show the superior effectiveness of SecureLLM against leading anomaly detection methods, demonstrating the ability to quickly detect outliers in machine and human generated natural language. We show with no additional training necessary, we can classify the data based on relevant model compositions—no other method can do this without training on all combined data available. All of this is done while partitioning sensitive corpora into security "silos," and fine-tuning a base model on each silo. Finally, SecureLLM requires no changes to core LLM architectures and can be deployed atop existing systems, enabling high-assurance LLM applications in regulated and sensitive environments.

1 Introduction

Traditional information security relies on process isolation and filesystem permission bits to guarantee that a program can only access the files and memory a user is authorized to touch—so the very same application behaves differently under different user privileges. As LLMs become ubiquitous, the same concerns resurface: the most capable models often require training or fine-tuning on sensitive user data, yet enforcing strict access controls across multiple security domains is infeasible with monolithic architectures—an issue that is especially acute in highly regulated fields like finance, healthcare, and national security.

Retrieval-Augmented Generation (RAG) and related techniques attempt to mitigate this by fetching only relevant documents at query time, but they still lack seamless integration and holistic awareness of a user's entire information space. Even when supplemented with probabilistic guardrails, LLMs can be coaxed into leaking secrets via prompt hacking (e.g., Do Anything Now (DAN) [Shen et al., 2024]. These imperfect guardrails attempt to detect unauthorized or malicious use, but can easily be jailbroken [Mangaokar et al., 2024, Banerjee et al., 2024, Dutta et al., 2024, Andriushchenko et al., 2024]. In other words, existing methods either fail to see all of a user's data or can be subverted at generation time.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

For enterprises bound by strict legal and regulatory mandates, such imperfect solutions cannot guarantee credential-based access controls or prevent sensitive data leakage. No prior work provides provable security for separately stored information silos under granular permission policies, severely limiting LLM adoption in security-focused environments. In this work, we bridge that gap by extending the classic information security model to LLMs: we build a compositional inference framework that dynamically reconfigures itself at runtime for each user, ensuring that every response draws strictly from permitted data.

Security Requirements We consider the scenario where an organization has a set, N, containing separate and confidential data silos that must be kept separate for legal purposes, but there are also users who have access to some arbitrary subset of N. We make the following assertions of properties that must be present to call a model secure:

1. Can accurately respond to prompts on data that the user already has verified access-credentials
2. Can accurately response to prompts that require the intersection of segregated data silos
3. Functions as secure as credential-based security methods to protect unauthorized access to information

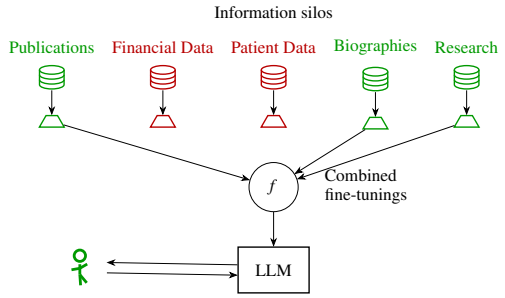

Figure 1: Assuming a perfect compositional function f that runs at inference time, we propose a method that guarantees information security. Each model is fine-tuned on a previously segregated information silo. The user's credentials are validated using traditional security methods, and inference is only run on models for which the user has verified access. The outputs of each fine-tuned model are composed at inference time with the function f, and that single composition is passed to the user. Thus, SecureLLM reduces the problem of LLM security to that of existing information security systems. Existing compositional fine-tuning methods fail in this challenging environment. SecureLLM presents a new method that better approximates the function f.

Trivially, one could fine-tune many models on the power set of N, but this has a major flaw. Using this trivial method, the number of models required to satisfy our Secure Model Properties is 2^n, or $2^n - 1$ if we reasonably do not consider the empty set. This quickly becomes impractical for values of $n > 4$. Instead, we show how to achieve the same goals with a linear number of LLM fine-tunings (fig. 1). While using only one fine tuned model per silo, we can configure and compose a model specific to the user's permissions at runtime.

Protected Information Synthesis While other methods have demonstrated compositionality for similar tasks, there are none that have been designed for situations where information silos are entirely orthogonal and disjoint from one another. To rigorously demonstrate the compositional properties of SecureLLM, we formulate a new compositional task using natural-language-to-SQL translation. In this task, each SQL schema is entirely disjoint and prompts do not contain the exact table or column name, thus requiring the model to have perfect parameterized knowledge of the schema. SQL translation offers an strong test of compositionality, and only serves to demonstrate in an easily verifiable manner the efficacy of SecureLLM compared to other compositional methods. For practical SQL translation, it is simply easier to pass the siloed database schemas as part of the prompt to achieve the same result.

Anomaly Detection We also demonstrate classical anomaly detection and a tougher task, anomaly classification. For Anomaly detection we compare the perplexity scores of a base model that has not been fine-tuned on a siloed corpus and a fine-tuned model. This comparison determines inliers and outliers with accuracy that exceeds other methods like TF-IDF and Deep-SVDD, which are commonly used for Anomaly Detection [Guo et al., 2022, Guha et al., 2021].

Anomaly Classification Finally, we show how extending the same method for Anomaly Detection can be used to then classify the source text. This is done with no additional fine-tuning or training—a capability beyond the reach of any other method. Then with the aid of a lightweight Random Forest Classifier trained on all desired combinations, we achieve classification with extremely high accuracy, strongly outperforming the best machine learning model architectures.

Contributions

1. **Inference-Time Compositional Security** We introduce SecureLLM, a novel framework that composes a linear number of silo-specific fine-tuned LLMs at runtime to enforce information security access controls—guaranteeing that every response is drawn strictly from a user's permitted data without retraining monolithic models for each policy.
2. **Disjoint-Schema Text-to-SQL Benchmark** We propose a challenging new benchmark in which each SQL schema is fully disjoint and intersectional queries cannot be answered by any single silo. SecureLLM significantly outperforms existing compositional methods on this task, demonstrating provably robust parameterized knowledge synthesis.
3. **Overlapping-Stories Crossover-FanFic Benchmark** We compile a three sets of fan-fiction natural language stories from `ArchiveOfOurOwn.org` as well as four additional sets of overlapping stories that contain the other three story universes. This dataset is used to evaluate the effectiveness of different Anomaly Detection and Classification techniques.
4. **Perplexity-Driven Anomaly Detection** We formulate a rigorous anomaly detection task by comparing perplexity scores between a base model and a single silo-tuned model. SecureLLM surpasses classic baselines (e.g., TF-IDF, Deep-SVDD) in accurately determining out-of-distribution text.
5. **Zero-Shot Anomaly Classification** Without any additional fine-tuning, we leverage the same perplexity signals to pinpoint the exact silo source of a leak. Augmented with a lightweight Random Forest classifier, this method achieves high-precision attribution that no prior approach can match.

2 Related Work

Model Composition Recent works like LoraHub [Huang et al., 2023] composes fine-tunings. Given a target task, LoraHub selects a set of fine-tunings, Low Rank Adapters (LoRAs) [Hu et al., 2021], that are added together. However, LoraHub is designed for soft tasks, where a model already tends to perform well. Most methods like PEM Addition use arithmetic operations directly on the weights [Zhang et al., 2023] of adapters like LoRA fine-tunings [Hu et al., 2021].

Differential Privacy Many recent works discuss a range of different privacy attacks against large language models and Deep Learning models in general. Membership inference attacks [Hisamoto et al., 2020, Nasr et al., 2019, Hu et al., 2022] and training data extraction attacks that reveal personally identifiable information [Carlini et al., 2019] have been shown be to successful even when such data was mentioned in a single document and this behavior worsens with an increase in model size [Carlini et al., 2021, Zanella-Béguelin et al., 2020]. Differential Privacy provides a solution for such attacks [Abadi et al., 2016, McMahan et al., 2017, Li et al., 2021, Yu et al., 2021]. Other methods borrow inspiration from differential privacy like Confidentially Redacted Training which provably prevents memorization of the training data [Zhao et al., 2022].

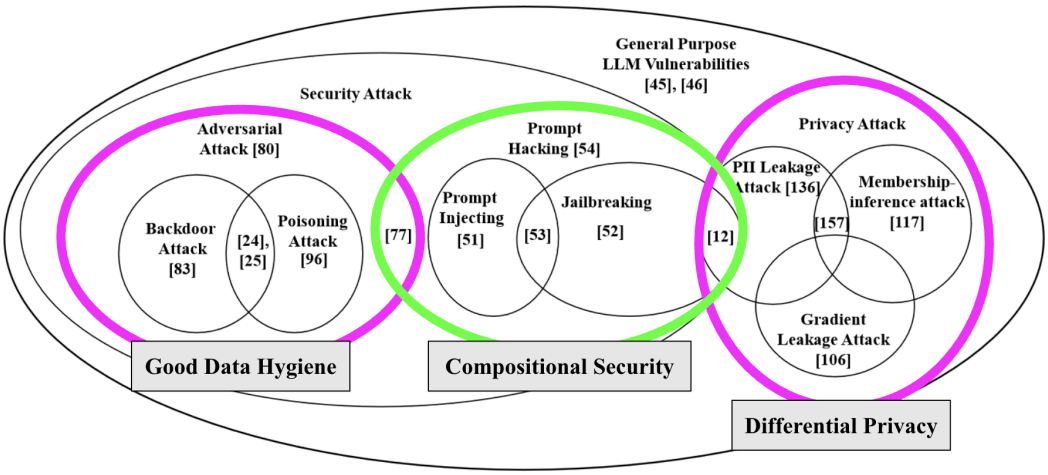

Figure 2: Figure re-used from Das et al. [2025]. We propose Inference-Time Compositional Security as a method to augment other security practices which protect against LLM vulnerabilities.

While related to the security problem we describe, Compositional Security is not designed to protect models from privacy instead, it can augment model safety along with good training-data hygene as shown in fig. 2.

Data Loss Prevention The study of Data Loss Prevention (DLP) is an information security field that specifically covers the creation of applications and methods to detect or prevent the leak of sensitive information to an unauthorized user, and is the primary focus of organizational information security systems. Most organizations use some form of classifiers with TF-IDF implementations in order to detect when a data loss or leak has occurred [Guo et al., 2022]. However, a recent paper shows that machine learning approaches outperform other methods used in DLP [Guha et al., 2021]. Because DLP is inclusive of other information security practices like firewalls or VPNs, we narrow our focus on methods of detecting leaks as the primary mechanism to prevent leaks. In this way, our DLP method is closely aligned to Anomaly Detection.

Anomaly Detection Specifically, Anomaly Detection (AD) involves identifying patterns or instances in data that deviate significantly from expected behavior. In the context of text generation, AD aims to identify generated text that does not conform to the patterns or characteristics of the training data [Chandola et al., 2009]. Thus, measuring perplexity across various combinations of information silos is a form of AD. Since perplexity reflects the model's uncertainty in predicting text, a higher perplexity score would signify detection of anomalies. One-Class Support Vector Machines (OC-SVM) are popular trained, unsupervised classifiers [Schölkopf et al., 2001]. An OC-SVM variant called Deep-SVDD boasts the best performance and combines machine learning with traditional OC-SVM [Kim et al., 2022].

Anomaly Classification Often referred to as Data Loss Prevention (DLP) classification [Hart et al., 2011], Anomaly Classification builds on outlier detection by assigning each document a "security-level" label rather than a topical category. In classic machine-learning terms, it mirrors topic modeling: given an input text, the model must predict which partition (i.e., security silo) it belongs to. We therefore evaluate SecureLLM's zero-shot classification performance against a suite of supervised baselines—LSTM, GRU, 1D-CNN, BiLSTM, and Transformer architectures—trained on the same silo-labeled data.

3 Framework

Inference-Time Composition SecureLLM takes several fine-tunings, each trained on distinct information silos, and composes them at inference time. The goal of the composed model is to answer questions about both individual silos and questions that span silos. For example, in our case, a natural-language to SQL LLM would need to be able to generate joins across the databases of multiple silos to answer complex questions that have never been seen at training time. This is a trivial task for humans, but one that challenges LLMs. We go a step further: not only must such an LLM work, it must operate through a combination of fine-tunings, i.e., not only has it never seen combinations of silos at training time, its fine tunings have only ever seen a single silo each. This challenges and defeats current fine-tuning methods. The upshot of this difficult task is that it solves several key security problems for LLMs.

Given N data silos $\{S_1, S_2, \cdots, S_N\}$ and N fine-tuned LLMs $\{M_1, M_2, \cdots, M_N\}$ where 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 composed 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 $S_i, \forall i \in T$ and should fail to answer any question about information not contained in the target silos $S_j, \forall j \notin T$ 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 while a successfully composed model should be able to answer such questions without the need of any training.

It is critical that the composed model M_T has no knowledge of any information silo that the user is not authorized to access, i.e. data silos $S_i, i \notin T$. Without this condition, a trivial solution is to train

a single model M_{All} on all data silos $\{1, \cdots, N\}$ however this approach is susceptible to leaking confidential information as the model would have knowledge of information contained in silos that users are not authorized to view and violates the third principle outlined in the introduction. We refer to M_{All} as the Exponential Model that has seen every combination and such a model is used as an insecure upper bound to performance in our experiments.

We compare the following existing methods for composing fine-tunings against our approach: LoraHub [Huang et al., 2023] and PEM Addition [Zhang et al., 2023]. We also considered Weight Averaging [Chronopoulou et al., 2023], Energy-Based Modeling [Du et al., 2020], and concatenation [Mangrulkar et al., 2022], but each performed much worse on our secure-composition task than LoraHub and PEM Addition. We therefore omit their results from the main comparisons.

Normally, LoraHub requires a two-step *COMPOSE–ADAPT* process involving additional training on the combined dataset. Because that violates the security principles we describe in the Introduction, we only perform the *COMPOSE* step. More information regarding all runtime-time compositional methods can be found in appendix B.

Ours: Maximum Difference For each adapter, we select the embeddings from each fine-tuning with the strongest response (either positive or negative) at each attention layer. In order to accomplish this, each LoRA fine-tuning is evaluated separately on input x. Then a mask of zeros with the same dimension as the output is created, h_{max}, to aggregate LoRA responses. For each LoRA fine-tuning response L_i, an element-wise comparison is made, and if the absolute values of the fine-tuning response is greater than the aggregated response, then the signed response from that fine-tuning replaces the element in the aggregated response.

Ours: Logit Composition Given fine-tunings to compose $M_1, \cdots, M_n$ and input x, we define logit composition as performing the complete forward pass for each fine-tuning independently to obtain logit probabilities. We select the maximum value of each logit.

Anomaly Detection using Perplexity We can use model perplexity starting from plain-text to evaluate the likelihood that it could come from a given composed model M_T. For instance, let M_G be a general knowledge model that was autoregressively trained on a very large dataset. Given n data silos $\{S_1, S_2, \cdots, S_n\}$, n fine-tuned LLMs $\{M_1, M_2, \cdots, M_n\}$ are created. By iteratively evaluating the perplexity of a plain-text statement h_0, we can determine if $h_0 \in S_i, \forall_i$.

Conversion from plain-text to logits used to compute perplexity is done by tokenizing the plain-text, and then assigning a value of 1 to the corresponding index i of the logit vector $X^{1\times N}$ for each token k and 0 for every other index in the logit vector.

The Compositional Perplexity Score is computed using the natural exponent of the fine-tuned cross-entropy loss minus the vanilla model cross-entropy loss. For these experiments we use Llama-2-7b as are baseline vanilla model, and we fine tune a LoRA adapter for each data silo, S_1, S_2, S_3. The vanilla model is defined as f_θ, the fine-tuned as $f_{\theta'}$, the labels as z, and cross-entropy loss as $\ell(f_\theta, z)$. The anomaly score S_{LLM} for LLM Perplexity is defined as $S_{LLM} = \frac{-e^{\ell(f_{\theta'}, z)}}{e^{\ell(f_\theta, z)}}$

Inference-Time Compositional Security SecureLLM ensures that only the fine-tuned adapters corresponding to silos a user is *actually permitted to access* are loaded into the model at runtime. Ideally, this is controlled by standard enterprise security checks (e.g. verifying user credentials and group memberships). If a user does *not* have the necessary privileges for a particular silo, that silo's adapter is never applied and it is never accessed from the user's perspective. As a result, the composed model simply lacks the relevant parameters for that silo's knowledge—and, crucially, *cannot leak it*. This arrangement differs from a single "fully-trained" model that has *all* data in its weights and must rely on imperfect guardrails to block disallowed content. Even RAG often requires carefully engineered prompts and adjacency metadata, can struggle to combine knowledge from disjoint resources, and may not preserve the same parametric "fluency" for complex tasks where the LLM must have deeper learned representations. By contrast, SecureLLM never even loads or sums the parameters for unauthorized silos, so there is no risk of prompting the model to "jailbreak" that disallowed knowledge.

CFG Generated	Baseline Exponential Model	Baseline Generalized Model	LoraHub	PEM Addition	Ours (Maximum Difference)	**Ours (Logits)**
Silos_1	0.0 (100.0%)	0.0 (98.3%)	1.9	0.9	0.4	**0.1**
Silos_2	0.0 (96.7%)	0.0 (100.0%)	2.6	0.8	0.3	**0.1**
Silos_3	0.0 (100.0%)	0.0 (100.0%)	1.2	0.7	0.2	**0.1**
$\text{Silos}_{1\cup2}$	0.0 (99.2%)	0.5 (0.0%)	1.7	0.7	0.7	**0.2**
$\text{Silos}_{1\cup3}$	0.0 (100.0%)	0.4 (1.7%)	2.0	0.7	0.6	**0.3**
$\text{Silos}_{2\cup3}$	0.0 (100.0%)	0.5 (1.7%)	2.4	0.7	0.7	**0.2**
$\text{Silos}_{1\cup2\cup3}$	0.0 (98.3%)	1.0 (0.0%)	1.8	1.0	0.9	**0.2**
$\mu \pm \sigma$	0.0 ± 0.0	0.35 ± 0.38	1.95 ± 0.47	0.78 ± 0.15	0.56 ± 0.26	$\mathbf{0.19 \pm 0.1}$

Table 1: Normalized tree-edit distance for CFG-generated question and SQL pairs with accuracy reported in parentheses (average and std. dev. only applies to normalized tree-edit distance). The exponential baseline sees all combinations of silos at training time, this is intractable and insecure, but has maximal performance. The generalization baseline sees all silos but not combinations of silos at training time, this is tractable but insecure. The other methods are used to build a SecureLLM. As described above, we do not include detailed reports on methods which underperform both LoraHub and PEM Addition. Note that our methods significantly outperform prior work. They retain all the generalization performance there is (since the generalization model sees all silos at once, while the fine-tunings each see silos separately, the generalization model should nominally perform better), even outperforming the generalization baseline.

4 Data generation

Secure-NL2SQL While there are countless other NL2SQL datasets, none specifically focus on SQL queries for disjoint and unrelated databases silos. We present Secure-NL2SQL which contains three silos of disjoint schemas pertaining to different subjects, as well as the superset of unions between those three silos for a total of seven permutations ($S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3}$). The dataset contains CFG generated questions and corresponding SQL queries across silos.

Each dataset is normalized according to 6NF [Date et al., 2003] and table and columns name are chosen to be non-trivial such that a general model would not be able to guess the table name of column name based on context given by the question. The scope of generated SQL statements is limited. All statements generated from our CFG, are SELECT statements that only contain the SQL keywords FROM, NATURAL JOIN, and WHERE. The majority of the complexity is in the WHERE clause which requires specialized knowledge about the schema along with language comprehension to properly generate based on the input question. The task for the LLM is to generate the WHERE clause of an SQL statement which answers the input question. More details on Secure-NL2SQL is given in appendix D.

Crossover-FanFic Using the publicly available data on `ArchiveOfOurOwn.org`, we compiled human-written fanfiction from three very popular fandoms to constitute three silos: Harry Potter (HP), Marvel Cinematic Universe (MCU), and DC Comics (DCU). These fandoms were chosen in particular because four crossovers between each universe also have existing and highly appraised works which constitute the union data silos to test compositionality. This dataset contains over 100,000 lines of text, which is more than enough to autoregressively fine-tune models on the concepts unique to each silo. With this dataset, we demonstrate that Perplexity in combination with Compositional Security from SecureLLM can accurately classify human-generated that the a particular has likely never seen. This dataset is only used to evaluate the performance of our derivative Data Loss Prevention and Anomaly Detection method.

5 Experiments

Inference-Time Compositional Security We first begin by obtaining individual fine-tunings that are knowledgeable in a single silo by fine-tuning a Llama-2-7b model for each silo separately. The fine-tuning results in a Low-Rank Adaptation (LoRA) for each silo which can independently be applied to the base Llama-2-7b model. These fine-tunings are combined at inference-time in our experiments. We additionally train two insecure baseline models that act as an upper-bound using LoRA; the baseline generalized model is trained on all the individual silos together and must then generalize its knowledge to the union silos for which it has not been trained on. While the baseline

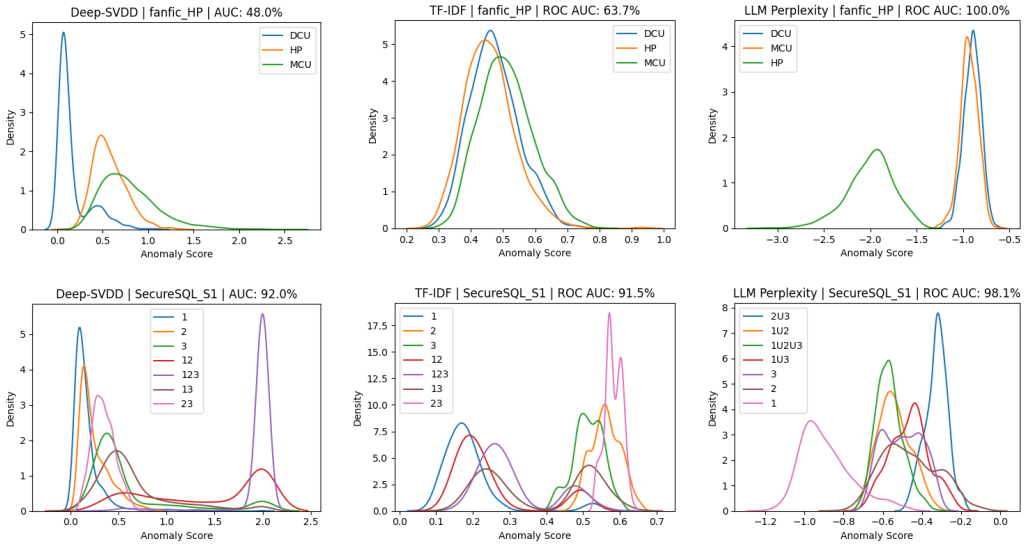

Figure 3: Anomaly Detection comparing Deep-SVDD, TF-IDF, and LLM Perplexity to detect when a leak has occured by comparing the outlier and inlier anomaly score. LLM Perplexity significantly outperforms common methods used for Anomaly Detection. Only Harry Potter Fanfic and S1 of the SQL dataset are shown for brevity. The ROC AUC scores for all data silos are in table 2; graphs for all data silos are shown in appendix F. Note: anomaly scores are relative to each method and cannot be compared directly across methods.

exponential model has been trained on all the individual silos along with the union silos. We fine-tune all models with one epoch until saturation.

While Exact Match (EM) accuracy is typically recorded for NL2SQL datasets, this metric is not granular enough to show differences in method performance. Instead, we calculate the tree-edit distance Zhang et al. [1996] between the ground query and the generated query. By computing the number of edit operations required to transition between the two, we can show how close a given generated query is to the correct query, whereas using only EM is a binary representation of correctness. Further details on the experimental setup for this section can be found in appendix C.1.

In table 1, we report performance against the Secure-NL2SQL dataset. For every probe silo combination, our methods have by far the lowest normalized tree-edit distances of all compositional methods. Our logit composition method approaches the upperbound established by the insecure Generalized Model, indicating an efficient inference-time composition with minimal losses. Furthermore, our method exceeds the capabilities of a generalized model trained on all individual silos when it comes to responses that require parameterized knowledge over the intersection of multiple data silos.

Because the SQL dataset was generated from a Context Free Grammar, we also provide results of an ablation over this dataset for the same questions translated into natural language representations of the same questions by ChatGPT and an ablation for questions where the column values are obfuscated. These can be found in appendix E.

Anomaly Detection Logit Composition outperformed all other methods, so we then compare that method with TF-IDF and Deep-SVDD for Anomaly Detection. We report the Area Under the Curve (AUC) of the computed anomaly score and its associated Density. Anomaly scores cannot be compared directly as they are relative to each method, so instead we measure the separation of anomaly scores from the inlier and outlier data silos and the area generated under that separation.

Table 2 summarizes the graphs in fig. 3 by reporting only the AUC. From these tables, we can see that LLM Perplexity outperforms popular methods for Anomaly Detection. In addition to being the highest performing method, LLM Perplexity is also extremely lightweight because it relies on training a LoRA adapter for Llama-2-7b. In contrast, Deep-SVDD requires 10x more time to train. However, TF-IDF is the fastest and most lightweight method as it does not require any training. Nevertheless, accuracy is superior for LLM Perplexity when finetuned on Llama-2-7b.

All three methods tested are trained unsupervised using 80% of the inlier data silo at train time. At test time, we provide the other 20% of the inlier data silo and 100% of the outlier data silos. For the

Secure-NL2SQL Dataset

Inlier Data Silo	Deep SVDD	TF-IDF	LLM Perplexity
S1	92.0%	91.50%	**98.1%**
S2	92.9%	79.82%	**99.7%**
S3	95.3%	86.16%	**100.0%**

CrossOverFanFic Dataset

Inlier Data Silo	Deep SVDD	TF-IDF	LLM Perplexity
DCU	26.4%	75.15%	**100.0%**
HP	48.0%	63.69%	**100.0%**
MCU	15.5%	82.24%	**100.0%**

Table 2: Area Under the Curve (AUC) for the inlier data silo is shown for each anomaly detection method. Our proposed method of using Compositional Perplexity for anomaly detection outperforms published methods for a simple SQL dataset, and significantly outperforms published methods by a wide margin for a linguistically similar set of fan fiction stories.

Secure-NL2SQL Dataset, we compute scores over each sample. In the CrossOverFanFic Dataset, because each story is continuous, we compute anomaly scores over a sliding window of 128 tokens.

DeepSVDD generates its own anomaly score and we record this raw score for the comparison. We use the standard implementation of TF-IDF from the python package `sklearn`. Finally, we implement Compositional Perplexity as described above. Further details on the experimental setup for Anomaly Detection can be found in appendix C.2.

In fig. 3, we show anomaly score-density curves for $S1$ and HP, respectively from Secure-NL2SQL and CrossOverFanFic. The separation is immediately apparent for Compositional Perplexity; there is significantly more overlap with both DeepSVDD and TF-IDF, whereas LLM Perplexity shows much stronger separation and shorter tails minimizing overlap and maximizing the AUC. This remains consistent for all silos tested. We report complete anomaly detection results in appendix F.

Anomaly Classification By extending the perpelxity comparison method used in anomaly detection to test a given sample against all desired compositions, we can achieve effective classification of anomalies. Classic Anomaly Detections methods (e.g., Deep-SVDD, TF-IDF) detect outliers but offer no source attribution, so we evaluate SecureLLM's zero-shot anomaly classification accuracy against a suite of machine learning classification methods on our two benchmarks, Secure-NL2SQL and CrossoverFanFic.

Once again, we use the Logit Composition method on finetuned LoRA adapters from Llama-2-7b. The two datasets are divided into train, validation, and test splits. From the train set, we create the fine-tuned adapters. These fine-tuned adapters can be used in isolation to generate predictions by comparing the generated cross-entropy loss for all possible compositions. We then independently normalize cross-entropy loss values for each desired composition by subtracting the mean and dividing by the standard deviation of the data.

For Zero-Shot predictions comparable to other deep-learning methods, we train a Random Forest classifier from the validation set. Finally, using the the computed dataset cross-entropy losses and the Random Forest classifier, we test on the test set to generate precision and recall scores, which are then used to report the final f1-score metric for data silo.

For comparison, we present LSTM, GRU, 1d-CNN, BiLSTM, and Transformer networks as benchmark methods along side our method using Llama-2-7b finetuned LoRA adapters. More details on the experimental setup for Anomaly Classification can be found in appendix C.3.

Exp. 2 Secure-NL2SQL	LSTM	GRU	1d-CNN	BiLSTM	Trans.	**Our Method (No-Training)**	**Our Method Zero-Shot**
Silos_1	0.61	0.65	0.87	0.52	0.26	0.59	**0.96**
Silos_2	0.61	0.73	0.83	0.58	0.46	0.91	**1.00**
Silos_3	0.66	0.93	0.92	0.88	0.61	0.67	**1.00**
$\text{Silos}_{1\cup2}$	0.47	0.46	0.87	0.43	0.54	0.82	**0.97**
$\text{Silos}_{1\cup3}$	0.46	0.62	0.80	0.37	0.33	0.55	**0.93**
$\text{Silos}_{2\cup3}$	0.61	0.74	0.80	0.53	0.48	0.50	**0.96**
$\text{Silos}_{1\cup2\cup3}$	0.38	0.65	0.87	0.52	0.46	0.21	**0.96**
Accuracy	0.54	0.68	0.85	0.55	0.46	0.60	**0.97**

Table 3: Anomaly Classification for Secure-NL2SQL dataset. F1-Scores are reported for each method as well as the Weighted Average Accuracy along the bottom line. All methods except Our Method (No-Training) must be trained on all desired silos before inference time. On overall weighted accuracy our untrained method outperforms the LSTM, BiLSTM, and Transformer. Our Zero-Shot method clearly outperforms all other methods across all metrics.

Exp. 2 X-overFanFic	LSTM	GRU	1d-CNN	BiLSTM	Trans.	**Our Method (No-Training)**	**Our Method Zero-Shot**
HP	0.10	0.12	0.20	0.16	0.16	0.70	**0.99**
MCU	0.11	0.07	0.34	0.15	0.09	0.39	**0.98**
DCU	0.03	0.02	0.10	0.04	0.16	0.15	**0.98**
HP-MCU	0.59	0.57	0.59	0.58	0.59	0.00	**0.83**
HP-DCU	0.05	0.03	0.07	0.07	0.07	0.03	**0.20**
MCU-DCU	0.18	0.11	0.14	0.20	0.20	0.10	**0.64**
HP-MCU-DCU	0.03	**0.05**	**0.05**	0.02	0.02	0.00	**0.05**
Accuracy	0.33	0.31	0.36	0.35	0.35	0.14	**0.75**

Table 4: Anomaly Classification for CrossoverFanFic dataset. We report the F1-Scores for each method as well as the Weighted Average Accuracy along the bottom line. All methods except Our Method (No-Training) are methods trained on all data silos at once. Our Zero-Shot method clearly outperforms all other methods across all metrics, except GRU and 1d-CNN for the union of all three story universes.

Shown in table 3, our method is capable of near perfect Anomaly Classification for the Secure-NL2SQL dataset. Results for a sliding window of 128 tokens is shown in table 4. In contrast, CrossoverFanFic appears to be a much more difficult task to accurately identify, particularly when the source samples comes from the crossover of three separate stories. This difference could be explained by the fact that the Secure-NL2SQL contains very unique key terms that link to one specific data silo, whereas the CrossoverFanFic dataset is a lot more ambiguous over a small 128 token context window (an ablation of different sliding window sizes shown in appendix H). However, despite these challenges identifying one of the datasets, our method doubled the best accuracy when compared to the other methods tested. This remains true for all silos tested. We report complete Anomaly Classification results for all silos in appendix G.

6 Conclusion

In this work, we reframe LLM security as an inference-time compositional problem, transplanting the rigor of information security access controls into the realm of large language models. By fine-tuning a linear number of silo-specific adapters and dynamically composing them only when a user's credentials permit, SecureLLM provides a provable guarantee: a model never loads parameters for unauthorized data, and thus cannot leak it. Crucially, despite never training on any combination of silos, our Logit- and Maximum-Difference composition operators achieve SQL generation performance on par with—or exceeding—a baseline model that indiscriminately sees all data.

We further introduce Compositional Perplexity, a lightweight yet powerful anomaly detector, and show that it outperforms classical DLP methods (TF-IDF, Deep-SVDD) by wide margins on both synthetic NL-to-SQL and real-world fan-fiction benchmarks. Without additional gradient updates, the same perplexity signals support zero-shot anomaly classification—with a simple Random Forest back end—accurately attributing each sample to its originating silo.

Noteably, SecureLLM requires no changes to LLM architectures; it integrates seamlessly with existing enterprise credential systems, and scales linearly rather than exponentially in the number of security domains. As such, it pioneers a new direction for enabling high-assurance, regulated-environment LLM applications. We anticipate several promising extensions—formalizing compositional security under adversarial threat models, generalizing from database schemas to document collections, and exploring richer compositional primitives—to broaden both the theoretical foundations and practical impact of compositional LLM security.

Limitations Further investigation is required to characterize logit composition's behavior at scale and for different tasks like document summarization and Q&A. However, Based on results from PEM Addition and LoraHub, there is a strong hypothesis that SecureLLM will likewise extend very easily to other tasks [Hu et al., 2021, Zhang et al., 2023]. Additionally, more experiments are also needed to evaluate its efficacy and efficiency as the number of silos and model size, to quantify any computational or memory overhead, and to establish formal guarantees on composition fidelity.

Reproducibility All code and data required to reproduce our work is provided in the online supplement which will be made public under an open source license. Further information regarding limitations, ethics, and computing requirements are given in appendix A.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Three claims are described in the abstract, and five contributions are given in the introduction. Each one is described in detail in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are described in the second-to-last paragraph of the conclusion. Limitations are described in further detail in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The results of this paper come from experimental results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The appendix describes in full detail everything needed to reproduce results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All data and code is provided in the supplement of this submission.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: This paper describes all details to reproduce results in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Individual silo results are given as the average over very large datasets, then result are aggregated on the experiment level with 1-error sigma stated where necessary.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All compute resource information is provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The NeurIPS Code of Ethics has been reviewed by the authors and this paper is in compliance with the guidance therein.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is discussed under the ethics section of the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The data provided is either generated or from publicly available sources. No pretrained models are provided.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Attribution to ArchiveOfOurOwn.org authors is presented in the provided dataset.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Details regarding the assets are provided in the appendix and code documentation is provided in the supplemental information of this submission.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Crowdsourcing was not used for this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Human subject research did not occur as part of this project.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Additional Information

A.1 Limitations

We disentangle and address a very specific slice of LLM safety, one that is often commingled with a larger story about safety. SecureLLM only concerns itself with security in the traditional sense: quantized access permissions to data. It relies on traditional security techniques to manage access permissions. There is a widely held belief that LLM security is a totally disjoint new field, but as we show, with the SecureLLM approach we can reduce many of those security problems back to traditional access permission issues. Many security problems of LLMs are manageable through traditional means when one can assume that only vetted actors have access, just as they are with current document storage systems. The same systems which ensure patient privacy, financial privacy, and that manage secret information today can be used to manage collections of fine-tunings, and the same supervision methods can trace access to the LLM.

From this perspective SecureLLM solves data leakage and prompt injection attacks, in the same sense that traditional security solves data leakage: those without permissions cannot access this information, and those with permissions have full access with training and supervision. Although, in the future one might imagine extensions that provide more fine-grained permissions. Organizations are already set up for this form of security, both for managing the user permissions and for the associated documents, making the deployment of SecureLLM straightforward. In settings where this structure is not available or not appropriate, SecureLLM is not directly applicable, but those scenarios never had meaningful security to begin with.

Explicitly out of scope are other notions of safety and security. For example, the LLM may still fabricate information, produce toxic or biased results, follow guidance that it should not, etc. The only mitigation that we offer is that only a user that has permissions to that data will be impacted directly; the user will still need appropriate training about the limitations and dangers of LLMs.

When it comes to simple anomaly detection, TF-IDF is extremely lightweight and easily understandable, two important properties for enterprise applications. For this reason, it is likely to continue to dominate enterprise application of anomaly detection for security applications. However, as results improve for Inference-Time Compositional Security for LLMs and pressure increases to use LLMs in enterprise workflows, the adoption of methods like compositional security are more likely.

A.2 Ethics

Compositional Security used in SecureLLM could pose some ethical issues. A clear extension of the method is to recognize secure data instead of answering questions about it. One could do this by selecting silos and checking which silos are required to have high likelihood for some statement which could be used for surveillance to detect sensitive topics. At the moment, surveillance is limited by the need to process a deluge of data; practically this results in mostly processing metadata. By tuning to specific secret data and then monitoring for it, it is possible to detect and track pieces of information with unprecedented accuracy. As with many other dual-use technologies, we hope that network appliances of this kind will not be used for nefarious purposes.

A.3 Reproducibility

We use NVIDIA Titan RTX 24GB VRAM GPUs for all our experiments. SecureLLM requires approximately 30 minutes of training per silo, other machine learning methods like LSTM, GRU, etc. 5 minutes each. For each run, approximately 20 GB of VRAM is needed as we use half precision for all training and inference. Each PEFT can be trained and inferenced on one GPU. We estimate a total of 10 GPU hours for everything except Deep-SVDD which used a total of 24 hours of GPU time.

SecureLLM. We first begin by obtaining individual fine-tunings that are knowledgeable in a single silo by fine-tuning a Llama-2-7b model for each silo separately. The fine-tuning results in a Low-Rank Adaptation (LoRA) for each silo which can independently be applied to the base Llama-2-7b model. Once the individual LoRA fine-tunings are obtained we compose them using one of several compositional schemes with the requirement that the composition happens at inference time with no additional training. We additionally train two insecure baseline models that act as an upper-bound using LoRA, the baseline generalized model is trained on all the individual silos together

and must then generalize it's knowledge to the union silos. While the baseline exponential model also breaks privacy guarantees by training on all the individual silos along with the union silos, the term exponential refers to the fact that training such a model while preserving privacy would mean that $\mathcal{O}(2^N)$ models would need to be trained where N is the number of silos in the database. Both baseline models are considered insecure as there is no method of removing knowledge about certain silos at inference time when the user does not have the sufficient credentials unlike our proposed SecureLLM method which can remove and add fine-tunings with each silo's knowledge at inference time. We fine-tune all models with one epoch until saturation (achieving near 100% accuracy on the CFG validation set) using a frozen Llama-2 7B [Touvron et al., 2023] with a trainable LoRa fine-tuning [Hu et al., 2021] using LoRa parameters $r = 8, \alpha = 32$ and a dropout [Srivastava et al., 2014] of 0.1, an Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.0002, a batch size of 32, and a weight decay of 0.002.

- **LSTM**. 100 embedding size, 128 hidden size, learning rate of $1e^{-3}$
- **GRU**. 100 embedding size, 128 hidden size, learning rate of $1e^{-3}$
- **1d-CNN**. 100 embedding size, 128 filters, filter sizes (3,4,5), learning rate of $1e^{-3}$
- **BiLSTM**. 100 embedding size, 128 hidden size, learning rate of $1e^{-3}$
- **Transformer**. 104 embedding size, 8 heads, 2 layers, learning rate of $1e^{-3}$

All code and data required to reproduce our work will be provided at this GitHub repository under the MIT license in the supplemental materials of this submission.

B Compositional Security

SecureLLM takes several fine-tunings each trained on distinct information silos and composes them at inference time. The goal of the composed model is to answer questions about both individual silos and questions that span silos. For example, in our case, a natural-language to SQL LLM would need to be able to generate joins across the databases of multiple silos to answer complex questions that have never been seen at training time. This is a trivial task for humans, but one that challenges LLMs. We go a step further: not only must such an LLM work, it must operate through a combination of fine-tunings, i.e., not only has it never seen combinations of silos at training time, its fine tunings have only ever seen a single silo each. This challenges, and defeats, current fine-tuning methods. The upshot of this difficult task is that it solves several key security problems for LLMs.

Given N data silos $\{S_1, S_2, \cdots, S_N\}$ and N fine-tuned LLMs $\{M_1, M_2, \cdots, M_N\}$ where 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 composed 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 $S_i, \forall i \in T$ and should fail to answer any question about information not contained in the target silos $S_j, \forall j \notin T$ 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 while a successfully composed model should be able to answer such questions without the need of any training.

It is critical that the composed model M_T has no knowledge of information silo that the user is not authorized to access, i.e. data silos $S_i, i \notin T$. Without this condition, a trivial solution is to train a single model M_{All} on all data silos $\{1, \cdots, N\}$ however this approach is susceptible to leaking confidential information as the model would have knowledge of information contained in silos that users are not authorized to view and thus is not a valid approach. This approach is also problematic for scenarios that employ security through contradiction, in that some silos may directly contradict information in another silo in order to protect sensitive information (SecureLLM could potentially solve this by applying weights to silos of higher confidentiality). We refer to M_{All} as the Exponential Model that has seen every combination and such a model is used as an insecure upper bound to performance in our experiments.

An alternative to composing fine-tunings while also preserving privacy would be to create an exponential number of models, one for the powerset of information silos. This would maximize performance

and minimize the amount of generalization needed, as long as one had a way to automatically generate cross-silo questions, perhaps with another LLM. This is obviously impractical. In essence, our method provides the advantages of the exponential approach but with linear storage and training runtime.

B.1 Inference-Time Compositional Methods

We discuss several existing methods, none of which perform well. Finally we describe two new methods that do perform well with one clear winner.

LoraHub The LoraHub method for composition introduced by Huang et al. [2023] is a two-step process involving element-wise summation of LoRA fine-tunings (*COMPOSE*), and then learning weight optimizations via gradient-free methods to apply to each fine-tuning (*ADAPT*). For this paper, we do not implement the *ADAPT* stage because weights for every possible combination would need to be learned, and we could not say that this process is completed at inference time.

In our evaluation, we do not perform *ADAPT* for two reasons. It does not modify the performance of the system because our evaluation set is balanced, i.e., ADAPT would make the model better at one silo than another but not change the composition of silos. Additionally, ADAPT would be expensive in practice and would need to be run per user unless one wanted to save an exponential number of ADAPT combinations for the entire powerset of silos.

The *COMPOSE* method consists of adding the encoder/decodder components of each Lora to act as one encoder/decoder pair. Given $m_i = A_i B_i$, the combined fine-tuning, $\hat{m}$ and $\hat{m}(x)$ on input x is obtained by

$$\hat{m} = (B_1 + \cdots + B_n)(A_1 + \cdots + A_n), \tag{1}$$

$$\hat{m}(x) = (B_1 + \cdots + B_n)(A_1 + \cdots + A_n)(x). \tag{2}$$

We observe that LoraHub performs poorly on the secure composition task. The authors warn that combining too many fine-tunings can lead to poor performance, however this cannot be the source of poor performance as we compose only up to three LoRAs.

PEM Addition The summation method introduced by Zhang et al. [2023] is similar to LoraHub, however, instead of summing the embeddings of the encoder and decoder prior to receiving the input x, one executes each fine-tuning independently at the attention-layer level, and then adds the result. This version of summed composition shows improved performance over LoraHub.

Average of Adapter Weights Computing the simple average of each Lora fine-tuning response, as suggested by Chronopoulou et al. [2023], $\sum_{i \to n} \frac{L_i}{n}$, produced compositions that were 50% less effective than PEM Addition in initial informal tests.

Variations of Energy Based Modeling Du et al. [2020] proposes a disjunctive composition process based on Energy Based Modeling, $-logsumexp(-E_1(x), -E_2(x), \cdots)$. Every variation tried performed significantly worse than PEM Addition, and upon closer inspection, this process substantially distorts encoder and decoder embeddings.

Adapter Concatenation The Mangrulkar et al. [2022] library implements weight concatenation, however we found that concatenating LoRA encoder/decoder fine-tunings performed significantly worse than PEM Addition.

B.2 Our Methods

Maximum Difference The intuition behind this method is to select the embeddings from each fine-tuning with the strongest response (either positive or negative) at each attention layer. In order to accomplish this, each LoRA fine-tuning is evaluated separately on input x. Then a mask of zeros with the same dimension as the output is created, h_{max}, to aggregate LoRA responses. For each LoRA fine-tuning response L_i, an element-wise comparison is made, and if the absolute values of the fine-tuning response is greater than the aggregated response, then the signed response from that fine-tuning replaces the element in the aggregated response.

Logit Composition Given fine-tunings to compose $M_1, \cdots, M_n$ and input x, we define logit composition as performing the complete forward pass for each fine-tuning independently to obtain logit

Algorithm 1 Element-wise Maximum Difference

$h_{\text{max}} \leftarrow \text{zeros\_like}(x)$
for $i = 0$ **to** $n - 1$ **do**
  $h_{\text{mask}} \leftarrow \text{zeros\_like}(h_{\text{max}})$
  $L_{\text{mask}} \leftarrow \text{zeros\_like}(h_{\text{max}})$
  for all elements e in h_{max} **do**
    if $|h_{\text{max}}[e]| > |L_i|$ **then**
      $h_{\text{mask}}[e] \leftarrow 1$
    else if $|h_{\text{max}}[e]| < |L_i|$ **then**
      $L_{\text{mask}}[e] \leftarrow 1$
    end if
  end for
  $h_{\text{max}} \leftarrow h_{\text{max}} \cdot h_{\text{mask}} + L_i \cdot L_{\text{mask}}$
end for
$h = \textit{Attention}(x) + h_{\text{max}}$

probabilities. We select the maximum value of each logit. One could instead sum logits for each fine-tuning. We found little difference between the two implementations, although the sum may have issues as the number of fine-tunings increases.

C Experimental Setup

C.1 Inference-Time Composition

To demonstrate the capabilities of model composition at inference-time, we first begin by obtaining individual fine-tunings that are knowledgeable in a single silo by fine-tuning a Llama-2-7b model for each silo separately. The fine-tuning results in a Low-Rank Adaptation (LoRA) for each silo which can independently be applied to the base Llama-2-7b model. Once the individual LoRA fine-tunings are obtained we compose them using one of several compositional schemes with the requirement that the composition happens at inference time with no additional training. We additionally train two insecure baseline models that act as an upper-bound using LoRA, the baseline generalized model is trained on all the individual silos together and must then generalize it's knowledge to the union silos. While the baseline exponential model also breaks privacy guarantees by training on all the individual silos along with the union silos, the term exponential refers to the fact that training such a model while preserving privacy would mean that $\mathcal{O}(2^N)$ models would need to be trained where N is the number of silos in the database. Both baseline models are considered insecure as there is no method of removing knowledge about certain silos at inference time when the user does not have the sufficient credentials unlike our proposed SecureLLM method which can remove and add fine-tunings with each silo's knowledge at inference time. We fine-tune all models with one epoch until saturation (achieving near 100% accuracy on the CFG validation set) using a frozen Llama-2 7B [Touvron et al., 2023] with a trainable LoRa fine-tuning [Hu et al., 2021] using LoRa parameters $r = 8, \alpha = 32$ and a dropout [Srivastava et al., 2014] of 0.1, an Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.0002, a batch size of 32, and a weight decay of 0.002.

While Exact Match (EM) accuracy is typically recorded for NL2SQL datasets, we found this metric to not be granular enough to show differences in method performance. Instead, we calculate the tree-edit distance Zhang et al. [1996] between the ground query and the generated query. By computing the number of edit operations required to transition between the two, we can show how close a given generated query is to the correct query, whereas using only EM is a binary representation of correctness.

We report the results of the two insecure baseline models along with the secure composition ($M_1 \oplus M_2 \oplus M_3$ where M_i was trained on Silo_i) using multiple compositional methods (as described in Framework) including our best method (Framework: Logit Composition) with and without using 6NF-like database normalization, which is equivalent to the scenario where a user has credentials to access $T = \{1, 2, 3\}$. We note that neither the baseline generalized model nor the secure compositions have seen the union Silos ($S_{1\cup2}$, $S_{1\cup3}$, $S_{2\cup3}$, and $S_{1\cup2\cup3}$) and that only the exponential baseline model has been trained on those silos. The performance of the composed fine-tunings on the individual

Silos would give an indication as to whether the resulting composition is able to retain the knowledge of each individual fine-tuning from each separate Silo; This performance is expected to be traded off for privacy while the better compositional methods mitigate the extent of this trade off and maintain maximal privacy. The performance on the union Silos indicates whether the composed fine-tunings are able to successfully generalize knowledge from the individual fine-tunings which is an essential component in answering questions that no individual fine-tuning or silo can answer.

We emphasize that no additional training or gradient updates are required for our secure composition which is the critical factor that prevents the exponential training runs required to achieve a proper composition for every possible subset of the available silos given the users credentials.

C.2 Anomaly Detection

For all methods tested in anomaly detection (AD), we report the Area Under the Curve (AUC) of the computed anomaly score and its associated Density. Anomaly scores cannot be compared directly as they are relative to each method, so instead we measure the separation of anomaly scores from the inlier and outlier data silos and the area generated under that separation.

All three methods tested are trained unsupervised using 80% of the inlier data silo at train time. At test time we provide the other 20% of the inlier data silo and 100% of the outlier data silos.

SecureSQL Dataset . We compute anomaly scores over each sample in the SecureSQL dataset as they are independent of one another and have defined start and stop points within the sample.

CrossoverFanFic Dataset . We compute anomaly scores over a sliding window in the Crossover-FanFic dataset because each story is continuous and cannot be divided intentionally into smaller sections. We found that a sliding window of 128 tokens is ideal, and present an ablation of each methods as the sliding windows is reduced to 8 tokens.

Deep-SVDD generates it's own anomaly score and we record this raw score for the comparison. Deep-SVDD attempts to learn unsupervised differences between datasets by finding a hyperplane and hypersphere enclosing most of the training data and reports the anomaly score accordingly [Kim et al., 2022]. If there is no higher dimensional separation between a majority of individual tokens (like we would expect in natural language fan fiction), then this separation may not be easily discoverable.

The anomaly score S_{DSVDD} measures the distance between the center c and a data point x_i using learned weights $\mathcal{W}$ [Kim et al., 2022]

$$S_{DSVDD} = ||\phi(x_i; \mathcal{W}) - c||^2 \tag{3}$$

TF-IDF is trivially implemented with the python package `sklearn` where we compute the anomaly score S_{TF_IDF} for term t and document d as

$$S_{TF\_IDF} = 1 - tf\_idf(t, d) \tag{4}$$

$$= 1 - \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \times (\log\left(\frac{n}{df(t)}\right) + 1) \tag{5}$$

LLM Perplexity score is implemented by computing the natural exponent of the fine-tuned cross-entropy loss minus the vanilla model cross-entropy loss. For these experiments we use Llama-2-7b as are baseline vanilla model, and we fine tune a LoRA adapter for each data silo, S_1, S_2, S_3. The vanilla model is defined as f_θ, the fine-tuned as $f_{\theta'}$, the labels as z, and model cross-entropy loss as $\ell(f_\theta, z)$. The anomaly score S_{LLM} for LLM Perplexity is defined as

$$S_{LLM} = \frac{-e^{\ell(f_{\theta'}, z)}}{e^{\ell(f_\theta, z)}} \tag{6}$$

C.3 Anomaly Classification

Our Method.

We implement SecureLLM using finetuned LoRA adapters from Llama-2-7b. The two datasets are divided into train, validation, and test splits. For SecureSQL, 1000 samples from S_1, S_2, and S_3 are used to train a finetuned adapter for each data silo, giving us M_1, M_2, and M_3. These fine-tuned adapters can be used in isolation to generate predictions by comparing the generated cross-entropy loss for all possible compositions. We then independently normalize cross-entropy loss values for each desired composition by subtracting the mean and dividing by the standard deviation of the data.

For Zero-shot predictions comparable to other methods, we need to train a Random Forest classifier using with 500 additional samples from each independent silo (S_1, S_2, S_3) as well as 700 samples from the unions of these silos ($S_{1\cup2}, S_{2\cup3}, S_{1\cup3}, S_{1\cup2\cup3}$). Finally, using the the desired cross-entropy losses and the Random Forest classifier, we test our data on 500 final samples of each independent silo (S_1, S_2, S_3) and 300 samples from the unions of these silos ($S_{1\cup2}, S_{2\cup3}, S_{1\cup3}, S_{1\cup2\cup3}$) to generate precision and recall scores which are then used to report the final f1-score metric for data silo.

For comparison, we present LSTM, GRU, 1d-CNN, BiLSTM, and Transformer networks as benchmark methods along side our method using Llama-2-7b finetuned LoRA adapters. Shown in tables 3 and 4, our method is capable of near perfect Anomaly Classification for the Secure-NL2SQL dataset. In contrast, CrossoverFanFic appears to be a much more difficult task to accurately identify, particularly when the source samples comes from the crossover of three separate stories. This difference could be explained by the fact that the Secure-NL2SQL contains very unique key terms that link to one specific data silo, whereas the CrossoverFanFic dataset is a lot more ambiguous over a small 128 token context window. However, despite these challenges identifying one of the datasets, our method doubled the best accuracy when compared to the other methods tested.

For the CrossoverFanFic dataset, each silo has very different words counts across silos and their crossover stories (or story intersections). Thus we take a raw percentage instead. For training, we use 80% of the samples from Harry Potter (HP), Marvel (MCU), and DC Comics (DCU) to finetune our three Llama-2-7b LoRA adapters. Again, this is all that is needed to generate unsupervised predictions, however for the supervised predictions, we use a similar split of 10% independent silos and 70% intersection silos to train the Random Forest classifier, and report final results on the last 10% and 30% of independent and intersection silos respectively.

Other Methods. We test several machine learning methods along side our SecureLLM Llama-2-7b finetuned LoRA adapters. These methods are LSTM, GRU, 1d-CNN, BiLSTM, and Transformer networks. The same splits that were used for Our Method was used for these comparisons, however, the first stage of training differs slightly. Instead of training three separate models, we aim to provide labels for each silos at train time and train one single model to predict independent silos. Then this single model is used train the supervised Random Forest classifier across the larger set of seven classes that includes the intersection silos. This gives us comparable precision and recall scores which are then used for the final f1-scores for each data silo and their intersections.

The same splits that were used for Our Method was used for these comparisons, however, the first stage of training differs slightly. Instead of training three separate models, we aim to provide labels for each silos at train time and train one single model to predict independent silos. Then this single model is used train the supervised Random Forest classifier across the larger set of seven classes that includes the intersection silos. This gives us comparable precision and recall scores which are then used for the final f1-scores for each data silo and their intersections.

D Secure-NL2SQL Dataset

We generate SQL databases, one per silo, with 2-3 tables per database, that share columns which can be joined together between databases. However, the databases are otherwise disjoint and contain different topics. For each database we generate natural language questions along their equivalent SQL. Then, we generate questions and SQL pairs that span pairs and triples of databases. Two methods are used to generate these pairs: a CFG (see fig. 4) and ChatGPT 4[1] (see fig. 5). The CFG generates both the SQL and the question in parallel. We do this at scale, with 100,000 pairs per silo or combination of silos. For the unioned questions, we also generate 300 pairs per silo or combination of silos.

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 ($S_{1\cup2}$)

Figure 4: Examples of input/output pairs of a question paired with the target SQL query which are unconditional samples from a Context-Free Grammar.

Our goal is to automatically create a challenging dataset for compositions of silos. While there are countless other NL2SQL datasets, none specifically focus on SQL queries for disjoint and unrelated databases silos. Secure-NL2SQL contains three silos of disjoint schemas pertaining to different subjects, as well as the superset of unions between those three silos for a total of seven permutations ($S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3}$). The dataset contains automatically generated questions and corresponding SQL queries across each silos.

Each dataset is normalized according to 6NF Date et al. [2003] and table and columns name are chosen to be non-trivial such that a general model would not be able to guess the table name of column name based on context given by the question. The scope of generated SQL statements is limited. All statements generated from our CFG, are SELECT statements that only contain the SQL keywords FROM, NATURAL JOIN, and WHERE. The majority of the complexity is in the WHERE clause which requires specialized knowledge about the schema along with language comprehension to properly generate based on the input question. The task for the LLM is to generate the WHERE clause of an SQL statement which answers the input question.

[1]Version of ChatGPT 4 from early 2024 was simply known as "ChatGPT 4" which was prior to the deployment of ChatGPT-4o and is no longer available from OpenAI

Q: What is the average age of instructors who are aged 72 or older or teach art at level 9? 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) that was rephrased with ChatGPT

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.sloth)
FROM inventory INNER JOIN store ON
    store.bear = inventory.bear
WHERE inventory.pony = 0
    AND (store.alpaca = 'NY'
     OR store.alpaca = 'CA'
     OR store.alpaca = 'MA')
    AND store.raccoon >= 2
```

(b) Sample from Silo 2 (S_2) with obfuscated column names

Figure 5: Examples of input/output pairs of a question paired with the target SQL query which (a) are from the ChatGPT rephrased silos and (b) use column names obfuscated by an arbitrary but consistent mapping.

There are countless other natural-language-to-SQL datasets out there, which we do not aim to replace. As such, we focus specifically on within and between silo questions. We also cover only a portion of SQL. We do not aim to exhaustively test how well models understand SQL, we aim to understand how well models generalize their knowledge from questions about individual silos to questions that span silos. As such, we consider only a subset of SQL which is otherwise an imposingly complex language.

We automatically generate SQL databases, one per silo, with 2-3 tables per database, that share columns which can be joined together both within and between databases. However, the databases are otherwise disjoint and contain different topics. For each database we generate natural language questions along their equivalent SQL. Then, we generate questions and SQL pairs that span pairs and triples of databases. Two methods are used to generate these pairs: a CFG (see fig. 4) and ChatGPT 4 (see fig. 5). The CFG generates both the SQL and the question in parallel. We do this at large scale, with 100,000 pairs per silo or combination of silos. To ensure that our results scale to more realistic queries we also generate 300 pairs per silo or combination of silos.

We limit the scope of generated SQL statements. All statements generated from our CFG, are SELECT statements that only contain the SQL keywords FROM, NATURAL JOIN, and WHERE. The majority of the complexity is in the WHERE clause which requires specialized knowledge about the schema along with language comprehension to properly generate based on the input question. The task for the LLM is to generate the WHERE clause of an SQL statement which answers the input question.

We introduce a useful normalization, for which we provide ablations in the results section. This normalization is closely related to 6NF [Date et al., 2003] by ensuring table and columns are not "guessable" by a non-fine-tuned model based on context provided in the prompt. In general, this transformation could help all SQL LLMs, and is bidirectional. As described in the results section, our composition methods are far superior irrespective of this normalization, but we believe it is a valuable observation that is likely to lead to many more LLM-specific normalizations as they become serious consumers of SQL.

From each SQL schema we randomly generate an instance of that SQL database. For each sample, the LLM's output is assembled into an SQL statement then executed on the database to obtain the query results, if the query results are exactly equivalent to the query result of the ground truth SQL statement then the LLM's output is considered correct. Furthermore, we ensure that each query in training and validation responds with at least one record to avoid trivial false positives. Given that natural language questions can lead to long queries, this is a high bar, as even the smallest mistake leads to zero performance.

Determining if a generated query is correct is a non-trivial problem as the problem of determining if two SQL statements are semantically equal is undecidable [Chu et al., 2017]. Accuracy was determined by empirically executing the generated query on a database and comparing the results to the ground query. Using a single database increases the susceptibility of false positives, thus we employ a method inspired by *test suite accuracy* [Zhong et al., 2020] by instantiating several databases with diversified data, which reduces the probability of false positives. However, there are issues with using accuracy as a metric as a single incorrect token would render the entire generated query as incorrect. Thus

We employ a second score that parses the generated query conditions into a tree then calculates the tree-edit distance Zhang et al. [1996] between the ground query and the generated query. This is the number of edit operations required to transition between the two, which are normalized by the number of nodes in the ground tree. These are averaged across all queries for a silo or collection of silos. This edit distance score provides a far more fine-grained view into the performance of models, fine-tunings, and compositions of fine tunings.

E Inference-Time Composition Data Ablation

GPT Generated	Baseline Exponential Model	Baseline Generalized Model	LoraHub	PEM Addition	Ours (Maximum Difference)	**Ours (Logits)**
Silos_1	0.0 (87.5%)	0.1 (79.2%)	2.0	1.1	0.5	**0.2**
Silos_2	0.2 (61.7%)	0.2 (56.7%)	2.8	1.0	0.5	**0.3**
Silos_3	0.1 (56.7%)	0.2 (51.7%)	1.4	1.1	0.5	**0.2**
$\text{Silos}_{1\cup2}$	0.2 (29.2%)	0.4 (0.0%)	1.5	0.9	0.6	**0.4**
$\text{Silos}_{1\cup3}$	0.1 (33.3%)	0.3 (3.3%)	2.2	0.6	0.5	**0.3**
$\text{Silos}_{2\cup3}$	0.1 (50.0%)	0.3 (2.5%)	1.9	0.6	0.5	**0.2**
$\text{Silos}_{1\cup2\cup3}$	0.2 (20.8%)	0.4 (0.0%)	2.0	0.7	0.6	**0.2**
$\mu \pm \sigma$	0.15 ± 0.07	0.28 ± 0.14	1.96 ± 0.47	0.84 ± 0.22	0.52 ± 0.05	$\mathbf{0.27 \pm 0.06}$

Table 5: Results on the ChatGPT-paraphrased questions. See table 1 for a detailed explanation. Our method continues to outperform all others, and again outperforms the generalization baseline. Scaling to realistic queries still favours our approach.

Obfuscated Generated	Baseline Exponential Model	Baseline Generalized Model	LoraHub	PEM Addition	Ours (Maximum Difference)	**Ours (Logits)**
Silos_1	0.0 (99.2%)	0.0 (94.2%)	1.8	1.1	0.5	**0.2**
Silos_2	0.0 (92.5%)	0.0 (100.0%)	3.1	1.4	0.5	**0.2**
Silos_3	0.0 (100.0%)	0.0 (100.0%)	0.9	0.8	0.5	**0.1**
$\text{Silos}_{1\cup2}$	0.0 (98.3%)	0.4 (0.0%)	1.3	1.4	0.7	**0.3**
$\text{Silos}_{1\cup3}$	0.0 (77.5%)	0.6 (1.7%)	1.6	2.5	1.0	**0.5**
$\text{Silos}_{2\cup3}$	0.0 (100.0%)	0.4 (1.7%)	1.9	2.5	0.9	**0.3**
$\text{Silos}_{1\cup2\cup3}$	0.0 (80.0%)	0.7 (0.8%)	1.6	2.2	1.1	**0.5**
$\mu \pm \sigma$	0.01 ± 0.01	0.31 ± 0.3	1.73 ± 0.69	1.7 ± 0.7	0.73 ± 0.26	$\mathbf{0.29 \pm 0.15}$

Table 6: Following table 1 while obfuscating the table names. Our methods continue to perform well showing that they are not taking advantage of a trivial solution. For real-world applications, one would likely use a much larger baseline model. This would improve the absolute execution scores, which would method would benefit from since it retains the performance of the underlying model in this challenging compositional task.

In table 1 we report performance for the CFG-generated data. Note that for every probe silo combination our methods have by far the lowest tree edit distances. Even without the database normalization described above, our methods outperform all others in every case. With the database normalization our method retains all the performance that exists, i.e., it nearly always matches the tree edit distance of the baseline generalized model.

One might wonder if these results are merely an artifact of the CFG-based approach. When replicating the same experiment with sentences rephrased by ChatGPT, see table 5, we come to the same conclusions. LoraHub and PEM Addition, along with all prior methods we attempted significantly underperform our approach. Note that this is an extremely challenging test set as the ChatGPT paraphrases are only used for testing, not for training.

To guard against a potential trivial solution to this problem, we also introduce a column-name obfuscated version. A model that is good at guessing a likely name for a table based on the entities it refers to might otherwise get a leg up. We are interested in the ability of models to retain compositional reasoning, rather than circumventing the task. In any case, in real world conditions column names are often rather complex. In table 6 each column is given an arbitrary but stable and coherent name, in this case an animal. Relative to table 1 our method loses little performance, meaning that it encourages compositional reasoning.

We report the results of the two insecure baseline models along with the secure composition ($M_1 \oplus M_2 \oplus M_3$ where M_i was trained on Silo_i) using multiple compositional methods (as described in

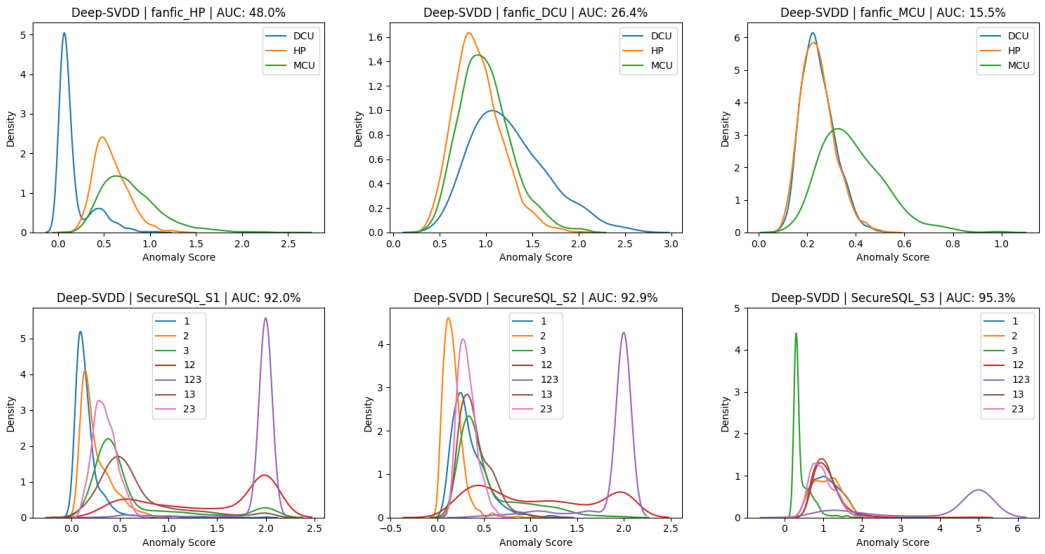

Figure 6: Anomaly Detection using Deep-SVDD to detect when a leak has occured by comparing the outlier and inlier anomaly score. LLM Perplexity significantly outperforms common methods used for Anomaly Detection.

Framework) including our best method (Framework: Logit Composition) with and without using 6NF-like database normalization, which is equivalent to the scenario where a user has credentials to access $T = \{1, 2, 3\}$. We note that neither the baseline generalized model nor the secure compositions have seen the union Silos ($S_{1\cup2}$, $S_{1\cup3}$, $S_{2\cup3}$, and $S_{1\cup2\cup3}$) and that only the exponential baseline model has been trained on those silos.

The performance of the composed fine-tunings on the individual Silos would give an indication as to whether the resulting composition is able to retain the knowledge of each individual fine-tuning from each separate Silo; This performance is expected to be traded off for privacy while the better compositional methods mitigate the extent of this trade off and maintain maximal privacy. The performance on the union Silos indicates whether the composed fine-tunings are able to successfully generalize knowledge from the individual fine-tunings which is an essential component in answering questions that no individual fine-tuning or silo can answer.

F Additional Anomaly Detection Results

In fig. 3 we show the graphical anomaly score-density curves for the first data silo in the two datasets tested, SecureSQL and CrossoverFanFic. The inlier silo is listed in the title of each graph as well as the method used to compute anomaly scores and the reported Area Under the Curve (AUC). These graphs very quickly show the separation of each inlier silo for each method. There is significantly more overlap with both DeepSVDD and TF-IDF, whereas LLM Perplexity shows much strong separation and shorter tails minimizing overlap and maximizing the AUC.

Table 2 summarizes the graphs in fig. 3 by reporting only the AUC. From these tables we can see that LLM Perplexity outperforms popular methods for Anomaly Detection. In addition to be being the highest performing method, LLM Perplexity is also extremely lightweight because it relies on training a LoRA adapter for Llama-2-7b. In contrast, Deep-SVDD is more than a magnitude of time slower to train. However, TF-IDF is the fastest and most lightweight method as it does not require any type of deep learning. Nevertheless, accuracy is superior for LLM Perplexity when finetuned on Llama-2-7b.

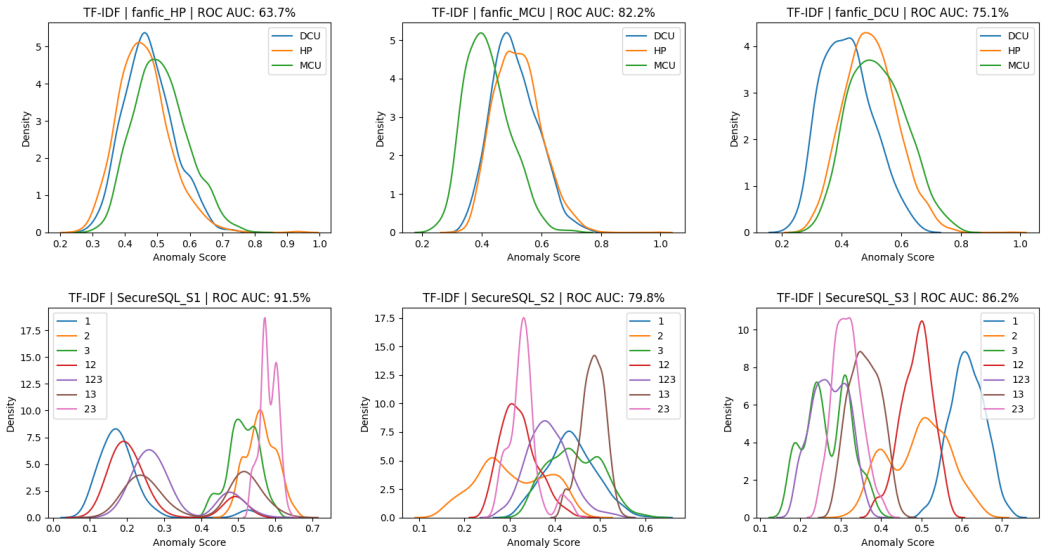

Figure 7: Anomaly Detection using TF-IDF to detect when a leak has occured by comparing the outlier and inlier anomaly score. LLM Perplexity significantly outperforms common methods used for Anomaly Detection.

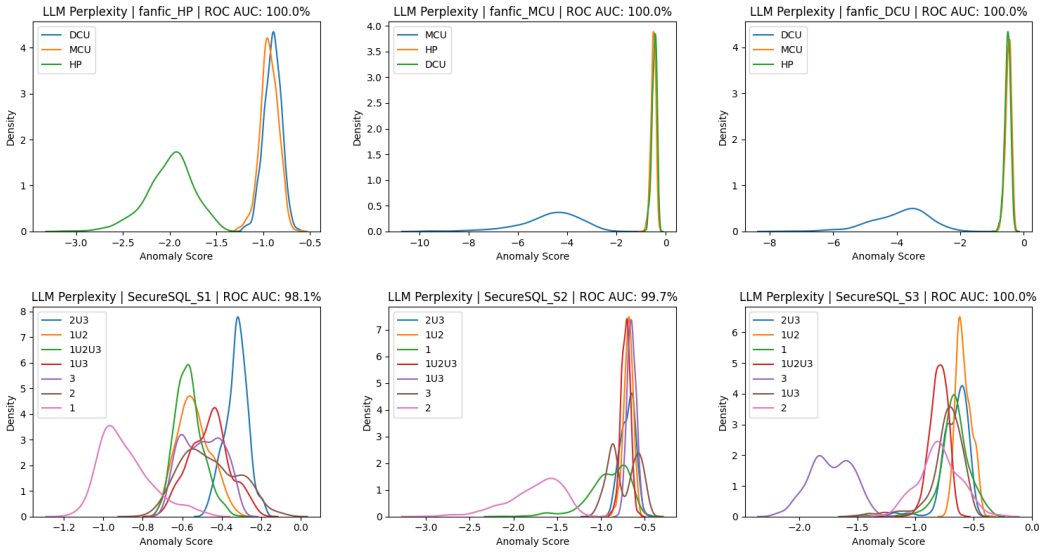

Figure 8: Anomaly Detection using LLM Perplexity to detect when a leak has occured by comparing the outlier and inlier anomaly score. LLM Perplexity significantly outperforms common methods used for Anomaly Detection.

G Additional Anomaly Classification Results

G.1 Anomaly Classification - Secure-NL2SQL

SecureLLM	precision	recall	f1-score	N
$Silos_1$	0.947	0.969	0.958	256
$Silos_2$	0.993	1.000	0.996	273
$Silos_3$	0.993	0.997	0.995	300
$Silos_{1\cup2}$	0.980	0.963	0.971	300
$Silos_{1\cup3}$	0.942	0.913	0.927	300
$Silos_{2\cup3}$	0.957	0.967	0.962	300
$Silos_{1\cup2\cup3}$	0.960	0.967	0.963	300
macro avg	0.967	0.968	0.968	2029
weighted avg	0.967	0.967	0.967	2029

lstm	precision	recall	f1-score	N
$Silos_1$	0.567	0.664	0.612	256
$Silos_2$	0.641	0.582	0.610	273
$Silos_3$	0.664	0.647	0.655	300
$Silos_{1\cup2}$	0.432	0.507	0.466	300
$Silos_{1\cup3}$	0.528	0.413	0.464	300
$Silos_{2\cup3}$	0.599	0.627	0.612	300
$Silos_{1\cup2\cup3}$	0.387	0.370	0.378	300
macro avg	0.477	0.476	0.475	2029
weighted avg	0.544	0.541	0.540	2029

gru	precision	recall	f1-score	N
$Silos_1$	0.549	0.785	0.646	256
$Silos_2$	0.698	0.755	0.725	273
$Silos_3$	0.945	0.917	0.931	300
$Silos_{1\cup2}$	0.491	0.440	0.464	300
$Silos_{1\cup3}$	0.671	0.577	0.620	300
$Silos_{2\cup3}$	0.776	0.703	0.738	300
$Silos_{1\cup2\cup3}$	0.673	0.623	0.647	300
macro avg	0.686	0.686	0.682	2029
weighted avg	0.689	0.683	0.682	2029

cnn	precision	recall	f1-score	N
$Silos_1$	0.850	0.883	0.866	256
$Silos_2$	0.848	0.821	0.834	273
$Silos_3$	0.909	0.937	0.923	300
$Silos_{1\cup2}$	0.851	0.897	0.873	300
$Silos_{1\cup3}$	0.781	0.820	0.800	300
$Silos_{2\cup3}$	0.840	0.770	0.803	300
$Silos_{1\cup2\cup3}$	0.898	0.850	0.873	300
macro avg	0.854	0.854	0.853	2029
weighted avg	0.854	0.854	0.853	2029

bilstm	precision	recall	f1-score	N
$Silos_1$	0.528	0.516	0.522	256
$Silos_2$	0.689	0.495	0.576	273
$Silos_3$	0.857	0.897	0.876	300
$Silos_{1\cup2}$	0.392	0.467	0.426	300
$Silos_{1\cup3}$	0.494	0.297	0.371	300
$Silos_{2\cup3}$	0.545	0.523	0.534	300
$Silos_{1\cup2\cup3}$	0.440	0.623	0.516	300
macro avg	0.493	0.477	0.478	2029
weighted avg	0.563	0.547	0.546	2029

transformer	precision	recall	f1-score	N
$Silos_1$	0.354	0.203	0.258	256
$Silos_2$	0.519	0.410	0.458	273
$Silos_3$	0.605	0.623	0.614	300
$Silos_{1\cup2}$	0.480	0.613	0.539	300
$Silos_{1\cup3}$	0.366	0.300	0.330	300
$Silos_{2\cup3}$	0.461	0.493	0.477	300
$Silos_{1\cup2\cup3}$	0.401	0.527	0.455	300
macro avg	0.398	0.396	0.391	2029
weighted avg	0.456	0.459	0.451	2029

G.2 Anomaly Classification - CrossoverFanFic

SecureLLM	precision	recall	f1-score	N
HP	0.989	1.000	0.994	959
MCU	0.955	1.000	0.977	513
DCU	0.960	1.000	0.979	309
HP-MCU	0.802	0.853	0.827	3644
HP-DCU	0.610	0.122	0.203	864
MCU-DCU	0.717	0.583	0.643	1479
HP-MCU-DCU	0.029	0.293	0.053	75
macro avg	0.723	0.693	0.668	7843
weighted avg	0.797	0.750	0.752	7843

lstm	precision	recall	f1-score	N
HP	0.155	0.078	0.104	959
MCU	0.127	0.103	0.114	513
DCU	0.176	0.019	0.035	309
HP-MCU	0.488	0.733	0.586	3644
HP-DCU	0.122	0.031	0.050	864
MCU-DCU	0.331	0.120	0.176	1479
HP-MCU-DCU	0.018	0.160	0.032	75
macro avg	0.177	0.156	0.137	7843
weighted avg	0.337	0.385	0.333	7843

gru	precision	recall	f1-score	N
HP	0.219	0.087	0.124	959
MCU	0.099	0.058	0.074	513
DCU	0.200	0.013	0.024	309
HP-MCU	0.469	0.728	0.571	3644
HP-DCU	0.142	0.020	0.035	864
MCU-DCU	0.260	0.072	0.112	1479
HP-MCU-DCU	0.026	0.333	0.049	75
macro avg	0.177	0.164	0.124	7843
weighted avg	0.324	0.372	0.312	7843

cnn	precision	recall	f1-score	N
HP	0.268	0.162	0.202	959
MCU	0.313	0.374	0.341	513
DCU	0.295	0.058	0.097	309
HP-MCU	0.507	0.703	0.589	3644
HP-DCU	0.205	0.044	0.072	864
MCU-DCU	0.326	0.089	0.140	1479
HP-MCU-DCU	0.027	0.320	0.050	75
macro avg	0.243	0.219	0.186	7843
weighted avg	0.385	0.398	0.359	7843

bilstm	precision	recall	f1-score	N
HP	0.208	0.125	0.156	959
MCU	0.149	0.144	0.147	513
DCU	0.098	0.026	0.041	309
HP-MCU	0.502	0.696	0.583	3644
HP-DCU	0.152	0.049	0.074	864
MCU-DCU	0.319	0.143	0.198	1479
HP-MCU-DCU	0.010	0.093	0.018	75
macro avg	0.180	0.160	0.152	7843
weighted avg	0.349	0.382	0.347	7843

transformer	precision	recall	f1-score	N
HP	0.195	0.130	0.156	959
MCU	0.102	0.088	0.094	513
DCU	0.246	0.113	0.155	309
HP-MCU	0.515	0.699	0.593	3644
HP-DCU	0.140	0.047	0.071	864
MCU-DCU	0.291	0.150	0.198	1479
HP-MCU-DCU	0.012	0.093	0.021	75
macro avg	0.188	0.165	0.161	7843
weighted avg	0.350	0.385	0.352	7843

H Anomaly Classification Sliding Window Ablation

W	LSTM	GRU	CNN	Bi.	Tran.	Ours
128	0.33	0.31	0.36	0.35	0.35	0.75
96	0.33	0.31	0.35	0.34	0.34	0.75
64	0.35	0.32	0.34	0.36	0.33	0.74
32	0.36	0.33	0.34	0.37	0.33	0.72
16	0.40	0.38	NaN	0.41	0.37	0.67
8	0.42	0.41	NaN	0.42	0.41	0.59
4	0.48	0.48	NaN	0.48	0.47	0.55

Table 7: W is the window size tested for each method (LSTM, GRU, 1d-CNN, BiLSTM, Transformer, and LLM Perplexity. We report the same F1-scores shown in tables 3 and 4. When the window is 16 or smaller, the 1d-CNN does not produce a result.

Within our Anomaly Detection (AD) experiment, we also tested the optimal sliding window for CrossoverFanFic dataset. A window size of 128 was used at train time for all methods, with the window size only varying at test time. Table 7 shows how a window size of 128, 96, 64, 32, 16, 8, and 4 degrades the AD corresponding AD as less and less context is given to each. It is reasonable to assume that larger window sizes would yield more separation across each method, but to a point of diminishing returns. Yet, the window size of 128 used for our AD experiment is still much smaller than modern context windows for practically every LLM release since 2019 with T5 being the smallest at 512 tokens [Raffel et al., 2020].