NoEsis: A Modular LLM with Differentially Private Knowledge Transfer

Rob Romijnders*
University of Amsterdam

Stefanos Laskaridis, Ali Shahin Shamsabadi, Hamed Haddadi
Brave Research

Abstract

Large Language Models (LLMs) are typically trained on vast amounts of data, springing from various sources. Even when designed modularly, such as Mixture-of-Experts models, LLMs can leak private information on their source training data. Conversely, training such large models in isolation hinders generalization and does not allow for the sharing of knowledge. Therefore, we propose a framework, NOESIS, which builds upon the desired properties of *modularity*, *privacy*, and *knowledge transfer*. NOESIS integrates differential privacy with a hybrid architecture that combines domain-specific adapters, acting as experts, and common prompt tokens, acting as a knowledge-sharing backbone. The results from our evaluation on CODEXGLUE show that NOESIS can achieve provable privacy guarantees with knowledge transfer across domains, and empirically show protection against Membership Inference Attacks. On code completion tasks, NOESIS bridges 84% of the accuracy gap between a non-shared and a non-private baseline. This work addresses the growing need for privacy-preserving AI systems that can operate across institutional boundaries while respecting data sovereignty and preventing unauthorized knowledge extraction.

1 Introduction

Large Language Models have brought significant disruption to the field of Artificial Intelligence and have thoroughly transformed various use cases, from intelligent assistants [1] and code co-pilots [2] to agentic web browsing [3] and enhanced tutoring [4]. Moreover, they demonstrate great scaling potential, consuming petabytes of raw textual or multimodal data [5] without their performance plateauing. As this trend continues, all public data resources will eventually be consumed [6]. Therefore, tapping into private data silos will become the next significant source of training data [7, 8].

However, copyright [9] and privacy laws [10, 11] may prevent large models from being trained and served as-is, uniformly across the globe. This introduces the need to orchestrate model training that is somehow separated per region or source. Maintaining separate models for each data source, however, quickly becomes intractable and burdensome. At the same time, private organizations can own data they want to use for their custom LLM but not contribute (or expose) publicly [12, 13]. For instance, client institutions may wish to train domain-specific Copilots [14] while benefiting from other repositories, without leaking proprietary information [15] to the public domain.

*work done while interning at Brave Research.

Article presented at the Second IASEAI Conference (IASEAI'26).

Table 1: Learning approaches and deployment characteristics of each solution. NOESIS is the only method that is *modular*, *private* and benefits from *knowledge transfer* among domains.

Modular	Private	Transfer	Approach	Remarks
✗	✗	✓	Monolithic model [16]	Leaking privacy in model parameters
✗	✓	✓	Private learning ([16]+[17])	Low performance
✓	✓	✗	MoE (share-nothing) [18]	No benefit from knowledge transfer
✓	✗	✓	MoE (shared-expert) [19]	Privacy leakage in shared expert
✓	✓	✓	NOESIS (ours)	Addresses all three problems

To approach this problem, we draw from Modular Learning [20] for routing knowledge across parts of a neural network and adaptively serve to different domains. While off-the-shelf Mixture-of-Experts (MoE) models [21] adopt an architecture where different domains can share common parameters thus enabling knowledge transfer, they can introduce privacy risks [22] – exactly because of this sharing. In addition, training an entire MoE model under Differential Privacy (DP) significantly reduces its utility as training a large shared backbone network over multiple domains requires adding large amounts of DP noise. Therefore, there is an inherent tension between *modularity*, *privacy*, and *knowledge sharing*.

We propose a privacy-friendly architecture for differentially private modular learning, named Nexus-of-Experts (NoE). This inherits a domain-routed mixture of low-rank adapters model (Mix-LoRA), applied to the fully connected layer of each transformer block, further enhanced by shared tokens obtained through a DP training algorithm for knowledge transfer (Fig. 1). To the best of our knowledge, we are the first to explore the interplay between modularity, privacy, and knowledge transfer in the realm of LLMs. Compared to conventional MoE methods [23], we enable document-private learning without sacrificing efficiency. Compared to adaptive parameter-efficient fine-tuning (PEFT) alternatives [24], we facilitate knowledge transfer between domains while providing modularity at deployment time. The AI safety aspect of this work is to study data exposure and unauthorized knowledge extraction in the multi-institutional setting. In addition to proposing the new training method, we run a Membership Inference Attack to demonstrate the current safety vulnerability when training multi-institutional AI.

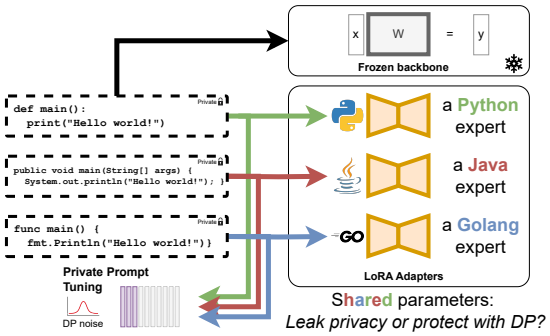


Figure 1: **Problem Setting:** A pretraining dataset and multiple private datasets, local to each domain, are available. Privacy is required in each domain. Domain-specific parameters can be included for improvement, but not shared with other domains, i.e., modularity. Simultaneously, we want private knowledge transfer: under differential privacy, learnings can be shared. NOESIS tackles this setting by adopting a hybrid PEFT paradigm to achieve the properties of **modularity**, **privacy**, and **knowledge transfer**.

Statement of Contributions: In summary, the contributions of this work are the following:

- We demonstrate that existing routing-based models leak private information, especially LLMs. This exposes the need for privacy-preserving training in such models.
- We propose NOESIS, a domain-routed, hybrid PEFT solution that utilizes a shared DP-trained set of prompt tokens as a knowledge-sharing backbone.
- We analyze the balance between privacy and model expressiveness by varying the number of trainable parameters and the corresponding noise required to ensure privacy. Our hybrid method performs up to 1.4% accuracy points better than the non-hybrid, LoRA-only variant.
- We apply our technique to the problem of code generation, adopting different programming languages as our domains and ensuring privacy at the document level. Our results show that we can achieve from 1.4 to 4.9% points better downstream accuracy, averaged across domains, while protecting our model against privacy leakage, both provably (*with DP guarantees*) and empirically (*against a commonly used Membership Inference Attack*).

2 Design choices

This paper studies the problem of fine-tuning a publicly pre-trained LLM for a downstream task (e.g., code completion) with training data that spans various private domains (e.g., programming languages) and can be modularly served in each such domain. This section explains the three properties of this problem and shows how NOESIS uniquely satisfies all of them (Table 1). As NOESIS introduces a new paradigm to LLM training, it is important to understand the desiderata of this problem and how they map onto the architectural choices and training process.

Knowledge Transfer. When training on multiple data sources, knowledge transfer refers to the improvement in results achieved by learning a joint model from all data sources, rather than learning a separate model for each data source [25]. It manifests in various settings, including continual learning [26], federated learning [27], and model pre-training [28, 29]. Whatever the setup, the goal is that the output model achieves higher accuracy on any dataset than a separate model trained for that particular dataset in isolation. The opposite behavior can also manifest, i.e. *catastrophic forgetting* [30, 31], where domain knowledge clashes and the network "forgets" backpropagated knowledge. In our multi-domain setting, a design goal is to make more accurate predictions in each domain, with the help of the potentially vast amounts of data in another domain. This must happen while maintaining privacy with respect to potential adversaries from other domains (see Sec. 4.5)[2]. This trade-off is particularly challenging in multi-domain settings where different domains may have varying data sizes and data heterogeneity.

Modular Learning. Modular learning refers to models whose execution subgraphs can be specialized in terms of function [20]. One such example is MoE models [21, 23], which are routed over their input samples. For our purpose, the benefits of modularity are three-fold: i) *computational* and ii) *privacy-related*, while also iii) enabling the potential of *knowledge transfer* explained above. Computationally, modularity enables us to train once and serve model components everywhere, eliminating the need to maintain multiple copies and replicate knowledge. Additionally, other computational benefits are lower memory cost, lower cost of context switching when serving multiple domains, and lower communication cost of the model and/or its updates in a federated setting. At the same time, separating domain-specific and shared parameters enables a clear separation of concerns by domain. This is particularly important for serving under different jurisdictions and providing formal guarantees regarding the flow of information and the privacy of source data. Modularity also makes the knowledge sharing between domains explicit, as dedicated experts for each domain are trained to specialize in that domain, and other dedicated parameters are used for the shared knowledge. This architectural separation is crucial for privacy preservation, as it allows us to apply different privacy guarantees to shared versus domain-specific components.

Preserving privacy. Machine learning models can memorize and unintentionally leak information about their training data [12]. This capability introduces a significant privacy concern when the training data contains private information. Our goal is to improve model accuracy by training on multiple private datasets from different domains, without compromising private data is one domain to users/adversaries in the other domain. DP training algorithms provide a guarantee by bounding the leakage of each individual data point that may occur during training [32]. Therefore, we formalize the threat model when training a model on private datasets from multiple domains, where an adversary in one domain should not be able to discern information about training data in another domain, and we design a DP training algorithm for this setting[2].

These requirements are essential in various real-world scenarios. An example is a globally operating organization that wants to train a single model but wants to serve the model modularly across different regions to ensure that the model does not leak information between regions due to different copyright or privacy laws [33, 34]. Another example can be an institution that wants to train a model on private data that resides in silos but needs to ensure that the model does not leak information about the private data to the public [13]. Finally, the modular aspect can be helpful in scenarios where an organization wants to train a model on multiple domains, such as legal, technical, and administrative documents [35], but wants to ensure that the model does not leak information between such domains.

[2]It is evident that utility in the form of positive knowledge transfer and privacy can be opposing factors, i.e., the best transfer is non-private, and the best privacy protection is not to share. Our method finds a balance.

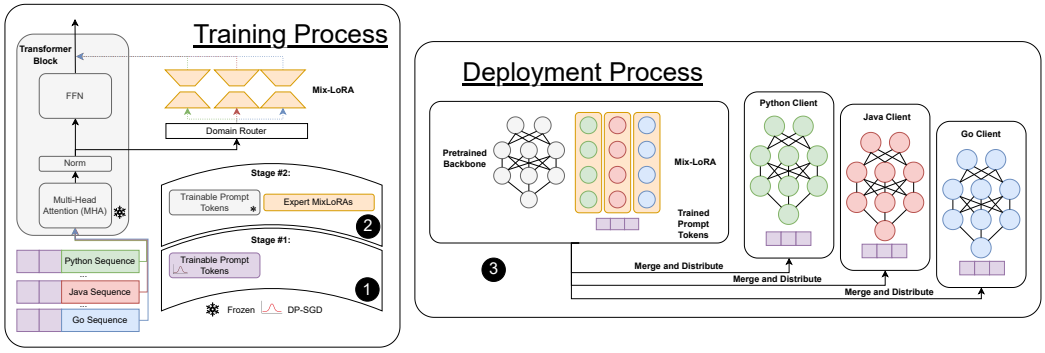


Figure 2: Training and deployment process of NOESIS. The training comprises a two-stage process: Stage 1: Training of private prompt token parameters across domains; Stage 2: Training of expert Mix-LoRA per domain. Deployment involves merging the LoRA parameters with the backbone and sharing the trained, prompt tokens with the respective downstream client.

3 NOESIS

3.1 Architecture & Training Process

Aiming for the properties of *modularity*, *knowledge transfer*, and *privacy*, we introduce NOESIS, a hybrid, DP-trained, domain-routed Nexus-of-Experts (NoE) solution that adopts a shared set of trainable tokens and Mix-LoRA experts and enables knowledge transfer between domains. Conceptually:

$$\boxed{\text{NoE} = \text{DP-SGD}(\text{Prompt-Tuning}(X_1, \ldots, X_K)) + \text{SGD}(\text{Mix}\{\text{LoRA}(X_1), \ldots, \text{LoRA}(X_K)\})} \tag{1}$$

Domain-routed models. NOESIS is fundamentally a modular architecture that enables the specialization of network components to different domains. To achieve this, we adopt a domain-level mixture of experts model. Given dataset $D^{(k)}$ from domain k, where $k \in [1, K]$, we route the data from each domain to the respective expert, deterministically. Instead of operating directly on the backbone model parameters, we use a Mix-LoRA [36, 24] architecture that keeps the backbone network frozen and only tunes the adapters per domain, applied on top of feed-forward network (FFN) layers of the transformer blocks. The rationale behind this decision is to enable the "logic" of the network to be tuned per domain while keeping the attention intact [24], which has been shown to be effective in routing-based neural networks [24, 37, 38, 39]. Formally for Mix-LoRA, let $W \in \mathbb{R}^{p \times q}$ be the weight matrix of a linear layer [3] in the FFN of a transformer with parameters θ. We decompose this matrix:

$$W = W + \alpha \sum_{k=1}^{K} \underline{B^{(k)} A^{(k)}}, \tag{2}$$

where $A^{(k)} \in \mathbb{R}^{r \times q}$ and $B^{(k)} \in \mathbb{R}^{p \times r}$ are the trainable (signified by the underline) low-rank matrices for the k-th domain-specific adapter, out of K domains. The experimental section will refer to the rank r as the *adapter rank*. $\alpha \in \mathbb{R}$ is a scalar constant to reduce the variance impact of the adapter.

Enabling Knowledge Transfer. Knowledge transfer has been shown to be valuable in routing-based neural networks [40, 41]. By freezing the backbone network and specializing an expert LoRA per domain, there are no trainable shared parameters to enable knowledge transfer [28, 42, 43]. This means that data and the learning in each domain would not benefit from each other in terms of the predictive task, which may be especially disadvantageous for scarce domains [44].

To facilitate knowledge transfer, we employ a differentially private prompt-tuning mechanism that enables knowledge transfer between domains while minimizing the number of shared trainable parameters and ensuring privacy. Prompt tuning has emerged as a promising method for efficiently fine-tuning large language models (LLMs) by introducing differentiable prompts, which are real-valued tokens that can be learned through back-propagation [45, 46]. Prior work has demonstrated

[3] In reality, there is a bottleneck structure in W of $W_i^{d_{\text{model}} \times d_{\text{ff}}}$ and $W_o^{d_{\text{ff}} \times d_{\text{model}}}$, where d_{model} is the model hidden size ($d_{\text{model}} = 768$, $d_{\text{ff}} = 2048$ for CodeT5+) and we have one LoRA for each.

Table 2: An overview of datasets and domains used for the training and evaluation of NOESIS. We chose code-completion as next-token prediction is a well-defined task with clear domains.

Domain	Source	Description	Training Set	Evaluation Set
Pre-training	GitHub Code [50]	Upstream data	3.7 million	
Python	PY150 [51]	Published dataset†	100,000	50,000
Java	Java GitHub Corpus [52]	Published dataset†	12,934	8,268
Go	CodeSearchNet [53]	Extracted from CodeSearchNet †	2,000	1,559

† Dataset explicitly deduplicated in pre-training dataset [16].

that DP prompt-tuning can achieve substantial predictive accuracy even with stringent privacy budgets, such as $\varepsilon < 1$ [47]. By freezing the LLM backbone parameters and training only the differentiable prompts, we can reduce the number of trainable parameters, thereby minimizing the amount of noise required for DP.

Specifically, given input sample $X = \{x_0^{(k)}, x_1^{(k)}, \ldots, x_\tau^{(k)}\}$ of τ tokens, for a sample of domain k, a common set of trainable prompt embeddings P is prepended to the input sequence, $X' = [P; X]$. Differentially Private Stochastic Gradient Descent (DP-SGD) [47] tunes prompts under the framework of DP by clipping the per-sample gradient to a fixed norm and adding calibrated Gaussian noise to these clipped gradients before applying them to update prompts. Concretely, this means $P^{(t+1)} \leftarrow P^{(t)} - \eta\tilde{g}_P^{(t)}$, where $\tilde{g}_P^{(t)}$ is the noisy clipped gradient with respect to the input at step t and η is the step size.

Using this hybrid of private prompt-tuning and routable low-rank adapters (NoE), we can efficiently adapt the model to different domains while preserving the benefits of shared knowledge. At the same time, the domain-specific adapters enable the model to specialize in the unique aspects of each domain, resulting in improved performance on domain-specific tasks. This hybrid PEFT approach enables lightweight fine-tuning and, therefore, permits efficient noise budgeting[4].

Training Process. The training steps of NOESIS are outlined in Algorithm 1 and visually illustrated in Figure 2 (left). It follows a two-stage procedure, where private learning is performed for the *shared parameters* (Step ❶) and then non-private learning is conducted for the *domain-specific adapters* (Step ❷). The private parameters are collectively referred to as P. The parameters for the expert LoRA are collectively referred to as E and comprise all B and A matrices in Equation 2. The algorithm starts by computing the noise multiplier σ using the privacy accounting [48]. The dataset is then augmented with domain labels, which are used for deterministic routing, and the training loop is started. In each iteration, a batch of documents is randomly drawn from all domains, and the gradients are computed for the domain-specific and shared parameters. In the case of private learning, the gradients are clipped and noised according to DP-SGD [17]. In the second stage, SGD refers to Stochastic Gradient Descent [49].

3.2 Deployment & Privacy Guarantees

Deployment Process. The deployment of NOESIS (Step ❸) addresses the following use case: K departments hold data for their own respective domain. For example, one department holds legal data, another department holds technical documentation, and a third department has data on the supply chain. All data relates to the same company and learning could thus benefit from positive transfer. A trusted server receives all data to train the model with cross-domain insights, such as knowledge transfer. The trusted server returns to each department the domain-specific model parts (LoRA) and the shared parameters, which are then used for inference in each separate domain. The privacy of a document in each domain is maintained, and the model can simultaneously benefit from the shared knowledge. This process is illustrated in Fig. 2 (right).

Document-level DP. We recall the definition from Dwork and Roth [32]. A randomized algorithm $f(\cdot)$ is (ε, δ)-differentially private if the following holds for any two adjacent datasets D, D', and for

[4]We explore the sensitivity of NOESIS to the number of trainable parameters in Sec. 4.3, along with the alternative using LoRA as the knowledge sharing architecture.

Algorithm 1 Stepwise Training of NOESIS

Input: Pre-trained model θ_0, training sets $\{D_k\}_{k=1}^{K}$ for K domains, collectively named D with N_k documents in domain k and total number of documents N, batch size B, learning rate η, number of iterations T_1, T_2, privacy budget ε, δ, gradient clipping bound C
Output: Modular LLM with privacy guarantees

$\sigma \leftarrow$ PRIVACCOUNTING$(\varepsilon, \delta, B, N, T_1)$ ***// Stage 0: Compute noise multiplier***

$P_0 \leftarrow$ Randomly initialize prompt tokens ***// Stage 1: DP training of shared parameters***
for $t = 1$ **to** T_1 **do**
  Randomly draw batch $\mathcal{B}_t$ of size B from D
  $P_t \leftarrow P_{t-1} -$ DP-SGD$(\theta_0, P_{t-1}, \mathcal{B}_t, \sigma, C)$
end for

$E_0 \leftarrow$ Randomly initialize expert adapters ***// Stage 2: Training of domain-specific parameters***
for $t = 1$ **to** T_2 **do**
  Randomly draw batch $\mathcal{B}_t$ of size B from D
  $E_t \leftarrow E_{t-1} -$ SGD$(\theta_0, P_T, E_{t-1}, \mathcal{B}_t)$
end for

any subset $\mathcal{S}$ of outputs:

$$\Pr[f(D) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[f(D') \in \mathcal{S}] + \delta, \tag{3}$$

where $D = \{d_i\}_{i=1}^{N}$ is a dataset that contains N documents. Each d_i is a string of tokens of arbitrary length. In our multi-domain setting, a dataset is all the documents in all domains. Two datasets D and D' are adjacent if they are identical except for one document in any domain being removed. For this privacy protection, we can calculate the noise multiplier as a function of total dataset size N and batch size B in Algorithm 1. In keeping with prior work, we run all privacy-focused experiments with $\varepsilon = 1.0$ and $\delta = 10^{-6}$, which is smaller than the rule-of-thumb of one divided by dataset size [54, 55].

4 Evaluation

Our evaluation involves the task of code completion across different programming languages, which we consider private domains. We build a set of assessments that illustrate the behavior of NOESIS. The first set of experiments (Sec. 4.2) is designed to investigate the effectiveness of knowledge transfer across domains, while preserving privacy over DP. This is compared to the status quo techniques from PEFT and regular fine-tuning. Subsequently, we perform a sensitivity analysis (Sec. 4.3) and ablation study (Sec. 4.4), which investigate the effect of different components to the behavior of NOESIS. Lastly, we demonstrate that baseline MoE models are vulnerable to a Membership Inference Attack and show that NOESIS mitigates this vulnerability (Sec. 4.5), thereby illustrating the practical implications of our approach's gains.

4.1 Experimental setup

Model. We use a decoder-only model architecture for our experiments, inspired by the GPT family of models [28]. Specifically, we use the decoder part of the pre-trained CodeT5+ (220M) model. Given the previous tokens as input, the model was pretrained to autoregressively predict the next token in a sequence by using the softmax as log-likelihood, and cross-entropy as the loss function. This setup is particularly suitable for machine learning tasks and evaluating our paradigm in programming languages, where the goal is to auto-complete code one token at a time.

Datasets. We choose the Python, Java, and Go programming languages for our multi-domain training. We chose these domains as they are varied in nature (statically and dynamically typed, weakly and strongly typed), and, on a practical level, as the CodeT5+ model was trained on data specifically deduplicated for datasets in these languages [16]. Such deduplication in the pretraining data is crucial for simulating the setting of having private domains for fine-tuning. The dataset for Python is the PY150 dataset [51], and the dataset for Java is the JavaCorpus [52], which follows a setting similar to Gong et al. [19]. The Go data comes from the CodeSearchNet dataset [53]. Table 2 notes the respective sizes of these datasets. We obtain the data from the HuggingFace Hub, which is

Table 3: The main experimental comparison of different models on modularity, privacy, and knowledge transfer. NOESIS uniquely addresses all three aspects, achieving high accuracy while obtaining DP at $\varepsilon = 1.0$ and enabling knowledge transfer across domains. (Knowledge transfer is the increase in accuracy compared to "Share Nothing," which does not have shared parameters between domains.)

	Model	ε	Modular	Private	Transfer	PEFT	Python	Java	Go
(i)	Share Nothing	0.0	✓	✓	✗	✓	68.31	60.19	64.17
(ii)	Solo (separate models)	1.0	✗	✓	✗	✗	30.19	14.84	4.84
(iii)	Monolithic Fine-tuning [17]	1.0	✗	✓	✓	✗	36.05	23.54	18.34
(iv)	Single Common Adapter (rc=512)* [57]	1.0	✗	✓	✓	✓	48.21	36.16	27.24
(v)	Prompt-Tuning Only (pt32)† [47]	1.0	✗	✓	✓	✓	35.03	24.29	9.67
(vi)	NOESIS (pt32)†	1.0	✓	✓	✓	✓	**69.14**	**61.18**	**66.53**

*rc, rank r of the common LoRA; †*pt: number of trainable prompt tokens*

provisioned by the CODEXGLUE effort [56]. More details about the datasets can be found in the Appendix.

Training Setup & Hyperparameters. Our method is built with PyTorch (v2.4.1) and HuggingFace transformers (v4.45.0), with the functionality for DP handled by Opacus (v1.5.2). We train our models on 4×4090 GPUs. During training, we use a context length of 512 tokens and a learning rate of 10^{-3}. For training under the DP guarantee, we apply gradient clipping with a norm of 1.0 and add Gaussian noise with a standard deviation that is scaled according to the targeted privacy budget. We utilize the privacy accountant of the Opacus library to calculate the noise multiplier constant [48]. A private document within the dataset can be of any length. Therefore, an epoch is defined as sampling one 512-token block for each document in the dataset. This ensures the optimal use of compute resources and maintains the privacy guarantee, as each 512-token sequence is a subslice. Details about our hyperparameter values can be found in the Appendix. The code and datasets are published anonymously and will be open-sourced upon acceptance.

Baselines. We compare NOESIS against various trainable baselines, namely: *i) Share Nothing*: a modular model without shared parameters, which serves to simulate the effect of not having knowledge transfer; *ii) Solo*: a separate model fine-tuned separately for each domain; *iii) Monolithic Fine-tuning*: DP fine-tuning the whole model without any routable components [17], which is to compare with Solo and confirm the effect of knowledge transfer in monolithic models [41]; *iv) Single Common Adapter*: A common LoRA trained with DP across domains [57], which is to compare the influence of using PEFT instead of monolithic fine-tuning; *v) Prompt-Tuning Only*: tuning only the prompts under the DP guarantee, which serves to investigate the impact of knowledge transfer without any modularity in the form of Mix-LoRA [47]. Details about the baselines can be found in the Appendix.

4.2 Experimental Results

The experimental results are presented in two parts. The modularity, privacy, and knowledge transfer properties are treated first. Second, we analyze the privacy impact and run ablation experiments.

The results of comparing NOESIS against baselines are shown in Table 3. We compare our model against the Share Nothing model, which has no shared parameters and thus no knowledge transfer. We see that NOESIS achieves an improvement of 0.83% points on Python, 1.0% points on Java, and 2.41% points on Go. This suggests that knowledge transfer is beneficial for next-token prediction, particularly in domains with limited data, such as Go, which has 50 times less data than Python. Qualitative examples of NOESIS versus Share Nothing across all three domains can be found in the Appendix.

Comparing NOESIS to Prompt-Tuning Only, we observe an improvement of more than 30% points across the three domains. This shows the benefit of using modularity as a privacy-friendly model architecture. Secondly, all models score higher accuracy than the Solo models and the Monolithic Fine-tuning model, indicating the importance of modularity and privacy in multi-domain training. Comparing, moreover, the experiments of the Solo models against the Monolithic training, we observe the effect of positive knowledge transfer in the non-modular setting: the latter model achieves higher accuracy than each of the Solo models. Finally, using the Single Common Adapter setting achieves about nine accuracy points improvement compared to monolithic fine-tuning, which corroborates earlier studies on parameter-efficient fine-tuning and DP [58, 57].

Figure 3: Sensitivity analysis to the number of shared parameters. We find that prompt-tuning achieves the best trade-off between utility and parameter count, achieving the highest accuracy across domains. 'rc' is the rank of the common LoRA, 'pt' is the number of prompt tokens.

Model	# Shared Params.	Python	Java	Go
(rc64)	$7680 \times 768 = 5.9$M	68.73	60.32	65.32
(rc4)	$480 \times 768 = 369$k	68.77	60.45	65.39
(rc1)	$120 \times 768 = 92$k	68.74	60.43	65.26
(pt120)	$120 \times 768 = 92$k	69.13	61.09	66.49
(pt32)	$32 \times 768 = 25$k	**69.14**	**61.18**	**66.53**
(pt8)	$8 \times 768 = 6$k	69.13	61.16	66.49

Figure 4: Ablation study. Removing the shared, common prompts impacts the data-heavy domain most – Python. Removing the domain experts impacts the data-scarce domain most – Go.

	NoEsis (pt32)	w/out common prompts	w/out domain experts
	ε **= 1.0**	ε **= 0.0**	ε **= 1.0**
Python	69.14	4.65	33.91
Java	61.19	14.31	24.30
Go	66.58	15.42	19.13

4.3 Sensitivity of Shared Parameters

Investigating the effect of knowledge transfer, we ask *"how many shared parameters are optimal for knowledge transfer?"* This optimal number of parameters is contended by three directions: there can be too many shared parameters, vulnerable to *overfitting*; there can be too few shared parameters, leading to *underfitting*; thirdly, more shared parameters require more DP noise [17], which negatively affects the signal-to-noise ratio during learning [59]. For this experiment, we train with $4\times$ more and $4\times$ less trainable prompt tokens to highlight this trade-off. We also evaluate a variant with a LoRA as a common knowledge-sharing backbone (explained in the Appendix) over variable ranks, but find that such a design yields lower accuracy. Table 3 shows the analysis of parameter size. The model with 32 shared prompt tokens (pt32) achieves the highest accuracy across all domains. This indicates that a moderate number of shared parameters is optimal for knowledge transfer, as using either 8 or 120 tokens has worse results. Using a LoRA as a common parameter generally has lower accuracy than any prompt-tuning method. We hypothesize that this is due to the LoRA requiring more parameters. For example, LoRA with rank 1 and 120 prompt tokens has the same number of parameters, but the prompt-tuning method has overall better accuracy. *This indicates that prompt-tuning is a parameter-efficient option for providing knowledge transfer under differential privacy.* The same, however, cannot be stated for the domain experts (Table 3(v)).

4.4 Ablation Study

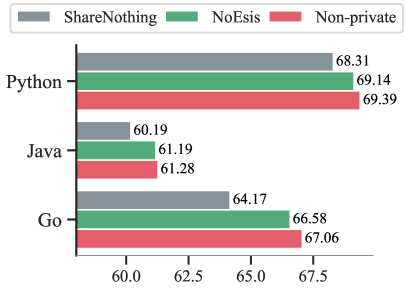
We run an ablation study to understand the impact of each component of NoEsis on the final performance. We surgically remove parts of the training method to see the performance impact. The results are shown in Table 4. Removing any of the components of NoEsis is detrimental to its downstream accuracy, with a minimum drop of 36.89% points and an average of 50% for common prompts and 30% for domain experts. This can be seen from the difference in the columns of Table 4). The common prompt seems to be more important for accuracy, but this may be an effect of our training stage ordering. However, it yields an important conclusion that LoRA experts do not learn or "undo" the knowledge of prompt tokens, but rather incorporate it into their existing knowledge. Moreover, the common prompts are not a nuisance and do encode important bits of information across domains. Finally, and in line with the comparison with the Share-Nothing (*(i)*), it appears that knowledge transfer (whether through the removal of common prompts or fine-tuning without them) has a more severe impact on smaller datasets, particularly Go, which has $50\times$ fewer documents than Python. Arguably, scarce datasets tend to benefit more from knowledge sharing.

Next, we quantify the "cost of privacy," by comparing NoEsis in the same setup, but training without the DP guarantee. This yields an upper bound in performance. NoEsis reaches 99.6% of that optimal accuracy for Python, 99.8% for Java and 99.3% for Go. As shown in Fig. 6, between the Share Nothing baseline (non-shared) and the non-private variant, NoEsis is able to achieve 84% of the way on average, showing the benefit of knowledge transfer. The results from the last line of Fig. 6 are particularly noteworthy, and further corroborate that scarce languages (i.e. Go) benefit most from the knowledge transfer of NoEsis, boosted by more than 2.4% points over the Shared Nothing setup. *This highlights the effectiveness of our approach in leveraging shared knowledge across domains to enhance token completion tasks, especially for a lower-resource domain.*

Figure 5: Mix-LoRA models have a privacy vulnerability in the shared parameters that are shared between domains; we are the first to expose this privacy threat. NoEsis reduces the vulnerability while maintaining good predictive accuracy (Table 3). The results are shown as 'non-private result' → 'private result'.

Attack	AUROC (%)		TPR@1 (%)	
Python (from Java)	51.8 →	50.6	1.1 →	1.0
Python (from Go)	51.7 →	50.7	1.2 →	1.1
Java (from Go)	65.4 →	55.7	2.6 →	1.4
Java (from Python)	64.8 →	55.0	2.3 →	1.3
Go (from Python)	53.2 →	50.6	1.1 →	0.7
Go (from Java)	54.0 →	50.1	1.3 →	0.6

Figure 6: Between the results of a non-shared model, which is the baseline, and a non-private model, which obtains the highest accuracy, NoEsis bridges the gap in accuracy by about 84+%.



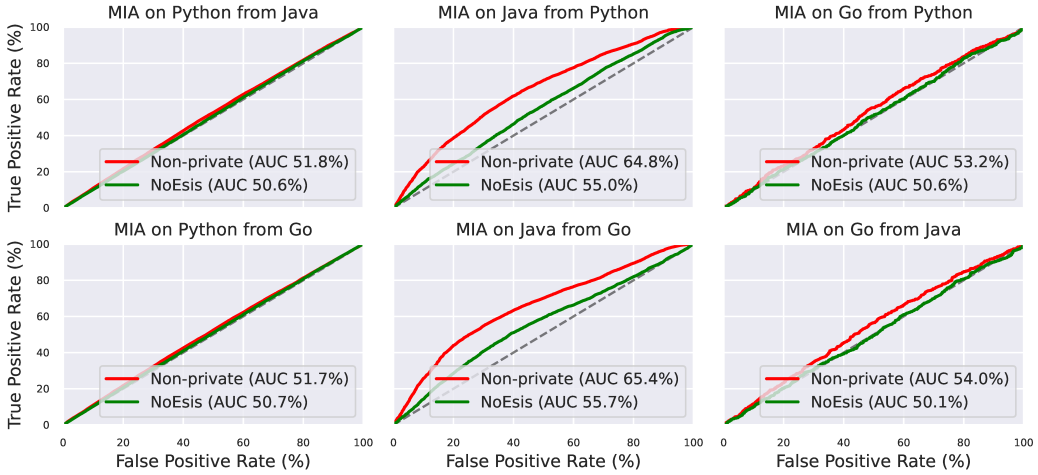


Figure 7: The cross-domain Membership Inference Attack. The high TPR signifies a privacy leak in modular models like Mix-LoRA. Table 5 also reports the TPR at a 1% False Positive Rate (FPR). In terms of both AUROC and TPR, NoEsis (green) provides better privacy protection, compared to the non-private baseline (red), while keeping the decrease in accuracy limited, as shown in Figure 6.

4.5 Empirical Assessment of Privacy

To complement the analytical privacy guarantees, we measure empirical privacy based on a Membership Inference Attack (MIA) [60]. This attack aims to determine whether an input sequence, s, is a member of the model's training data. Several MIAs on LLMs have been proposed [61, 12]. We leverage an MIA based on the predictive log-likelihood scores and extend the attack to a multi-domain setting, named *cross-domain MIA*. The attack is particularly relevant in the context of routing-based models [62, 19, 63, 18, 64, 38], where shared parameters can leak information between domains. To the best of our knowledge, we are the first to uncover this cross-domain attack for modular deep learning models like MoE.

The cross-domain attack follows the threat model where an attacker in a particular domain can make arbitrary predictions with that domain's parameters, and the attacker's goal is to infer the membership of a training sample in another domain. The scoring function is $S(\mathbf{s}) = \mathcal{L}(\mathbf{s}; \mathcal{M})$, the average log-likelihood assigned to each successive token of the input text. The attacker compares this score to a threshold τ to guess the membership of the input text: $\mathbf{s}$ is inferred as a member if $S(\mathbf{s}) > \tau$.

The success of the attack is measured by two key metrics: the Area Under the ROC Curve (AUC) and the True Positive Rate (TPR). True and false positives are further defined in the Appendix. Figure 7

shows the results of the MIA, which demonstrates that the original Mix-LoRA models memorize and leak private information and that NOESIS reduces this threat. Attacking Java, for example, from the Go domain, NOESIS has less empirical privacy leakage, as measured by AUROC, decreasing from 65.4% to 55.7%. Following Carlini et al. [65], Table 5 reports the TRP at a 1% FPR, which measures the attack when the attacker makes only 1% false positives. Across the six attacks, NOESIS has a lower (better) TPR, compared to a Mix-LoRA model trained without DP guarantees.

5 Related Work

Routing-based models have been extensively studied to build composable and modular language models [21, 23]. The mixture-of-experts architectures consist of multiple "expert" sub-models, each specialized for a particular domain or task, and a "router" that dynamically selects which expert to activate for a given input sequence or token. Towards parameter-efficient fine-tuning [66], multiple works studied routed models with low-rank adapters [36]. Specifically, Li et al. [24] introduce Mix-LoRA for memory efficiency in language modeling; Wu et al. [38] introduce Mix-LoRA to study domain specialization; and Shen et al. [39] introduce Mix-LoRA for multi-modal instruction tuning. Most similar to ours, Feng et al. [37] introduces Mix-LoRA for domain-based routing on large language modeling domains, such as finance, medicine, and coding. However, those works do not study privacy and knowledge transfer. Other PEFT approaches besides prompt-tuning [46] include prefix-tuning [45], in-context learning [67] or adapter learning [68, 69]. We choose a hybrid of Mix-LoRA and prompt-tuning, which has been shown to be effective for privacy-aware fine-tuning on datasets like the ones in Section 4 [47, 38].

Preserving privacy in learning algorithms has been studied, with DP being a popular formalism [32]. DP has been used to train LLMs. However, these approaches have primarily focused on applying DP to all model parameters [58, 70, 71]. Other work suggests using public data during private fine-tuning [72, 73], showing that access to public data can improve the performance of DP-trained models. In our case, we use public data as the pre-trained model initialisation, and fine-tune with smaller, private datasets. Similar to our setting, Tholoniat et al. [74] studies DP in an MoE setting but focuses on learning the routers and showing emergent expertise behaviour. In our case, we aim at private PEFT with knowledge transfer between domains. Regarding the MIA, Hu et al. [75] explores an attack in a multi-modal setting, but our work studies membership inference in the multi-domain setting where access to predictive log-likelihoods is available.

Knowledge Transfer in Modular Language Models consists of interaction between multiple domains to obtain better performance. In the context of Large Language Models, this has been studied with large-scale pretraining with various data sources [28, 29, 76]. The term was introduced by Hokamp et al. [41] and has been observed in the context of modular language models in the MultiCoder model [19]. However, this work does not consider the privacy of each domain, nor does it study the privacy vulnerability that occurs when training shared parameters in the backbone of a neural network.

6 Conclusion

This paper introduces a first-of-its-kind, hybrid PEFT architecture for safe collaboration that we call Nexus-of-Experts. Our method, NOESIS, establishes a framework for modular learning that enables knowledge transfer across domains that should remain private. Our evaluation demonstrates that NOESIS effectively balances privacy vs. utility, achieving high accuracy across domains while providing strong guarantees at $(\varepsilon = 1., \delta = 10^{-6})$-DP. At the same time, we uncover a real privacy vulnerability in baseline modular models and show how our method can defend against such attacks and protect training sources. The experimental results show that NOESIS bridges 84% of the accuracy gap between non-private and non-shared models, demonstrating the effectiveness of our hybrid approach. As AI systems increasingly operate across institutional and national boundaries, new developments will be essential for ensuring that technological advancement proceeds in a manner that respects privacy rights and enables safe collaboration between organizations. We believe NOESIS with modularity, privacy, and knowledge transfer properties is one important step in that direction.

References

[1] Xin Luna Dong, Seungwhan Moon, Yifan Ethan Xu, Kshitiz Malik, and Zhou Yu. Towards next-generation intelligent assistants leveraging llm techniques. In *Proceedings of the SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[2] Chen et al. Evaluating large language models trained on code. *arXiv preprint 2107.03374*, 2021.

[3] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. In *International Conference on Machine Learning (ICML)*.

[4] Nachiket Kotalwar, Alkis Gotovos, and Adish Singla. Hints-in-browser: Benchmarking language models for programming feedback generation. *arXiv preprint 2406.05053*, 2024.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint 2001.08361*, 2020.

[6] Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *International Conference on Machine Learning (ICML)*, 2024.

[7] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631, 2024.

[8] Alex Iacob, Lorenzo Sani, Bill Marino, Preslav Aleksandrov, William F Shen, and Nicholas Donald Lane. Worldwide federated training of language models. *arXiv preprint 2405.14446*, 2024.

[9] Jialiang Xu, Shenglan Li, Zhaozhuo Xu, and Denghui Zhang. Do LLMs know to respect copyright notice? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*. Association for Computational Linguistics, 2024.

[10] EU-regulation. Regulation (eu) 2024/1689. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R1689`. Accessed: 2024-11-11.

[11] Preston Bukaty. *The California Consumer Privacy Act (CCPA): An implementation guide*. IT Governance Publishing, 2019. ISBN 9781787781320.

[12] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security Symposium*. USENIX Association, 2021.

[13] OpenAI. Gpt-3.5 turbo fine-tuning and api updates, August 2023. URL `https://openai.com/index/gpt-3-5-turbo-fine-tuning-and-api-updates/`. Accessed: 2025-01-26.

[14] GitHub. Github copilot, 2024. URL `https://github.com/features/copilot`. Accessed: 2024-11-11.

[15] Liang Niu, Shujaat Mirza, Zayd Maradni, and Christina Pöpper. CodexLeaks: Privacy leaks from code generation language models in GitHub copilot. In *USENIX Security Symposium*. USENIX Association, 2023.

[16] Yue Wang, Hung Le, Akhilesh Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, 2023.

[17] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC conference on computer and communications security*, 2016.

[18] Hao Zhou, Zhijun Wang, Shujian Huang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, Weihua Luo, and Jiajun Chen. Moe-lpr: Multilingual extension of large language models through mixture-of-experts with language priors routing. *arXiv preprint 2408.11396*, 2024.

[19] Zi Gong, Yinpeng Guo, Pingyi Zhou, Cuiyun Gao, Yasheng Wang, and Zenglin Xu. Multicoder: Multi-programming-lingual pre-training for low-resource code completion. *arXiv preprint 2212.09666*, 2022.

[20] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Ponti. Modular deep learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=z9EkXfvxta. Survey Certification.

[21] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. *arXiv preprint 2407.06204*, 2024.

[22] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

[23] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 2022.

[24] Dengchun Li, Yingzi Ma, Naizheng Wang, Zhiyuan Cheng, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts. *arXiv preprint 2404.15159*, 2024.

[25] Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. Transfer learning in natural language processing. In Anoop Sarkar and Michael Strube, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-5004. URL https://aclanthology.org/N19-5004/.

[26] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[27] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021.

[28] Alec Radford. Improving language understanding by generative pre-training. *OpenAI distribution network*, 2018.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*. Association for Computational Linguistics, 2019.

[30] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Elsevier, 1989.

[31] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

[32] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 2014.

[33] Foo Yun Chee. Apple to delay launch of ai-powered features in europe, blames eu tech rules, 2024. URL https://www.reuters.com/technology/artificial-intelligence/apple-delay-launch-ai-powered\-features-europe-blames-eu-tech-rules\-2024-06-21/. Accessed: 2024-12-05.

[34] Stefano Fratta. Building ai technology for europeans in a transparent and responsible way, 2024. URL https://about.fb.com/news/2024/06/building-ai-technology-for-europeans\-in-a-transparent-and-responsible-way/. Accessed: 2024-12-05.

[35] Salesforce. Introducing agentforce 2.0: The digital labor platform for building a limitless workforce, December 2024. URL https://www.salesforce.com/uk/news/press-releases/2024/12/17/agentforce-2-0-announcement/. Accessed: 2025-01-26.

[36] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

[37] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning method for large language models. In *Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING*. ELRA and ICCL, 2024.

[38] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. In *International Conference on Learning Representations (ICLR)*, 2024.

[39] Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. Multimodal instruction tuning with conditional mixture of lora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2024.

[40] Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint 2110.03742*, 2021.

[41] Chris Hokamp, John Glover, and Demian Gholipour Ghalandari. Evaluating the supervised and zero-shot performance of multi-lingual translation models. In *Conference on Machine Translation, WMT*. Association for Computational Linguistics, 2019.

[42] Anurag Arnab, Xuehan Xiong, Alexey Gritsenko, Rob Romijnders, Josip Djolonga, Mostafa Dehghani, Chen Sun, Mario Lučić, and Cordelia Schmid. Beyond transfer learning: Co-finetuning for action localisation. *arXiv preprint 2207.03807*, 2022.

[43] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[44] Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Anders Freeman, Carolyn Jane Anderson, Molly Q Feldman, Michael Greenberg, Abhinav Jangda, and Arjun Guha. Knowledge transfer from high-resource to low-resource programming languages for code llms. *Proceedings of the ACM on Programming Languages*, 2024.

[45] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, 2021.

[46] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*. Association for Computational Linguistics, 2021.

[47] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of stochastic parrots: Differentially private prompt learning for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[48] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint 2109.12298*, 2021.

[49] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[50] CodeParrot. Github code dataset, 2025. URL `https://huggingface.co/datasets/codeparrot/github-code`. Accessed: 2025-01-09.

[51] Veselin Raychev, Pavol Bielik, and Martin Vechev. Probabilistic model for code with decision trees. *ACM SIGPLAN Notices*, 2016.

[52] Miltiadis Allamanis and Charles Sutton. Mining source code repositories at massive scale using language modeling. In *10th working conference on Mining Software Repositories (MSR)*. IEEE, 2013.

[53] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint 1909.09436*, 2019.

[54] Alberto Blanco-Justicia, David Sánchez, Josep Domingo-Ferrer, and Krishnamurty Muralidhar. A critical review on the use (and misuse) of differential privacy in machine learning. *ACM Computing Surveys*, 2022.

[55] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. Differential privacy: An economic method for choosing epsilon. In *IEEE Computer Security Foundations Symposium*. IEEE, 2014.

[56] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint 2102.04664*, 2021.

[57] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

[58] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR)*, 2021.

[59] Xinyu Tang, Ashwinee Panda, Milad Nasr, Saeed Mahloujifar, and Prateek Mittal. Private fine-tuning of large language models with zeroth-order optimization. *Transactions Machine Learning Research (TMLR)*, 2025.

[60] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Symposium on security and privacy (SP)*. IEEE, 2017.

[61] Hongyan Chang, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Reza Shokri. Context-aware membership inference attacks against pre-trained large language models. *arXiv preprint 2409.13745*, 2024.

[62] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint 2006.16668*, 2020.

[63] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[64] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint 2212.05055*, 2022.

[65] Nicholas Carlini, Steve Chien, a Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy, SP*, 2022.

[66] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 2023.

[67] Rob Romijnders, Mohammad Mahdi Derakhshani, Jonathan Petit, Max Welling, Christos Louizos, and Yuki M. Asano. Private poetry: Private in-context learning via product of experts. *arXiv preprint 2602.05012*, 2026.

[68] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2019.

[69] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

[70] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations (ICLR)*, 2021.

[71] Rob Romijnders and Antti Koskela. Convex approximation of two-layer relu networks for hidden state differential privacy. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[72] Gavin Kerrigan, Dylan Slack, and Jens Tuyls. Differentially private language models benefit from public pre-training. In *Proceedings of the Second Workshop on Privacy in NLP*, 2020.

[73] Aditya Golatkar, Alessandro Achille, Yu-Xiang Wang, Aaron Roth, Michael Kearns, and Stefano Soatto. Mixed differential privacy in computer vision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[74] Pierre Tholoniat, Huseyin A Inan, Janardhan Kulkarni, and Robert Sim. Differentially private training of mixture of experts models. *AAAI privacy workshop*, 2024.

[75] Pingyi Hu, Zihan Wang, Ruoxi Sun, Hu Wang, and Minhui Xue. M$ˆ4$i: Multi-modal models membership inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[76] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv 2302.13971*, 2023.