Accelerating LLM Fine-Tuning via Embedding Knowledge Transfer

Anonymous ACL submission

Abstract

Incorporating Large Language Models (LLMs) for downstream tasks has recently garnered considerable attention, where fine-tuning plays a key role in LLMs' adaptation. These LLMs, often consisting of billions of parameters, require vast amounts of computational resources when customizing them for new tasks. To mitigate this, researchers have proposed the Parameter-Efficient Fine-Tuning (PEFT) as a practical solution by adjusting fewer parameters of a pre-trained LLM. However, these methods heavily rely on their own structural modifications that fail to establish an efficient knowledge-sharing mechanism to distill rich knowledge from other expert models, which may lead to inefficient fine-tuning. In this paper, we propose **Pen2Sword**, a lightweight fine-tuning framework for domain adaptation which efficiently transfers knowledge from a small expert model to a target large model via embedding layers, significantly enhancing the fine-tuning efficiency of large models. Specifically, we first selects optimal expert models via a preserving function, then facilitates knowledge transfer through vocabulary alignment and embedding expansion, and finally accelerates domain adaptation with a fast fine-tuning paradigm. Extensive empirical evaluations across multiple domains demonstrate that our Pen2Sword framework consistently accelerates domain-specific fine-tuning, improves model performance (e.g., +13.6% in code and +20.1% in math), and remains robust across diverse model families and PEFT methods. The codes and data are available at https://anonymous.4open.science/r/Pen2Sword.

1 Introduction

Rapid advancement of Large Language Models (LLMs), characterized by billions of parameters, has led to significant improvements in a wide range of Natural Language Processing (NLP) tasks (Hadi et al., 2023). While fine-tuning has produced many state-of-the-art results by adapting LLM to a new task, it requires substantial training data and time to enhance model performance (Lee et al., 2022).

Fortunately, existing studies on Parameter-Efficient Fine-Tuning (PEFT) (Hu et al., 2021; Dettmers et al., 2024; Liu et al., 2024b) reduce computational costs by updating only a small subset of parameters, enabling the proliferation of expert models for various domains. This progress naturally aligns with the Weak-to-Strong paradigm (Burns et al., 2023; Liu et al., 2024a), which leverages such small models to guide general-purpose LLMs in adapting to unseen tasks via pseudo-labels. Intuitively, combining PEFT with the Weak-to-Strong paradigm offers an improved training scheme. It has the potential to lower computational costs by leveraging small expert models to facilitate the adaptation of large models to new tasks. However, generating pseudo-labels remains time-consuming and memory-intensive, as it requires small model inference and co-loading both large and small models on the GPU. These limitations underscore the need for a more efficient method to enable targeted knowledge transfer with substantially reduced computational and memory costs.

To resolve the above issues, this paper aims to develop an effective parameter expansion mechanism (Chen et al., 2016, 2021a) that enables knowledge transfer by integrating new knowledge and preserving learned representations while adapting to new tasks. Despite its potential, two fundamental challenges hinder its application: (i) The parameter size mismatch between small expert models and large target models makes direct parameter transfer infeasible. (ii) Architectural inconsistencies across models pose significant barriers to designing generalizable knowledge transfer mechanisms. To address these challenges, we introduce **Pen2Sword**, a lightweight framework for ad-

vanced knowledge transfer via embedding layer. Pen represents a small expert model, while Sword refers to a target large general model. It consists of three key components: domain knowledge initialization, advanced knowledge transfer, and fast fine-tuning. Domain knowledge initialization module applys a preserving function to select expert models whose embeddings are most aligned with the target domain. Advanced knowledge transfer module performs vocabulary alignment and embedding expansion to enable efficient parameter transfer between expert and target models. Fast fine-tuning module fine-tunes the target model using the transferred embeddings to accelerate convergence and improve domain-specific performance.

We validate the effectiveness of Pen2Sword across diverse domains including code generation, mathematical reasoning, and general knowledge tasks. Experiments show that Pen2Sword consistently outperforms baselines by achieving faster convergence and stronger final performance. It also generalizes well to unseen tasks such as Physics and Politics without fine-tuning on those domains. Robustness analyses demonstrate compatibility with multiple PEFT strategies and stable performance across model families. Ablation results show that hidden size expansion and weighted fusion are key to the advanced knowledge transfer module's effectiveness. We further confirm that appropriate choices of dimension and fusion ratios empirically lead to faster convergence and more stable knowledge transfer in Pen2Sword.

Our contributions are summarized as follows:

- To the best of our knowledge, we are among the first to accelerate fine-tuning by transferring domain knowledge via embedding matrix, without relying on pseudo-labels or decoder-level supervision.
- We design a novel embedding expansion approach that combines vocabulary and hidden size alignment with weighted fusion, enabling the transferred embeddings to integrate domain knowledge from the small expert model while preserving the representation space of the large target model for faster convergence.
- Extensive experiments conducted across various domains demonstrate the Pen2Sword facilitates more efficient adaptation, improves final performance, generalizes to unseen tasks, and exhibits robustness across model families and PEFT methods.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) reduces resource demands by updating a small subset of parameters, achieving performance comparable to Full Fine-Tuning (FFT) (Houlsby et al., 2019; Lialin et al., 2023). However, methods like LoRA (Hu et al., 2021), QLoRA (Dettmers et al., 2024), and DoRA (Liu et al., 2024b) only updated Transformer block weights while neglecting important layers, such as embeddings. This limitation hindered domain-specific token integration and adaptability, especially in specialized domains, and prevents the capture of complex dependencies (Bhojanapalli et al., 2020; Zhang et al., 2022). Our method bridges this gap with task-specific embedding updates, enhancing domain integration without increasing computational cost.

2.2 Weak-to-Strong Paradigm

Weak-to-Strong paradigm uses a small model for supervision that inspires a large model's potential, gradually improving its performance to approach that of fully fine-tuned models (Burns et al., 2023). Methods such as Weak-to-Strong (Burns et al., 2023; Guo and Yang, 2024) used pseudo-labels from a small model to guide training, while Proxy-Tuning (Liu et al., 2024a) adjusted the predictions of a large untuned model based on the discrepancy between pseudo-labels from tuned and untuned small models. However, these approaches required substantial computational resources to generate pseudo-labels, limiting scalability and introducing a trade-off between resource efficiency and model performance. Additionally, Weak-to-Strong paradigm focused primarily on improving general abilities rather than domain-specific transfer, which limits its adaptability to specialized tasks. Despite these limitations, Weak-to-Strong paradigm remains valuable, as it leverages the crucial role of small models to enhance the learning efficiency of large models (Chen and Varoquaux, 2024).

2.3 Efficient Parameter Expansion

Parameter Expansion is a model growth methodology that uses small models to initialize large models, which helps improve the training efficiency of Transformers (Pandey et al., 2025). Net2net (Chen et al., 2016) introduced parameter expansion via Function-Preserving Initialization (FPI) while bert2BERT (Chen et al., 2021a)

extended FPI with Advanced Knowledge Initialization (AKI), leveraging upper-layer knowledge. ELLE (Qin et al.) introduced random noise into FPI, thereby accelerating subsequent pre-training. However, these methods primarily targeted pre-training scenarios and faced significant challenges, such as performance degradation and limitations in model scaling, whereas we extend the applicability to fine-tuning, thereby enhancing domain-specific adaptation and overall model efficiency.

3 Preliminary

Before introducing methodology, we summarize the key architectural components of Llama, Vicuna, and Baichuan (Touvron et al., 2023; Yang et al., 2023; Chiang et al., 2023; Vaswani et al., 2017). This includes the tokenizer, embedding layer, Transformer blocks, and output layer.

3.1 Tokenizer

The tokenizer converts raw text into subword tokens, which are used as input to the model. The set of all recognizable tokens is referred to as the vocabulary, whose size is denoted by k, and is typically shared across models within the same family. To support domain adaptation, additional domain-specific tokens are sometimes introduced. For example, CodeLlama-7B (Rozière et al., 2024) incorporates 16 more special tokens than Llama2-13B-Base, resulting in different k.

3.2 Embedding Layer

The embedding layer maps tokens to continuous representations using a learnable matrix $\mathbf{E} \in \mathbb{R}^{k \times d}$, where d is the hidden size. In this work, we focus on the embedding matrix due to its structural universality across transformer-based architectures and its foundational role in encoding token-level semantics. As the initial layer of the model, it determines how lexical information is projected into the model's latent space and directly influences all subsequent representations. This makes it a natural and effective target for transferring domain knowledge across models with different capacities.

3.3 Transformer Blocks

Each Transformer block (Vaswani et al., 2017) consists of three components: (i) multi-head self-attention, which computes token-level dependencies by attending to different positions in parallel; (ii) a feed-forward network, which applies non-linear transformations to each token independently; (iii) layer normalization, which stabilizes training and improves convergence.

3.4 Output Layer

The output layer projects the final representations into a task-specific output space using a linear transformation. This layer typically matches the dimensionality and format required by the downstream objective, such as predicting class labels, generating tokens, or producing numerical values.

4 Methodology

In this section, we present the design details of **Pen2Sword**, a lightweight fine-tuning framework for efficient knowledge transfer from small expert models to target large models.

4.1 Problem Statement

We aim to accelerate the fine-tuning of a large model $\mathcal{T}$ by transferring knowledge of an existing small expert model $\mathcal{S}$. Throughout the paper, $d_{\mathcal{T}}$ and $d_{\mathcal{S}}$ denote the hidden sizes of the models, where $d_{\mathcal{S}} < d_{\mathcal{T}}$; $k^{\mathcal{T}}$ and $k^{\mathcal{S}}$ represent the vocabulary sizes. Formally, our problem is a weak to strong fine-tuning problem: how can the model $\mathcal{S}$ accelerate the fine-tuning of the model $\mathcal{T}$? To formalize the fine-tuning process, we define the optimization problem for the model $\mathcal{T}$ as follows:

$$\max_{\theta \subseteq \Theta} \sum_{(x,y) \in \mathcal{D}} \sum_{t=1}^{N} \log(P_\theta(y_t \mid x, y_{<t})). \quad (1)$$

Here, $P_\theta(y \mid x)$ refers to the pre-trained autoregressive language model $\mathcal{T}$'s parameters, where θ denotes the trainable parameters and Θ denotes all parameters, including trainable and frozen ones.

4.2 Overview

As shown in Figure 1, Pen2Sword is a lightweight framework designed for efficient knowledge transfer from small expert models to larger target models. It operates in three stages. **(i) Domain Knowledge Initialization** employs a preserving function to dynamically select optimal expert models, ensuring domain relevance. **(ii) Advanced Knowledge Transfer** establishes knowledge flow through vocabulary alignment and latent embedding matrix operations. **(iii) Fast Fine-Tuning** leverages adjusted embedding layers to accelerate convergence compared to standard approaches. For implementation details, see Appendix A.1.

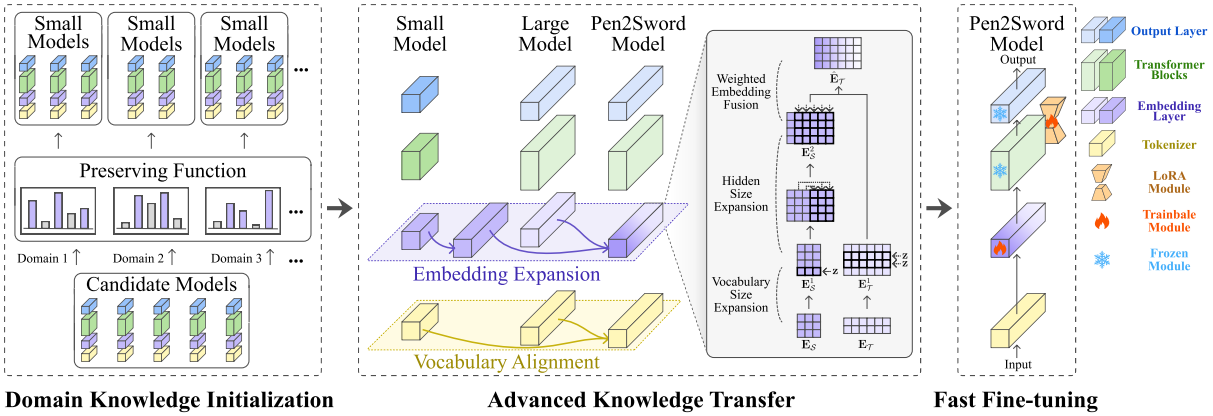


Figure 1: The overall framework of **Pen2Sword**.

4.3 Domain Knowledge Initialization via Preserving Function

We define a domain-preserving function $g(\mathcal{S}, \mathcal{D}) \in [0, 1]$ to select models specialized in a target domain. Here, $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ represents the set of candidate models, and $\mathcal{D}$ denotes the target domain. The function g evaluates the relevance of each candidate model $\mathcal{S}_i \in \mathcal{S}$ to the target domain $\mathcal{D}$ based on predefined metrics (e.g., feature-space similarity, domain-task performance, and transfer-learning potential). To provide a quantitative assessment, we evaluate candidate models along multiple criteria, using GPT-4o as the evaluation engine (detailed criteria and scoring are provided in Appendix A.2). The models with the highest relevance scores are selected as small expert models for subsequent knowledge transfer.

4.4 Advanced Knowledge Transfer via Latent Embeddings

This module tackles two challenges in knowledge transfer: (1) vocabulary inconsistency, and (2) dimensional mismatch of embedding matrices.

Vocabulary Alignment. Even within the same model series, small vocabulary differences between the small expert and the target large model can hinder lexical-level alignment. To resolve these minor differences, we unify the vocabularies of both models by constructing $\mathbf{V} = \mathbf{V}_{\mathcal{S}} \cup \mathbf{V}_{\mathcal{T}}$, where $\mathbf{V}_{\mathcal{S}}$ and $\mathbf{V}_{\mathcal{T}}$ denote the vocabularies of the small and large model, respectively.

Embedding Expansion. We propose a novel embedding expansion mechanism that aligns the vocabulary sizes and dimensions of the small and large model embeddings, and then constructs the Pen2Sword embedding via weighted fusion.

Vocabulary Sizes Expansion. We first align their embedding dimensions by expanding both matrices to accommodate the unified vocabulary size $k = |\mathbf{V}|$. The expanded matrices $\mathbf{E}^1_{\mathcal{S}} \in \mathbb{R}^{k \times d_{\mathcal{S}}}$ and $\mathbf{E}^1_{\mathcal{T}} \in \mathbb{R}^{k \times d_{\mathcal{T}}}$ are constructed as:

$$\mathbf{E}^1_{\mathcal{S}}[v, :] = \begin{cases} \mathbf{E}_{\mathcal{S}}(v) & \text{if } v \in \mathbf{V}_{\mathcal{S}} \\ \mathbf{z} & \text{if } v \notin \mathbf{V}_{\mathcal{S}} \end{cases} \tag{2}$$

$$\mathbf{E}^1_{\mathcal{T}}[v, :] = \begin{cases} \mathbf{E}_{\mathcal{T}}(v) & \text{if } v \in \mathbf{V}_{\mathcal{T}} \\ \mathbf{z} & \text{if } v \notin \mathbf{V}_{\mathcal{T}} \end{cases} \tag{3}$$

where $v \in \mathbf{V}$ and $\mathbf{z} \in \mathbb{R}^d$, with $d = d_{\mathcal{S}}$ for $\mathbf{E}^1_{\mathcal{S}}$ and $d = d_{\mathcal{T}}$ for $\mathbf{E}^1_{\mathcal{T}}$. The vector $\mathbf{z} \sim \mathcal{N}(0, 1)$ is a randomly initialized row vector sampled from a standard Gaussian distribution.

Hidden Sizes Expansion. The embedding dimension mismatch between a small model $\mathcal{S}$ and a large model $\mathcal{T}$ ($d_{\mathcal{S}} < d_{\mathcal{T}}$) precludes direct parameter transfer. Motivated by Chen et al. (2021a)'s layer expansion technique, we address this by expanding the small expert model's embedding matrix to match the target large model's hidden size while preserving functional equivalence. We define the expansion process from $\mathbf{E}^1_{\mathcal{S}} \in \mathbb{R}^{k \times d_{\mathcal{S}}}$ to $\mathbf{E}^2_{\mathcal{S}} \in \mathbb{R}^{k \times d_{\mathcal{T}}}$ as follows:

$$f(i) = \begin{cases} i & \text{if } i \in [1, d_{\mathcal{S}}] \\ u([1, d_{\mathcal{S}}]) & \text{if } i \in [d_{\mathcal{S}} + 1, d_{\mathcal{T}}], \end{cases} \tag{4}$$

$$c(i) = \sum_{j=1}^{d_{\mathcal{T}}} \mathbb{I}(f(j) = f(i)), \tag{5}$$

$$\mathbf{E}^2_{(*,i)} = \frac{1}{c(i)} \mathbf{E}^1_{(*,i)}, \tag{6}$$

where $u(\cdot)$ denotes uniform sampling from $[1, d_{\mathcal{S}}]$ used to assign source indices for the newly added

dimensions, and $f(\cdot)$ maps each target index to a corresponding source index in the original embedding. $\mathbb{I}(\cdot)$ is the indicator function, and $c(i)$ counts how many target dimensions are mapped to the same source index i, serving to normalize replicated embeddings and maintain relative scale.

Weighted Embedding Fusion. Inspired by prior findings that good initializations lead to faster convergence and better final performance (Qin et al.; Wang et al.), we propose the following fusion strategy to combine these two embedding spaces:

$$\hat{\mathbf{E}}_{\mathcal{T}} = \alpha \mathbf{E}^2_{\mathcal{S}} + (1-\alpha)\mathbf{E}^1_{\mathcal{T}}, \tag{7}$$

where $\alpha \in (0,1)$ represents the fusion ratio between the small and large model embeddings. We set $\alpha = 0.5$ based on empirical evidence.

4.5 Fast Fine-tuning via Dual Stages

To enhance the fine-tuning efficiency of the Pen2Sword model, we propose a dual-stage fine-tuning approach: **(i) Embedding Domain Adaptation:** If the embedding layer of the small model has not sufficiently captured domain-specific semantics, we optionally perform a warm-up stage prior to advanced knowledge transfer. Specifically, we continue pre-training the embedding layer on domain-specific corpora while freezing all other layers. In most cases, existing small expert models can be directly used without any additional training. **(ii) Domain-Aware Fine-tuning:** After advanced knowledge transfer, the Pen2Sword model is fine-tuned on domain-specific data. It is designed to work with parameter-efficient methods like LoRA, so the main computational cost comes from this stage, while embedding transfer helps the model converge faster without significantly increasing trainable parameters or memory usage (detailed discussion in Appendix A.6).

5 Experiments

In this section, we conduct experiments to evaluate Pen2Sword's performance on multiple domains and its generalization to unseen tasks. We further analyze the impact of expansion, fusion, and freezing strategies, their compatibility with different PEFT methods and model families, and the effects of the dimension ratio and embedding fusion ratio.

5.1 Experimental Settings

Datasets. We conduct domain-specific fine-tuning and evaluation on four categories: (i) Code Ability: fine-tuned on evol-codealpaca-v1 (Luo et al., 2023), evaluated on HumanEval (Chen et al., 2021b) and the MMLU Computer Science (CS) subset; (ii) Math Ability: fine-tuned on MetaMathQA (Yu et al., 2023), evaluated on MATH level-5 problems[1] (Hendrycks et al., 2021) and the MMLU Mathematics subset; (iii) Specialized Adaptation: evaluated on MMLU Physics and Politics subsets; (iv) General Ability: evaluated on the full MMLU benchmark (Hendrycks et al., 2020).

Models & Baselines. Our candidate models consist of Llama2-7B-Chat, CodeLlama-7B, CodeLlama-7B-Instruct, Vicuna-7B-v1.3, and Vicuna-7B-v1.5, whereas Llama2-13B-Base is designated as the target large model. We evaluate our method in comparison with five baselines: (i) Vanilla Model, the original pretrained model without any adaptation; (ii) +Instruction-tuned, adapted with instruction-following data; (iii) +LoRA, adapted to domain-specific data via LoRA (Hu et al., 2021); (iv) +Weak2Strong, adapted using pseudo-labels generated solely by a small expert model to supervise the target large model (Burns et al., 2023); and (v) +ProxyTuning[2], adapted using pseudo-labels jointly derived from a small expert model, a small anti-expert model, and a target large model (Liu et al., 2024a).

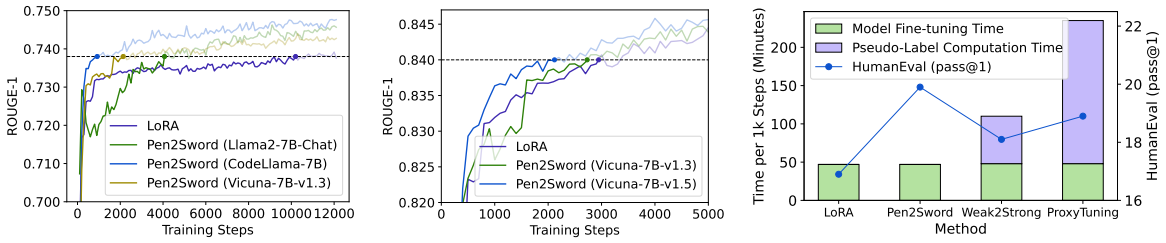
Evaluation Metrics. We use ROUGE-1/2/L to evaluate performance during fine-tuning, and report final results on HumanEval (pass@1), MATH (equivalent accuracy), and MMLU (accuracy). We build our benchmark implementation on HELM (Liang et al., 2022)[3].

Implementation Details. We executed all experiments on Linux servers using PyTorch with NVIDIA A800 GPUs and RTX 3090s. We fine-tuned token embeddings for all model families, and additionally fine-tuned query projections in Llama/Vicuna and attention weight packs in Baichuan. Our training protocol used 512-token sequences. We evaluated the model every 1000 steps on 100 validation samples and selected the best checkpoint based on ROUGE-1 score. For LoRA

[1] Level-5 represents the highest difficulty, assessing advanced mathematical proficiency.

[2] Although proxy-tuning was originally proposed as a decoding-time adaptation method to guide large models via small models without modifying their weights, we adapt it to generate pseudo-labels that supervise subsequent fine-tuning of the large model.

[3] https://github.com/stanford-crfm/helm



(a) Code domain. (b) Math domain. (c) Time vs performance.

Figure 2: **Fine-tuning efficiency comparison.** (a) and (b) compare training efficiency on code and math tasks, where Pen2Sword achieves comparable performance with fewer training steps. The specific small models used by Pen2Sword are shown in parentheses. (c) compares time per 1k training steps and HumanEval performance against baseline methods.

	Code Ability		Math Ability		Specialized Adaptation		General Ability
	HumanEval	CS	MATH	Mathematics	Physics	Politics	MMLU
Vanilla Model	15.9 (0.0%)	50.2 (0.0%)	1.9 (0.0%)	34.0 (0.0%)	40.8 (0.0%)	71.1 (0.0%)	**55.7** (0.0%)
+ Instruction-tuned	18.3 (+15.1%)	47.0 (-6.4%)	2.1 (+10.5%)	32.0 (-5.9%)	40.8 (0.0%)	71.6 (+0.7%)	53.6 (-3.8%)
+ LoRA	16.9 (+6.3%)	50.0 (-0.4%)	2.3 (+21.1%)	33.8 (-0.5%)	41.9 (+2.7%)	71.8 (+1.0%)	55.0 (-1.3%)
+ Weak2Strong	18.1 (+13.8%)	50.5 (+0.6%)	1.8 (-5.3%)	34.8 (+2.3%)	41.3 (+1.2%)	71.5 (+0.6%)	**55.7** (0.0%)
+ ProxyTuning	18.9 (+18.9%)	50.7 (+1.0%)	1.9 (0.0%)	33.7 (-0.8%)	41.7 (+2.2%)	71.5 (+0.6%)	55.6 (-0.2%)
+ Pen2Sword	**19.9** (+25.2%)	**51.2** (+2.0%)	**2.6** (+36.8%)	**35.2** (+3.4%)	**44.8** (+9.8%)	**72.1** (+1.4%)	54.7 (-1.8%)

Table 1: **Main performance comparison across methods and tasks. Vanilla Model** refers to the pretrained Llama2-13B-Base model without any adaptation. **+Instruction-tuned** refers to the pretrained Llama2-13B-Chat model, which has been instruction-tuned. **+LoRA** fine-tunes the vanilla model using domain-specific data. **+Weak2Strong**, **+ProxyTuning**, and **+Pen2Sword** fine-tune the vanilla model by incorporating supervision transfer, proxy alignment, and embedding knowledge transfer, respectively. Code/Math Ability results are averaged over tasks fine-tuned with code/math data. Specialized Tasks and General Ability results are averaged over tasks evaluated with models fine-tuned on code or math data. Best results are shown in **bold**. Performance changes are in the parentheses.

configuration, we set rank $r = 8$, scaling factor $\alpha = 16$, and dropout rate 0.05. We trained with AdamW ($\eta = 2 \times 10^{-5}$) using cosine learning rate scheduling (100 warmup steps, ratio=0.03), maintaining FP16 precision with batch sizes of 8.

5.2 Overall Performance

Pen2Sword demonstrates consistent advantages in both effectiveness and efficiency on code and math tasks. As illustrated in Figure 2(a) and Figure 2(b), Pen2Sword requires significantly fewer training steps than LoRA to reach the same ROUGE-1 score, reducing the steps from 10k to 1k in the code domain and from 3k to 2k in the math domain. It also achieves a higher ROUGE-1 score under the same training budget. In Table 1, numbers in parentheses indicate relative improvement over the Vanilla Model. This dual advantage in convergence speed and optimization efficiency is further reflected in the results, where Pen2Sword obtains the best final performance on HumanEval (19.9, +25.2%) and MATH (2.6, +36.8%), outperforming all baselines. While Weak2Strong and ProxyTuning transfer knowledge through pseudo-labels, the resulting supervision lacks semantic precision and tends to amplify model bias, which limits its generalization and leads to weaker downstream performance. In contrast, Pen2Sword avoids explicit label construction and instead aligns latent representations between models, enabling more stable optimization and consistently stronger results. As illustrated in Figure 2(c), Pen2Sword require a similar training cost to LoRA, with both taking approximately 47 minutes per 1k training steps. This is significantly more time-efficient than the Weak2Strong and ProxyTuning approaches, which require 110 and 235 minutes per 1k training steps, respectively. Regarding final performance metrics, Pen2Sword achieves a pass@1 score that exceeds that of LoRA by 17.7%, while outperforming Weak2Strong and ProxyTuning by 9.9% and 5.3%, respectively. In contrast to Weak2Strong and ProxyTuning, which introduce additional overhead for pseudo-label generation, Pen2Sword avoids

Variant	ROUGE-1	ROUGE-2	ROUGE-L
Pen2Sword	**0.745**	**0.479**	**0.653**
w/o Expansion	0.394	0.138	0.292
w/o Fusion	0.693	0.395	0.577
w/o Embedding Training	0.743	0.476	0.649

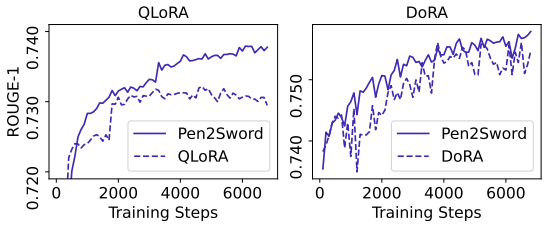
Table 2: Ablation study on the contribution of each component in Pen2Sword. Using CodeLlama-7B as the expert model and Llama2-13B-Base as the target model. Results are reported on code-domain fine-tuning after 6k steps.

such preprocessing, making it more suitable for compute-constrained or large-scale settings.

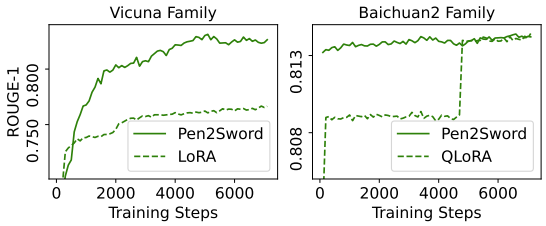
To evaluate cross-domain generalization, we test models fine-tuned on code or math data using specialized task benchmarks. Without access to Physics or Politics domain data during fine-tuning, Pen2Sword achieves the best performance on both tasks (+9.8% and +1.4%), while other methods show limited improvemnet to unseen domains. Although it incurs a slight drop (–1.8%) on MMLU compared to the vanilla model, overall performance remains competitive, indicating no degradation in general-domain ability. These results suggest that Pen2Sword supports more effective transfer to both domain-specific and general-purpose tasks without requiring task-specific supervision.

5.3 Ablation Study

We conduct ablation experiments to investigate the impact of the key components in Pen2Sword. We examine three variants of our method: **(i) w/o Expansion**: removes the hidden size expansion process and replaces the missing parameters with randomly initialized Gaussian noise. **(ii) w/o Fusion**: removes the weighted embedding fusion process and directly uses the expanded embedding matrix E_S^2 for subsequent fast fine-tuning. **(iii) w/o Embedding Training**: freezes the embedding layer after knowledge transfer and during fine-tuning. As shown in Table 2, removing any of the proposed components results in a noticeable performance degradation, confirming the effectiveness of each module in Pen2Sword. Among them, the hidden size expansion plays the most critical role: removing it causes a dramatic drop in ROUGE scores (e.g., ROUGE-L decreases from 0.653 to 0.292), indicating that appropriately increasing the representation capacity is essential for effective knowledge transfer between heterogeneous models. Weighted embedding fusion also contributes substantially; without it, the model maintains reasonable performance but shows a notable decline (e.g., ROUGE-L drops from 0.653 to 0.557), suggesting a reduced ability to integrate knowledge from small and large embeddings. In contrast, freezing the embedding layer only results in a marginal decrease, implying that most transferable knowledge is captured during the expansion and fusion stages, while subsequent embedding updates provide limited additional gains.



(a) Generalization across PEFT methods. The small model is CodeLlama-7B.



(b) Generalization across model families. The small model is Vicuna-7B-v1.5 for the Vicuna family and Baichuan2-7B-Chat for the Baichuan family.

Figure 3: Robustness analysis of Pen2Sword across different PEFT methods and model families.

5.4 Robustness Analysis

We evaluate the robustness of Pen2Sword under different PEFT methods and across diverse model families. Figure 3(a) shows that Pen2Sword consistently improves both convergence speed and ROUGE-1 score when combined with QLoRA and DoRA, suggesting that our Pen2Sword framework is compatible with diverse fine-tuning strategies. Figure 3(b) shows that Pen2Sword outperforms baselines on both Vicuna and Baichuan2 models, indicating robust generalization across architectures and vocabulary spaces. These results demonstrate that Pen2Sword remains effective under varied training configurations and model backbones. These results demonstrate that Pen2Sword remains effective under varied training configurations and model backbones, with additional experiments on larger and newer models provided in Appendix A.7 further confirming its robustness.

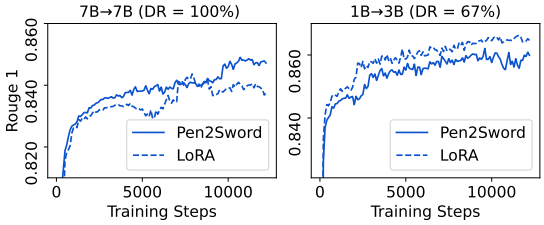


Figure 4: Performance comparison of Pen2Sword under different dimension ratio (DR). When the DR is 100%, the small model is Vicuna-7B-v1.5 and the large model is Vicuna-7B-v1.5. When the DR is 67%, the small model is Llama3.2-1B-Instruct and the large model is Llama3.2-1B-Base.

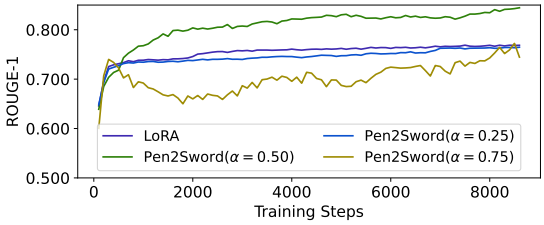


Figure 5: Performance comparison of Pen2Sword under different fusion ratios (α). The small model is vicuna-7B-v1.5 and the large model is Vicuna-13B-v1.3.

5.5 Influence of Dimension Gap

Our embedding expansion mechanism addresses hidden size mismatch by replicating and uniformly sampling low-dimensional embeddings to match the target space. While preserving parameter scale, this process may introduce redundancy and noise when the dimension gap is large. To investigate its impact, we define the Dimension Ratio (DR) as the hidden size of the small model divided by that of the target model. As shown in Figure 4 (left), when DR = 100% (e.g., 7B→7B), Pen2Sword performs well without embedding distortion. However, since the target model's hidden size matches the expert, the whole process does not increase model capacity. The added computation cost outweigh the limited performance gain, reducing the setting's practical value. As shown in Figure 4 (right), when DR = 67% (e.g., 1B→3B), performance degrades significantly. The small model expands its embedding by oversampling a limited set of dimensions, resulting in degraded downstream performance due to diluted expert information. In our main experiments, DR = 80% (e.g., 7B→13B) offers the best trade-off. It provides sufficient dimensional alignment to preserve expert knowledge while limiting the degree of expansion required, thus maintaining embedding quality. This balance enables Pen2Sword to achieve stable optimization and strong downstream performance with minimal overhead.

5.6 Analysis of Embedding Fusion Ratios

The fusion ratio α is a pivotal hyperparameter in Pen2Sword, governing the weighted combination of small expert and target large model embeddings during initialization. As illustrate in Figure 5, when α is small (e.g., 0.25), the fused representation remains close to the large model's original embedding. This leads to behavior similar to LoRA, offering limited performance gains and no notable acceleration. In contrast, large α values (e.g., 0.75) introduce stronger expert priors but incur a significant representation shift, requiring more training steps to align the embedding space with the large model's downstream layers. Notably, an intermediate α values (e.g., 0.50) strikes a balanced trade-off, which transfers expert knowledge effectively without disrupting the large model's compatibility. This setting enables our Pen2Sword method to reach the performance level of an 8k step LoRA model within just 1k steps, demonstrating substantial efficiency gains. This suggests that a moderate fusion ratio can significantly accelerate fine-tuning.

6 Conclusion

In this paper, we propose a lightweight fine-tuning framework for accelerating domain-specific adaptation of target large model by transferring embedding representations from small expert models. The framework first identifies domain-relevant experts using a preserving function, then aligns vocabulary and hidden dimensions between models, and constructs a fused embedding via weighted combination of expert and target representations. The transferred embedding serves as a strong starting point for downstream fine-tuning. We evaluate Pen2Sword on code and math domain across diverse model families and fine-tuning methods. Results on HumanEval, MATH, and MMLU show that Pen2Sword consistently improves convergence and task performance over competitive baselines, including vanilla model, LoRA, Weak2Strong, and ProxyTuning. Our method achieves efficient knowledge transfer without task-specific supervision and generalizes well across architectures and training configurations.

Limitations

First, we currently showcase Pen2Sword's effectiveness in accelerating the fine-tuning process but have not yet explored its application in other scenarios, such as inference. Second, Pen2Sword performs well when the dimension ratio between the small expert and large target model is at least 80%. For cases with larger dimension gaps, further research is needed to develop more effective methods to ensure robust knowledge transfer. Finally, due to limitations in our computational resources, the implementation has not yet been scaled to very large models, such as those with 70B or 405B parameters. Addressing these challenges will be a focus of our future work.

References

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. Low-rank bottleneck in multi-head attention models. In *International conference on machine learning*, pages 864–873. PMLR.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2021a. bert2bert: Towards reusable pretrained language models. *arXiv preprint arXiv:2110.07143*.

Lihu Chen and Gaël Varoquaux. 2024. What is the role of small models in the llm era: A survey. *Preprint*, arXiv:2409.06857.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. *Preprint*, arXiv:1511.05641.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Yue Guo and Yi Yang. 2024. Improving weak-to-strong generalization with reliability-aware alignment. *arXiv preprint arXiv:2406.19032*.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

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

Seanie Lee, Minki Kang, Juho Lee, Sung Ju Hwang, and Kenji Kawaguchi. 2022. Self-distillation for further pre-training of transformers. *arXiv preprint arXiv:2210.02871*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024a. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

Dhroov Pandey, Jonah Ghebremichael, Zongqing Qi, and Tong Shu. 2025. A comparative survey: Reusing small pre-trained models for efficient large model training. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '24, page 56–63. IEEE Press.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. ELLE: Efficient lifelong pre-training for emerging data. *Preprint*, arxiv:2203.06311. GSCC: 0000052.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training. *Preprint*, arxiv:2303.00980. GSCC: 0000045.

Zihan Wang, Deli Chen, Damai Dai, Runxin Xu, Zhuoshu Li, and Yu Wu. 2024. Let the expert stick to his last: Expert-specialized fine-tuning for sparse architectural large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 784–801, Miami, Florida, USA. Association for Computational Linguistics.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2022. Are all layers created equal? *Journal of Machine Learning Research*, 23(67):1–28.

A Appendix

A.1 Algorithm

Algorithm 1 Pen2Sword

Input: a small model set$\{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \ldots\}$, a untuned large model $\mathcal{T}$, a small domain model training dataset $Data_1$, a supervised fine-tuning dataset $Data_2$, the small domain model training epoch $Epoch_1$, the large model training epoch $Epoch_2$, the embedding matrices of small and large model $E_{\mathcal{S}}$ and $E_{\mathcal{T}}$, and the mixing ratio α.

Step 1: Domain Knowledge Initialization
$\mathcal{S} \leftarrow g(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \ldots; Domain)$
if Domain Pretraining is enabled **then**
  for $e = 1$ to E_1 **do**
    for each $batch$ in D_1 **do**
      Perform forward and backward propagation on $\mathcal{S}$.
      Update the embedding layer of $\mathcal{S}$.
    end for
  end for
end if
Step 3: Advanced Knowledge Transfer
$E_{\mathcal{S}/\mathcal{T}} \leftarrow$ do embedding matrix expansion for vocabulary size
$E_{\mathcal{S}} \leftarrow$ do embedding matrix expansion for hidden size
$E_{\mathcal{T}} \leftarrow \alpha E_{\mathcal{S}} + (1-\alpha) E_{\mathcal{T}}$
Initialize the embedding matrix of large model $\mathcal{T}$ with $E_{\mathcal{T}}$.
Step 4: Fast Fine-tuning
for $e = 1 \rightarrow Epoch_2$ **do**
  for $batch$ in $Data_2$ **do**
    Perform forward and backward of $\mathcal{T}$.
    Update embedding layer of $\mathcal{T}$.
    Update other layers of $\mathcal{T}$.
  end for
end for

Output: The large domain-tuned model $\mathcal{T}$.

Pen2Sword addresses domain adaptation by transferring knowledge from a small expert model to a target large model through embedding matrix expansion and fast fine-tuning. The whole framework consists of three critical steps:

Domain Knowledge Initialization (Step 1): A preserving function is utilized to identify the expert models from a small model set demonstrating the highest specialization for the target domain. The relevance of each model is evaluated based on a predefined metric. In our experiment, we use performance on domain-specific tasks as the scoring criterion for their domain knowledge. The top five models with the highest scores are then selected as the expert models. Then we trained on the small domain dataset, refining its embedding layers through forward and backward propagation.

Advanced Knowledge Transfer (Step 2): The embedding matrices of both the small and large models are expanded to align their vocabulary and hidden sizes. Specifically, the large model's embedding matrix is updated by merging the small model's embedding matrix, where there is a hyperparameter controlling the fusion ratio. In our experiment, we used 0.5 as fusion ratio. After this fusion of embeddings, the combined embedding matrix is transferred to the large model as its initialization. This step ensures that the embedding layers of both models are compatible, and domain-specific knowledge is effectively transferred.

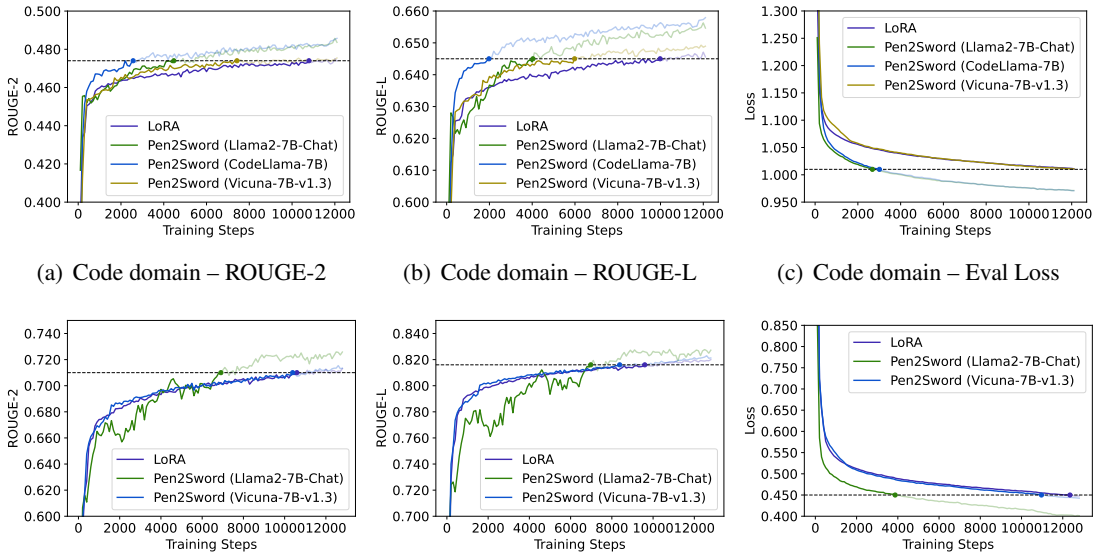
Fast Fine-tuning (Step 3): The large model is fine-tuned using the small domain dataset in two stages: first, the lower layers are refined to capture domain-specific features, followed by comprehensive updates across all layers to enhance the model's overall understanding and improve domain-specific performance. The detailed steps are outlined in Algorithm 1.

A.2 Details for Domain Preserve Function

For each candidate model, we used GPT-4o as the domain-preserving function through four steps: (i) Generate domain-specific questions covering key subdomains. (ii) Collect responses from candidate models. (iii) Score responses based on the metrics of accuracy, domain coverage, depth, and terminology appropriateness (0–5 per metric). (iv) Select top-performing models based on the evaluation scores. Here, we take the code domain as an example and report detailed scores of each candidate model in Table 6. Therefore, in Table 1, the small expert models used in Pen2Sword, Weak2Strong, and ProxyTuning are those selected by the domain-preserving function: LLaMA2-7B-chat, CodeLlama-7B, and Vicuna-7B-v1.3. The reported results correspond to the highest-performing model among them.

A.3 Extended Fine-Tuning Efficiency Results

Figure 6 complements the results in Figure 2 by presenting complete ROUGE-2/L scores and eval-



(a) Code domain – ROUGE-2 (b) Code domain – ROUGE-L (c) Code domain – Eval Loss

(d) Math domain – ROUGE-2 (e) Math domain – ROUGE-L (f) Math domain – Eval Loss

Figure 6: Training dynamics of Pen2Sword on code and math tasks. LoRA refers to directly fine-tuning the vanilla model (LLaMA2-13B-Base) on domain-specific data. The specific small models used by Pen2Sword are shown in parentheses.

Model	Accuracy	Domain Coverage	Domain Depth	Terminology Appropriateness	Total Score (0–20)
Llama2-7B-Chat	4.5	4.2	4.3	4.1	17.1
CodeLlama-7B	4.6	4.1	4.2	4.0	16.9
Vicuna-7B-v1.3	4.3	4.0	4.1	3.9	16.3
CodeLlama-7B-Instruct	4.0	3.7	3.8	3.6	15.1
Vicuna-7B-v1.5	3.9	3.6	3.7	3.5	14.7

Table 3: Domain-preserving evaluation of candidate models across multiple metrics.

uation loss on both code and math domains. These results further highlight Pen2Sword's ability to accelerate fine-tuning while preserving—or even enhancing—domain-specific performance.

Code Domain. Llama2-13B-Base fine-tuned with Pen2Sword achieves ROUGE-1 (0.73), ROUGE-2 (0.47), and ROUGE-L (0.64) within 4k training steps. In contrast, the baseline model (without Pen2Sword) requires 10k steps to reach the same performance level. This demonstrates that Pen2Sword reduces the required training steps by 60% while maintaining competitive performance. The evaluation loss rapidly drops to around 1.0 within 2k steps, showing Pen2Sword's ability to accelerate convergence by leveraging domain-specific embeddings from small models.

Math Domain. Llama2-13B-Base fine-tuned with Pen2Sword achieves a ROUGE-2 score of 0.47 and a ROUGE-L score of 0.64 by 7k steps, and reaches a ROUGE-1 score of 0.84 by 8k steps. In contrast, the baseline model requires 10k steps to achieve comparable performance. Thus, Pen2Sword reduces training cost by approximately 30% while maintaining accuracy. The evaluation loss drops to around 0.5 by 4k steps, indicating efficient optimization enabled by embedding transfer from the small expert model.

Conclusion. These results confirm the advantages of our Pen2Sword framework in accelerating fine-tuning and improving performance across domains and models. Llama2 benefits from stable and rapid convergence, Vicuna excels in embedding transfer between 7B and 13B variants, and Baichuan2 shows notable early-stage gains, especially for code generation. Across tasks and metrics, Pen2Sword consistently outperforms LoRA, demonstrating its value in optimizing large language models for domain-specific applications.

A.4 Extended Results for Benchmark Evaluation

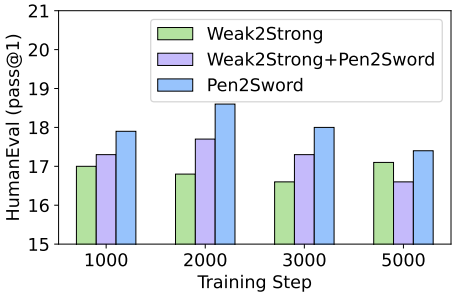
Table 4 extends Table 1 with step-wise and model-specific results on the HumanEval and MATH benchmarks. Table 4 provides the complete fine-tuning evaluation results for HumanEval and MATH benchmarks, extending the aggregated metrics in Table 1 along two additional dimensions: (i) step-wise performance across 1k–5k steps, and (ii) comparative analysis across five expert model architectures. The conclusions remain entirely consistent with those presented in Section 5.2 of the main text, while offering deeper empirical insights through these expanded evaluations.

General Model Experts. We evaluate our Pen2Sword framework using the Llama2 family, with the Llama2-7B-Chat model enhancing the performance of the Llama2-13B-Base model. As demonstrated in Table 4, in the code domain, the model with Pen2Sword significantly outperforms LoRA, achieving consistent improvements across all steps. However, in the math domain, Pen2Sword's performance is notably weaker, with reductions across all steps. This decline in performance may stem from the use of a general-purpose model as the small expert model, which may not be well-suited for tasks requiring specialized reasoning. Employing domain-specific models could offer a more effective approach to enhancing performance on such benchmarks.

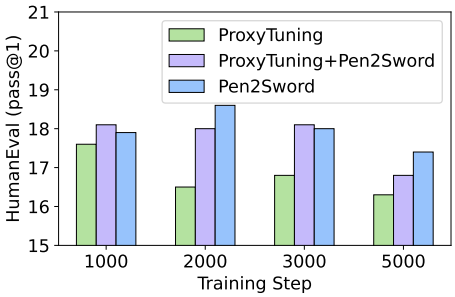
Domain-Specific Model Experts. To evaluate the effectiveness of domain-specific models within the Pen2Sword framework, we use CodeLlama-7B-Base and CodeLlama-7B-Instruct as the small models to accelerate the fine-tuning process of the Llama2-13B-Base model. As illustrated in Table 4, Pen2Sword with domain-specific experts consistently outperforms LoRA across all steps on the HumanEval benchmark. These results demonstrate the significant advantage of domain-specific models, especially in the initial fine-tuning stages, where they provide task-relevant knowledge crucial for enhancing code generation. Compared to general model experts, CodeLlama models yield substantially higher improvements in the 1k to 3k steps, demonstrating that specialized models are more effective for fine-tuning domain-specific tasks. This is particularly evident in the larger performance gains seen early on, while the rate of improvement diminishes in later stages.

Cross-Family Model Experts. To extend the applicability of Pen2Sword, we introduce Cross-Family Experts, using Vicuna-7B-v1.3 and Vicuna-7B-v1.5 to facilitate fine-tuning process of Llama2-13B-Base. These experts, though from a different family, offer complementary knowledge that enhance Pen2Sword's utility. As shown in Table 4, Pen2Sword with Cross-Family Experts outperforms LoRA in both domains. In the Code domain, Vicuna-7B-v1.3 yields a 3.6% and 16.0% improvement at 1k and 2k steps, respectively, while Vicuna-7B-v1.5 achieves a substantial 17.0% and 38.2% increase. In the Math domain, Pen2Sword with Vicuna-7B-v1.3 provides significant improvements at 1k (5.6%) and 2k (35.3%). However, Vicuna-7B-v1.5 shows a drop at higher steps, reflecting the challenge of using cross-family experts for specialized tasks like math reasoning. While Cross-Family Experts enhance performance, their effectiveness depends on the specific task, suggesting the need for task-specific expert models to further optimize performance. These results emphasize the importance of selecting the appropriate expert model for the task.

A.5 Additional Baseline Comparison Experiments



(a) Pen2Sword vs Weak2Strong.



(b) Pen2Sword vs ProxyTuned.

Figure 7: Compatibility of Pen2Sword with baselines.

In this part, Llama2-13B-Base is selected as the

	HumanEval (↑)				MATH (↑)			
	step=1k	step=2k	step=3k	step=5k	step=1k	step=2k	step=3k	step=5k
Vanilla Model	15.9 (0.0%)	15.9 (0.0%)	15.9 (0.0%)	15.9 (0.0%)	**1.9** (0.0%)	1.9 (0.0%)	1.9 (0.0%)	1.9 (0.0%)
+ LoRA	16.5 (+3.8%)	14.4 (-9.4%)	16.9 (+6.3%)	16.7 (+5.0%)	1.8 (-5.3%)	1.7 (-10.5%)	2.0 (+5.3%)	2.3 (+21.1%)
+ Weak2Strong								
+ Weak2Strong (Llama2-7B-Chat)	16.7 (+5.0%)	17.9 (+12.6%)	15.7 (-1.3%)	17.3 (+8.8%)	1.8 (-5.3%)	1.8 (-5.3%)	1.6 (-15.8%)	1.6 (-15.8%)
+ Weak2Strong (CodeLlama-7B)	18.1 (+13.8%)	17.9 (+12.6%)	17.5 (+10.1%)	16.9 (+6.3%)	-	-	-	-
+ Weak2Strong (CodeLlama-7B-Instruct)	17.1 (+7.5%)	16.7 (+5.0%)	16.7 (+5.0%)	17.3 (+8.8%)	-	-	-	-
+ Weak2Strong (Vicuna-7B-v1.3)	16.3 (+2.5%)	18.1 (+13.8%)	16.7 (+5.0%)	17.5 (+10.1%)	1.8 (-5.3%)	1.8 (-5.3%)	1.6 (-15.8%)	1.6 (-15.8%)
+ Weak2Strong (Vicuna-7B-v1.5)	17.3 (+8.8%)	15.8 (-0.6%)	17.5 (+10.1%)	16.7 (+5.0%)	1.7 (-10.5%)	1.8 (-5.3%)	1.6 (-15.8%)	1.6 (-15.8%)
+ ProxyTuning								
+ ProxyTuning (Llama2-7B-Chat)	17.1 (+7.5%)	15.4 (-3.1%)	17.1 (+7.5%)	16.7 (+5.0%)	1.7 (-10.5%)	1.8 (-5.3%)	1.8 (-5.3%)	1.7 (-10.5%)
+ ProxyTuning (CodeLlama-7B)	15.9 (0.0%)	16.5 (+3.8%)	16.1 (+1.3%)	15.5 (-2.5%)	-	-	-	-
+ ProxyTuning (CodeLlama-7B-Instruct)	18.9 (+18.9%)	17.5 (+10.1%)	16.5 (+3.8%)	14.4 (-9.4%)	-	-	-	-
+ ProxyTuning(Vicuna-7B-v1.3)	16.9 (+6.3%)	16.3 (+2.5%)	16.5 (+3.8%)	16.1 (+1.3%)	**1.9** (0.0%)	1.8 (-5.3%)	1.6 (-15.8%)	1.6 (-15.8%)
+ ProxyTuning (Vicuna-7B-v1.5)	16.9 (+6.3%)	16.5 (+3.8%)	16.9 (+6.3%)	17.7 (+11.3%)	**1.9** (0.0%)	1.8 (-5.3%)	1.6 (-15.8%)	1.7 (-10.5%)
+ Pen2Sword								
+ Pen2Sword (Llama2-7B-Chat)	17.7 (+11.3%)	17.3 (+8.8%)	17.9 (+12.6%)	16.9 (+6.3%)	1.4 (-26.3%)	1.5 (-21.1%)	1.4 (-26.3%)	1.2 (-36.8%)
+ Pen2Sword (CodeLlama-7B)	16.7 (+5.0%)	17.5 (+10.1%)	17.9 (+12.6%)	17.7 (+11.3%)	-	-	-	-
+ Pen2Sword (CodeLlama-7B-Instruct)	16.7 (+5.0%)	18.7 (+17.6%)	17.9 (+12.6%)	**18.1** (+13.8%)	-	-	-	-
+ Pen2Sword (Vicuna-7B-v1.3)	17.1 (+7.5%)	16.7 (+5.0%)	15.2 (-4.4%)	16.5 (+3.8%)	**1.9** (0.0%)	**2.3** (+21.1%)	**2.2** (+15.8%)	**2.6** (+36.8%)
+ Pen2Sword (Vicuna-7B-v1.5)	**19.3** (+21.4%)	**19.9** (+25.2%)	**18.3** (+15.1%)	17.3 (+8.8%)	1.7 (-10.5%)	1.9 (0.0%)	2.0 (+5.3%)	1.9 (0.0%)

Table 4: Performance comparison of our method and baselines. The table presents a comparison of the performance across different small expert model and fine-tuning steps on the HumanEval and MATH tasks, with scores averaged from three independent evaluations and standard deviations reported. *LoRA*,*Weak2Strong* and *ProxyTuning* are three baseline methods, and are compared with *Pen2Sword* (our proposed method). The best results within a model are highlighted in **bold**.

Method	ROUGE-1	Trainable Params (% of Total)	Avg. GPU Utilization (%)	Avg. GPU Memory (GB)
LoRA	0.735	1.28	36.21	57.17
LoRA + Pen2Sword-F	0.743	1.28	37.52	57.35
LoRA + Pen2Sword	0.745	1.31	38.69	58.95

Table 5: Comparison of ROUGE-1 performance, computational, and memory costs between LoRA-like PEFT methods and Pen2Sword variants on an NVIDIA A800 80GB GPU.

large model, with Llama2-7B-Chat, Vicuna-7B-v1.5, and CodeLlama-7B-Instruct serving as expert models to speed up its fine-tuning. To enhance the reliability of the results, each experiment was performed three times, and the outcomes were averaged to yield more stable performance metrics.

A.5.1 Pen2Sword, Weak2Strong, and Combination

We evaluate three approaches: (i) Pen2Sword: Use Pen2Sword to initialize the large model and then fine-tune it with ground-truth labels. (ii) Weak2Strong: Use Weak2Strong (Burns et al., 2023) to generate weak labels and fine-tune the model with them. (iii) Pen2Sword + Weak2Strong: Use Weak2Strong to generate weak labels, then initialize the large model using Pen2Sword, and subsequently fine-tune the model with these generated weak labels. As displayed in Figure 7(a), Weak2Strong shows a slight decline in performance from 1k to 3k training steps but achieves its highest performance at 5k steps. Pen2Sword demonstrates consistent improvements, reaching its peak performance at 2k steps, followed by a slight decline from 3k to 5k steps. When combined, Weak2Strong + Pen2Sword shows a noticeable improvement over Weak2Strong alone, particularly at 2k and 3k steps. However, Pen2Sword alone consistently outperforms both Weak2Strong and the combined approach at all steps, highlighting its superior performance, especially in long-term fine-tuning. Integrating Pen2Sword into Weak2Strong enhances its performance at each training step, leveraging Pen2Sword's stabilizing effects.

A.5.2 Pen2Sword, ProxyTuning, and Combination

We compare three approaches: (i) Pen2Sword: Use Pen2Sword to initialize the large model and then fine-tune it with ground-truth labels. (ii) ProxyTuning: Use ProxyTuning (Liu et al., 2024a) to generate weak labels and fine-tune the model with them. (iii) Pen2Sword + ProxyTuning: Use ProxyTuning to generate weak labels, then initialize the large model with Pen2Sword, and finally fine-tune it using the weak labels. As presented in Figure 7(b), ProxyTuning shows an overall downward trend in performance. In contrast, Pen2Sword demonstrates steady and continuous improvement, peaking slightly around the 2k training step, and

consistently outperforms ProxyTuning across all training steps. The combination of ProxyTuning and Pen2Sword performs similarly to Pen2Sword alone while outperforming ProxyTuning. This demonstrates that Pen2Sword not only surpasses ProxyTuning in performance but also contributes to stabilizing and enhancing the fine-tuning process when integrated with it, resulting in a consistent improvement in overall model performance.

A.6 Computational and Memory Footprint Comparison between Pen2Sword and LoRA-like PEFT Methods

To further illustrate the efficiency advantages of our method over LoRA-like PEFT approaches in terms of computational and memory resources, we analyze each stage of our framework in detail.

Fully exploiting small expert models. Our motivation stems from the increasing availability of domain-specific expert models. Pen2Sword leverages such off-the-shelf small expert models directly, without requiring any additional domain-specific fine-tuning. Consequently, this stage incurs negligible computational and memory overhead.

Fast adaptation at the embedding layer. This step involves replacing or adapting the embedding matrix, which is straightforward and computationally lightweight. It can be efficiently performed on the CPU without imposing a significant burden on GPU resources.

Efficient fine-tuning of the Pen2Sword model. The fine-tuning phase dominates the overall computational cost. The computational and memory requirements of Pen2Sword are comparable to those of LoRA-like PEFT methods. Our ablation study, particularly the Pen2Sword-F variant, demonstrates that Pen2Sword converges faster than LoRA under the same computational cost and number of trainable parameters.

Table 5 summarizes the average GPU utilization and memory usage measured on an NVIDIA A800 80GB GPU. As shown, our method introduces only marginal increases in computational cost and memory usage compared to standard LoRA, while benefiting from faster convergence and better utilization of small expert models.

Model	Score (0–10)
AlpacaEval	
Vanilla Model (Llama2-70B-Base)	4.00
+Pen2Sword (Llama2-70B-Chat)	6.10
MedQuAD	
Vanilla Model (Qwen2.5-3B-Instruct)	7.44
+Pen2Sword (medical_o1_verifier_3B_Qwen2.5)	8.86

Table 6: Performance Scores of Different Models on Two Datasets

A.7 Additional Experiments on Larger and Newer Models

To further evaluate the applicability of Pen2Sword, we conducted experiments on both larger models and newer model families. Importantly, in these experiments, we directly evaluate performance at the initial fine-tuning stage (0 steps) using the transferred embeddings, without any additional fine-tuning.

A.7.1 Evaluation on 70B Model

Following (Wang et al., 2024), we evaluated model responses using GPT-4o on the AlpacaEval dataset. By default, Pen2Sword assumes a dimension ratio of at least 80%; hence, we selected Llama-2-70B-Base as the target model and Llama-2-70B-Chat as the expert model. As shown in Table 6, even at this initial fine-tuning stage, Pen2Sword outperforms the vanilla model, demonstrating its effectiveness on larger models.

A.7.2 Evaluation on Qwen Model

We also experimented with the Qwen family in the medical domain, using the MedQuAD dataset and GPT-4o evaluation. In our setup, Qwen2.5-3B-Instruct served as the target model, and medical_o1_verifier_3B_Qwen2.5 served as the expert model. Again, Pen2Sword consistently improves performance over the vanilla models at the initial fine-tuning stage (0 steps), confirming that our method provides a strong initialization even for newer model families.