Beyond Neural Incompatibility: Cross-Scale Knowledge Transfer in Large Language Models through Latent Semantic Alignment

Jian Gu
Monash University
jian.gu@monash.edu

Aldeida Aleti
Monash University
aldeida.aleti@monash.edu

Chunyang Chen
Technical University of Munich
chun-yang.chen@tum.de

Hongyu Zhang
Chongqing University
hyzhang@cqu.edu.cn

Abstract

Large Language Models (LLMs) encode substantial knowledge in their parameters, which can be located, traced, and analyzed. Despite recent progress in neural interpretability, it is still unclear how to transfer such knowledge in a fine-grained manner, namely parametric knowledge transfer (PKT). A central challenge is to make cross-scale transfer effective and efficient when source and target models differ in architecture and parameterization. Existing methods that directly reuse layer parameters are therefore strongly limited by neural incompatibility. In this paper, we identify latent semantic alignment as the key prerequisite for cross-scale knowledge transfer. Instead of directly moving layer parameters, our approach uses activations as the transfer medium. SEMALIGN has two stages: an *layer attribution* stage that attributes task-relevant source layers and selects exactly one source layer for each target layer, and a *semantic alignment* stage that pairs them from shallow to deep and optimizes the target with source-side supervisory hidden states. The alignment is carried out in latent space. In the current realization, training follows a shallow-to-deep frontier schedule: at each stage, only the current target layer is trainable, the layer objective is a Fisher-weighted quadratic surrogate on target-space aligned logits, and the final output layer keeps KL distillation. The transferred object nonetheless remains the aligned representation itself. Evaluations on four benchmarks demonstrate the efficacy of our method. Further analysis reveals the key factors that ease cross-scale knowledge transfer and provides insights into the nature of latent semantic alignment.

1 Introduction

Language is the channel that lets humans and today's language models communicate, yet it throws away much of the fine detail that lives inside a model. When we teach a smaller model using instructions, explanations, or distilled datasets, we compress the source's rich internal signals into text and lose structure that matters for behavior. A better way to transfer knowledge would move internal states directly, similar in spirit to the idea of brainwave communication where the sender shares what it is thinking rather than what it can say. Large language models make this idea practical because their parameters and hidden states are accessible. Prior work shows that we can analyze these internals, find where knowledge lives, and measure how specific parts influence predictions using attribution methods and information flow tools (Kokhlikyan et al., 2020; Yu and Ananiadou, 2024; Ferrando and Voita, 2024; Chen et al., 2025). This builds up our motivation for better knowledge transfer, where a target should be able to receive those signals directly from a source without going through text. It promises less loss, lower cost, and more faithful transfer.

We study this idea under parametric knowledge transfer. The goal is to move internal knowledge from a larger source to a smaller target so that the target acts more like the source. Prior work explores two routes in parameter space. Seeking extracts source parameters with sensitivity measures, injects them into the target through a LoRA initialization, and then relies on post-alignment finetuning (Zhong et al., 2024). LaTen aligns parameter spaces before injection using a light mapping to reduce the cost of later training (Tan et al., 2025). Both show that knowledge transfer is possible, but also report instability when the models differ in module design and parameter values, a gap described as *neural incompatibility* (Tan et al., 2025). In our analogy above, the "brainwave communication" corresponds to sharing layer outputs, not sharing layer parameters. We therefore treat activations as the medium of transfer and align semantics first, before any parameter updates.

We propose SEMALIGN, a semantic-based approach for parametric knowledge transfer that uses

Findings of the Association for Computational Linguistics: ACL 2026, pages 21893–21905

layer outputs as the transfer signal. Compared with common extract-align-inject PKT pipelines, SemAlign introduces a two-stage workflow: First, we run layer attribution only on the source to locate layers that carry task-relevant signal, and we keep exactly one selected source layer for each target layer; Second, we perform a semantic-alignment stage: the selected source layers are paired with target layers from shallow to deep, their hidden states are decomposed into semantic components in the source space and recomposed as supervisory hidden states in the target space, and the target is optimized on those paired interfaces. In the realization, semantic alignment follows a shallow-to-deep frontier schedule. At each frontier stage, only the current target layer is trainable, all other target layers are frozen, the paired-layer objective uses a Fisher-weighted quadratic surrogate on target-space aligned logits, and the final layer keeps KL distillation. Unlike conventional intermediate-layer distillation, which directly matches source features in the source space, SemAlign first transports the supervision into the target semantic space and only then optimizes the target representation. In short, we align how layers behave rather than how weights look, which reduces neural incompatibility while keeping the procedure simple and efficient.

We evaluate SemAlign under the same setup as the prior work (Tan et al., 2025). We use four standard benchmarks on professional knowledge, mathematical reasoning, and code generation. The experiments are conducted with Llama-2 models (Touvron et al., 2023), performing task-related parametric knowledge transfer by pairing larger sources with smaller targets that differ in depth and width. Across all tasks, SemAlign improves target performance over task-matched baselines and over parameter-space transfer baselines. We concluded two main findings: First, our attribution-then-alignment workflow yields stable cross-scale transfer without an additional injection stage; Second, stagewise frontier optimization is sufficient to induce broader behavioral alignment, which keeps the method efficient in both compute and data. The replication artifact is publicly available online for open science [1].

To summarize, our contributions are as follows:

- We present a semantics-first view of parametric knowledge transfer for cross-scale language models. The formulation treats latent semantic alignment between paired layers as the prerequisite to transfer and uses layer outputs, not raw parameters, as the medium.
- We introduce SEMALIGN, which combines source-only layer attribution with a semantic-alignment stage and representation steering through semantic-based optimization. This two-stage design addresses neural incompatibility that limits parameter-space transfer in Seeking and LaTen (Zhong et al., 2024; Tan et al., 2025).
- We provide comprehensive experiments on extensive benchmarks with Llama-2. Results and analysis identify the key factors that ease cross-scale transfer and show consistent gains in efficacy. We provide further discussion in the appendix.

2 Related Work

2.1 Knowledge Attribution in Language Models

Knowledge attribution studies methods for identifying where knowledge resides in large language models and how those components influence predictions. The focus has moved from layer-level inspections to neuron-level and path-level analyses that scale to current models. One representative line designs a static, single-pass neuron score that separates "query" and "value" neurons and avoids repeated gradient passes (Yu and Ananiadou, 2024). Moving from units to mechanisms, information-flow routes rebuild prediction-time computation as a sparse graph and show how influential parts work together during inference (Ferrando and Voita, 2024). In practical analyses, CAPTUM provides operators for layer and neuron attribution, including Internal Influence, Neuron Integrated Gradients, and DeepLIFT or SHAP, which many studies adopt as reproducible baselines (Kokhlikyan et al., 2020). Recent evidence also reports degenerate knowledge neurons, where different neuron sets encode the same fact; this observation supports concept-aware or path-aware selection when using attribution to guide editing or transfer (Chen et al., 2025).

2.2 Semantic Analysis and Latent Space Alignment

Semantic analysis and latent-space alignment shape and match internal representations so that model

[1] https://github.com/jianguda/sem-transfer

adaptation preserves meaning rather than only optimizing an output loss. Within this view, a single research line proposes two connected methods that form a coherent pipeline. Vocabulary Defined Semantics (VDS) uses the model vocabulary to anchor directions in the hidden space and then clusters examples around these anchors, which stabilizes in-context learning by better matching data to the model's internal semantic frame (Gu et al., 2024). Building on that foundation, Semantic Aware Layer Freezing (SALF) treats the structure exposed by VDS as semantic anchors at the layer level and freezes those parts while tuning the remainder, which preserves core semantics and works with parameter-efficient finetuning and quantization (Gu et al., 2025). A complementary research thread adjusts hidden states at test time with small edits, showing that behavior can be steered through representation space without heavy retraining (Kong et al., 2024).

2.3 Parametric Knowledge Transfer

Knowledge transfer includes source-target distillation, representational matching across layers, and parameter mixing through model merging or task vectors. These approaches provide strong baselines and tools, yet they often work in the output space or assume closely related architectures (Xu et al., 2024; Yang et al., 2024, 2025; Liu et al., 2024). Classical distillation and hint-based supervision can also use intermediate features, but they typically match source distributions or source features directly in the source space. Our setting instead constructs a target-space supervisory signal from semantic bases, so the optimization remains tied to representation transfer across potentially incompatible latent spaces. Recent studies frame the problem as parametric knowledge transfer, where the goal is to move internal knowledge that lives inside a model, including parameters and intermediate computations such as activations and residual streams. A representative system, SEEKING, extracts sensitive components from a source, injects them into a target through LoRA initialization, and then applies post-alignment finetuning; results indicate that cross-scale transfer is feasible and that alignment quality is important for stability (Zhong et al., 2024). Follow-up work on Neural Incompatibility examines alignment as the main bottleneck across scales and distinguishes two design choices: PostPKT, which follows extract, inject, and train, and PrePKT, exemplified by LaTen, which aligns parametric spaces with light training before transfer (Tan et al., 2025). Our method adopts a semantics-first plan by using latent semantic alignment as a precondition for parametric knowledge transfer, to mitigate neural incompatibility.

3 Motivational Analysis

3.1 Preliminary: Vocabulary-Defined Semantics

For the recognizable semantic meanings of a given LM, *vocabulary-defined semantics* proposed defining a set of special representations in the latent space to associate with the labels on the vocabulary. It quantifies the semantic property of LM latent space leveraging local isotropy (Cai et al., 2021), and benefits parameter optimizations, such as efficient logits computation (Gu et al., 2024). For each label on the LM vocabulary, there is an associated representation in the latent space, termed as "semantic basis", and they share the same semantic meaning, as shown in Figure 1.

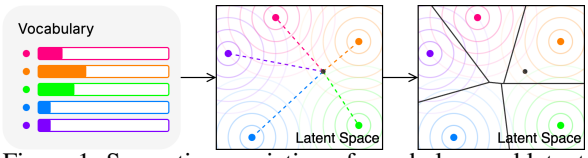


Figure 1: Semantic association of vocabulary and latent space. For each color label on the vocabulary (left), there is a color semantic basis in the latent space (middle). The semantics of the dark dot (indicating an arbitrary representation) in the latent space can be quantified as its cosine similarities to semantic bases. The semantics can be computed as probabilities on the vocabulary. When focusing on the nearest semantic basis for a given latent representation, a latent space can be quantified as discrete semantic regions (right).

For a given LM-head matrix, we conduct matrix multiplication to obtain semantic bases in the latent space. Since the computation direction is from logits to representations, instead of using the LM-head matrix $\mathbb{W}$, we use its pseudoinverse $\mathbb{W}^+$. If there are v labels in the vocabulary, there will be v unique semantic bases representing all semantic meanings. At the output side of LM, we multiply each onehot embedding $\vec{e}$ by the pseudoinverse matrix $\mathbb{W}^+$ to obtain the corresponding representation $\vec{s}$. That is, $\vec{s} = \vec{e} \cdot \mathbb{W}^+$. The computation is equivalent to solving the least-squares problem of a system of linear equations. Equivalently, the LM-head and its pseudoinverse form a readout/reconstruction pair:

the LM-head maps a hidden state to semantic coordinates on the vocabulary, while the pseudoinverse maps those coordinates back to a latent representation in the least-squares sense. The time cost of computing semantic bases is rather low. For language models like LLaMA 2 (7B, 13B, and even 70B), which have 32k labels in the vocabulary, it takes around 10 seconds on an A100 GPU. Moreover, this is a one-time computation with persistent value.

3.2 Empirical Finding: Vector Nature of Semantics

Centered on each semantic basis, there forms a "semantic field". The concept of semantic field is similar to the *field* term in physics (such as electric field, with the semantic basis analogous to the electric pole). The semantics of an arbitrary latent representation can be quantified as the overlapping impact of multiple semantic fields, and further computed as probabilities (Gu et al., 2024). The process is "composition of semantics", where multiple *semantic components* become a *resultant vector* via vector addition. We propose a hypothesis that the overlapping effects of semantic fields support a corresponding reversed operation, namely "resolution of semantics". That is, a single *resultant vector* in latent space may be resolved into multiple *component vectors* along the directions of semantic bases.

In detail, for a given latent representation $\vec{r}$, its semantic meaning can be projected to different semantic bases to obtain corresponding semantic components $\vec{c_i} = \texttt{proj}(\vec{r}, \vec{s_i})$ (analogy to "component force" in a force field). By accumulating the decomposed semantics, we get a "resultant semantics" $\sum_{i=1}^{n} \vec{c_i}$ (analogy to "resultant force" in a force field). The equation $\vec{r} \parallel \sum_{i=1}^{n} \vec{c_i}$ stands approximately true. In contrast, when taking a random collection of vectors as semantic bases and obtaining $\vec{c_i'} = \texttt{proj}(\vec{r}, \vec{s_i'})$, the equation $\vec{r} \perp \sum_{i=1}^{n} \vec{c_i'}$ stays true. It is consistent with the property of high-dimensional latent space that arbitrary vectors tend to be orthogonal to each other.

We conduct empirical experiments to validate the hypothesis. For a given dataset and LM, we first compute the outputs of each layer, and then decompose each layer output into semantic components and eventually recompose it back as a layer output. The question tested in this experiment is simple: does decomposition followed by recomposition recover a vector that keeps nearly the same semantic direction as the original hidden state? If the old layer outputs and the new layer outputs share almost the same direction in the latent space, namely their cosine similarity is high, the hypothesis stands. We run with Llama2-7B on HumanEval, to study whether the hypothesis stands with the output-side semantic bases, while using input-side semantic bases for comparison. As shown in Figure 2, the hypothesis stands with the case of using output-side semantic bases because of the very high cosine similarities, no matter the layer. In contrast, the situation of using input-side semantic bases is poor and the cosine similarities are close to zero, which reflects the common phenomenon in high-dimensional latent spaces that arbitrary vectors tend to be orthogonal to each other.

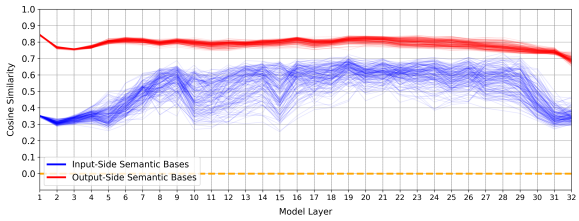


Figure 2: Empirical validation of semantic decomposition on HumanEval with Llama2-7B. The figure tests whether decomposition and recomposition preserve the direction of the original hidden state.

4 Approach

Our approach uses LM semantics to align the latent space between paired layers of the source and target models. The transferred knowledge under semantic alignment is the layer output, which serves as the supervisory signal for parameter optimization. We name our approach Semantic Alignment, shortened as SEMALIGN. The illustration of our approach is in Figure 3, and the main procedure is as follows: (1) first, we locate critical layers in the source LM by attribution and select exactly one source layer for each target layer; these selected source layers are then paired with all target layers from shallow to deep; (2) then, we decompose the semantics of source layer outputs in the source latent space and recompose them as supervisory layer outputs in the target latent space; (3) finally, in the target model, we optimize the paired target layers stage by stage so that, on the same inputs, their outputs approach the aligned supervisory states. In this way, the target model gradually reproduces the source behavior

through aligned activations rather than transferred source parameters.

4.1 Layer Attribution and Layer Pairing

Layer Attribution. We identify candidate source layers using *Layer Gradient* $\times$ *Activation* at the layer level: for each layer, we multiply the gradient of the supervised objective by the layer's activations and aggregate over tokens and channels to obtain a scalar importance score per layer. This choice is simple, stable, and available in standard attribution toolkits (Kokhlikyan et al., 2020).

Layer Pairing. Let the source have L_{src} layers and the target have L_{tgt} layers. In the refined SemAlign workflow, the extract stage selects exactly L_{tgt} critical source layers from the source LM only. Let their sorted indices be

$$\mathcal{S}_{\text{src}} = \{s_1 < s_2 < \cdots < s_{L_{\text{tgt}}}\}.$$

We then pair them directly with all target layers from shallow to deep:

$$(s_k,\, k), \qquad k \in \{1, \ldots, L_{\text{tgt}}\}.$$

This rule removes the need to locate critical target layers and gives a simple one-to-one correspondence for the subsequent semantic-alignment stage. In other words, attribution decides *which* source layers to transfer, and ordering decides *where* they supervise in the target.

4.2 Latent Semantic Alignment (from Source LM to Target LM)

We construct a training-free, semantics-preserving mapping from a *source* layer's latent space to a *target* layer's latent space. In this paper, the source and target come from the same LM family at different scales, so their vocabularies and output-side semantic bases are index-aligned (Section 3). The idea is to *decompose* a source vector into semantic components and *recompose* those components in the target basis with the same coefficients.

Let $\mathbf{h}^{\text{src}} \in \mathbb{R}^{D_{\text{src}}}$ be a source-layer output at the supervision interface. Let the source and target semantic bases be

$$\begin{aligned} S_{\text{src}} &= [\mathbf{s}_1^{\text{src}}, \ldots, \mathbf{s}_m^{\text{src}}] \in \mathbb{R}^{D_{\text{src}} \times m}, \\ S_{\text{tgt}} &= [\mathbf{s}_1^{\text{tgt}}, \ldots, \mathbf{s}_m^{\text{tgt}}] \in \mathbb{R}^{D_{\text{tgt}} \times m}, \end{aligned} \tag{1}$$

where column i in S_{src} and S_{tgt} refers to the same semantic atom. The semantic bases themselves need not be unit-normalized; in the practical implementation we keep the raw output-side bases induced by the LM-head and its pseudoinverse.

Semantics-preserving decomposition and recomposition. For conceptual exposition, we write the semantic coefficients of the source representation in cosine style:

$$a_i = \cos(\mathbf{h}^{\text{src}}, \mathbf{s}_i^{\text{src}}), \qquad i \in \{1, \ldots, m\}. \tag{2}$$

Collecting them gives $\mathbf{a} = [a_1, \ldots, a_m]^\top \in \mathbb{R}^m$. We then recompose these coefficients in the target space:

$$\tilde{\mathbf{h}}^{\text{tgt}} = S_{\text{tgt}}\, \mathbf{a} \in \mathbb{R}^{D_{\text{tgt}}}. \tag{3}$$

The vector $\tilde{\mathbf{h}}^{\text{tgt}}$ serves as the supervisory representation for the paired target layer. Because the same coefficient vector $\mathbf{a}$ is used on index-aligned semantic bases, the semantic content is preserved while only the ambient basis changes. In implementation, this decomposition and recomposition view is realized with the raw, non-normalized output-side semantic bases, rather than by explicitly normalizing the basis vectors.

In the practical implementation, we may additionally apply the frozen LM-head as a semantic readout at any layer. This step is optional: the proposed method still operates on representations, but the readout provides a convenient connection to common logit-space optimization. Under a shared vocabulary, the LM-head converts intermediate representations into target-space semantic logits that are directly comparable across the paired source and target layers.

4.3 Representation and Output Alignment

In the second stage, we adapt the target with a shallow-to-deep frontier schedule. At frontier stage k, only the current target layer k is trainable, while all other target layers are frozen. A direct hidden-state cosine loss remains a valid realization, but the default implementation applies the frozen target LM-head to intermediate states and optimizes a Fisher-weighted quadratic surrogate on the resulting target-space logits. This keeps the method representation-centered while moving the layer objective closer to the local softmax geometry of standard logit-space training, while avoiding simultaneous interference from many layer losses.

Let $\mathbf{h}^{(k)} \in \mathbb{R}^{B \times T \times D}$ be the target activations at the current supervision interface, chosen as the sublayer output *before* residual addition. Let $\tilde{\mathbf{h}}^{(k)} \in \mathbb{R}^{B \times T \times D}$ denote the aligned target-side

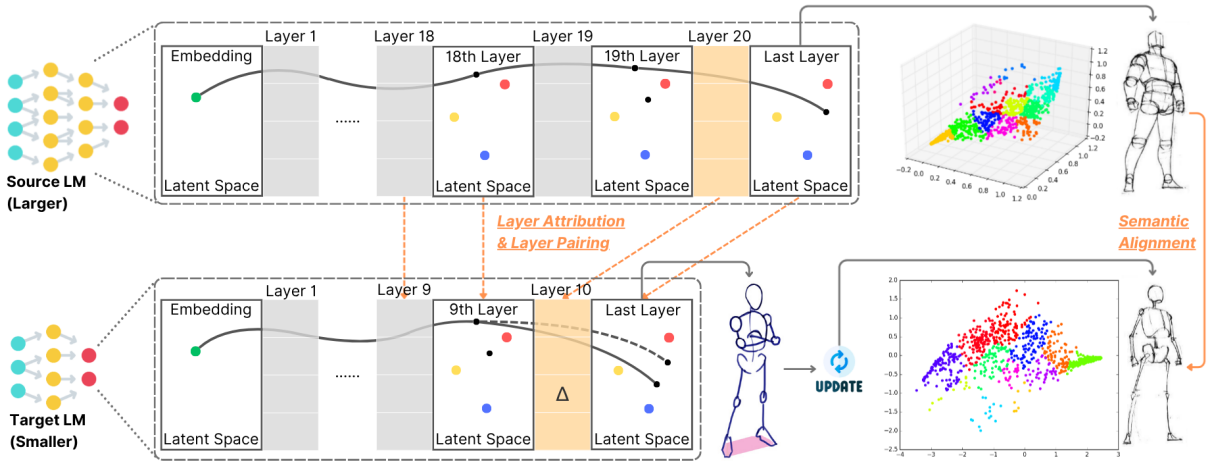


Figure 3: Illustration of our cross-scale knowledge transfer approach. Assume a 20-layer source LM and a 10-layer target LM, then: (1) the source-side extract stage selects critical source layers, and the selected source layers are paired one-to-one with all target layers from shallow to deep. A highlighted pair is shown in orange only for illustration; (2) represented by the dots in 3D and 2D spaces, the layer outputs from the source model are decomposed in the larger-dimensional source latent space and recomposed in the smaller-dimensional target latent space as the supervisory signal. It undergoes dimensional reduction but still preserves the semantics, represented by the changes to gray bots, which keep the body gesture while reducing details; (3) the current frontier target layer is updated to make its outputs close to the supervisory signal, while the other target layers remain frozen. It is similar to blue bots being adjusted to play the same body gesture as gray bots. After the cross-scale knowledge transfer, target layer outputs steer to the supervisory signal, represented by the dashed curve, and only a small set of layer parameters are optimized at each stage, marked by the delta symbol. In the last layer's latent space, the output is represented by the small dark dot whose distances to big color dots indicate the probabilities in the LM vocabulary. For the source LM, its dark dot is close to the red dot, so its output label is the corresponding red label; while for the target LM, its output label was the corresponding blue but becomes red after knowledge transfer, since its dark dot moves far away from the blue dot while approaching the red dot.

supervisory representation constructed from the paired source layer. Applying the frozen target LM-head to these intermediate states gives

$$\mathbf{q}^{(k)} = W_{\mathrm{U}}^{\mathrm{tgt}}\mathbf{h}^{(k)}, \qquad \tilde{\mathbf{q}}^{(k)} = W_{\mathrm{U}}^{\mathrm{tgt}}\tilde{\mathbf{h}}^{(k)}.$$

Let $\tilde{\mathbf{p}}^{(k)} = \mathrm{softmax}(\tilde{\mathbf{q}}^{(k)}/\tau)$ and $\boldsymbol{\delta}^{(k)} = (\mathbf{q}^{(k)} - \tilde{\mathbf{q}}^{(k)})/\tau$. We first define

$$\begin{aligned} a^{(k)} &= \sum_i \tilde{p}_i^{(k)} (\delta_i^{(k)})^2, \\ b^{(k)} &= \sum_i \tilde{p}_i^{(k)} \delta_i^{(k)}. \end{aligned} \tag{4}$$

The layer-level loss at the current frontier stage is then

$$\mathcal{L}_{\mathrm{layer}}^{(k)} = \mathrm{Avg}\left[\tfrac{1}{2}\left(a^{(k)} - (b^{(k)})^2\right)\right]. \tag{5}$$

This is the Fisher-weighted quadratic surrogate on target-space aligned logits, namely the second-order approximation of the KL divergence under the softmax geometry.

At the final layer, let

$$\mathbf{p}^{\mathrm{src}} = \mathrm{softmax}(\mathbf{z}^{\mathrm{src}}), \qquad \mathbf{p}^{\mathrm{tgt}} = \mathrm{softmax}(\mathbf{z}^{\mathrm{tgt}}).$$

We optimize

$$\mathcal{L}_{\mathrm{out}} = \mathrm{Avg}\left[\mathrm{KL}\left(\mathbf{p}^{\mathrm{src}} \,\|\, \mathbf{p}^{\mathrm{tgt}}\right)\right]. \tag{6}$$

At frontier stage k, the total objective is the weighted sum

$$\mathcal{L}^{(k)} = \lambda_{\mathrm{layer}}\mathcal{L}_{\mathrm{layer}}^{(k)} + \lambda_{\mathrm{out}}\mathcal{L}_{\mathrm{out}}. \tag{7}$$

Here $\mathrm{Avg}[\cdot]$ denotes averaging over the supervised positions of the active frontier layer. After the current frontier stage is optimized, we freeze layer k and move to layer $k{+}1$, so the semantic-alignment stage proceeds from shallow to deep.

The layer term is still a representation-alignment objective. The LM-head is only a frozen semantic readout: two representations are aligned by comparing the target-space semantic logits they induce under the same vocabulary. This differs from standard intermediate-layer distillation, which typically matches source features or source logits directly in the source space. Here we first transport the supervision into the target semantic space, and the LM-head only makes that target-side supervision

easier to optimize. The Fisher-weighted quadratic form keeps the correct softmax geometry while being less sensitive than exact per-layer KL to noisy low-probability tails. The output term has the usual distillation interpretation, and when the source distribution degenerates to a one-hot label, minimizing the KL term between the source and target distributions is equivalent to minimizing cross-entropy up to a source-only entropy constant. In this sense, the optional LM-head readout better connects SemAlign to common optimization on logits, even though the transferred object remains the aligned representation.

5 Experiments

In the experiments, we mainly study how our approach performs in parametric knowledge transfer compared with PKT baselines. We also conduct analysis based on the latent representation similarities between source and target models before and after latent semantic alignment.

5.1 Setup

Datasets. We use four well-established benchmarks, covering the most common downstream tasks: MMLU measures professional knowledge (Hendrycks et al., 2021); GSM8K measures mathematical reasoning (Cobbe et al., 2021); HumanEval and MBPP measure code generation (Chen et al., 2021; Austin et al., 2021).

Models. We conduct experiments with Llama-2 (Touvron et al., 2023) models, mainly chat versions instead of base versions for better instruction-following ability. Besides, we employ LM variants to study transfer from further-finetuned source models to the same target model, CodeLlama-13B-Python (Roziere et al., 2023) and WizardCoder-13B-Python (Luo et al., 2023). They are finetuned on Llama-2-13B with massive code data for enhanced coding performance.

Metrics. For MMLU and GSM8K, we calculate *accuracy* in the zero-shot setting; and for HumanEval and MBPP, we calculate *pass@1*. Larger scores mean better performance.

Baselines. We compare against the two representative PKT baselines, SEEKING and LATEN. SEEKING follows a PostPKT pipeline with extraction, injection, and post-alignment finetuning. LATEN follows a PrePKT pipeline with localization, pre-alignment, and injection. In contrast, SemAlign uses a two-stage workflow: source-side extract followed by a merged semantic-alignment stage with shallow-to-deep frontier optimization. More details are given in Appendix B.

5.2 Results

Cross-Scale Knowledge Transfer. In all four benchmarks, SEMALIGN improves substantially over Llama2-7B-Chat while remaining below the Llama2-13B-Chat source, and it stays closer to the source than the other transfer baselines on average. Concretely, the absolute gaps between SemAlign and 13B are 2.60 on MMLU (50.30 vs 52.90), 1.34 on GSM8K (19.21 vs 20.55), 1.41 on HumanEval (17.34 vs 18.75), and 0.42 on MBPP (18.78 vs 19.20), averaging 1.44 points. This average gap is smaller than SEEKING ($\approx$ 3.92) and LATEN ($\approx$ 3.43). At the task level, LaTen is numerically closest to the source on GSM8K (20.47 vs 20.55), while SEEKING overshoots the source on the same task (28.23). Overall, SemAlign's three-of-four closer margins indicate that it learns source behavior more faithfully than the baselines.

Models	MMLU	GSM8K	HumanEval	MBPP
Llama2-7B-Chat	44.20	16.07	14.05	17.80
Llama2-13B-Chat	52.90	20.55	18.75	19.20
Seeking	49.60	**28.23**	15.44	**20.60**
LaTen	44.40	20.47	14.63	18.20
SemAlign	**50.30**	19.21	**17.34**	18.78

Table 1: Results of Parametric Knowledge Transfer in Diverse Downstream Tasks.

On task leadership, SemAlign attains the best transferred performance on MMLU (50.30) and HumanEval (17.34), surpassing both Seeking (49.60, 15.44) and LATEN (44.40, 14.63), whereas SEEKING leads on GSM8K (28.23) and MBPP (20.60). This result pattern is also consistent with the design of the methods. In SemAlign, the target is optimized with supervisory hidden states constructed from the source model, together with the source output distribution. The objective is therefore to reproduce source behavior as faithfully as possible, rather than to perform an additional stage of task-specific finetuning. By contrast, SEEKING includes a post-alignment finetuning stage after injection, which naturally gives it more room to overshoot the source on some tasks. For the same reason, it is difficult for a transferred target produced by SemAlign to consistently outperform the source model itself.

A clear pattern is the trade-off between stability and aggressiveness across methods. SEEKING exceeds the 13B source on both GSM8K (+7.68) and MBPP (+1.40), suggesting stronger task-specific optimization. SemAlign remains below but close to the source, and stays within 2.60 points of the source on every task. A second plausible factor is that the current implementation uses model-wise output-side semantic bases as a practical approximation to layer-wise semantics; this approximation is robust overall, but it may be less precise on tasks that require especially sharp token-level distinctions.

Knowledge Transfer from Finetuned Models. In five of six source-task settings, SEMALIGN consistently outperforms the transfer baselines, indicating stronger parametric knowledge transfer. With Llama2-13B-Chat as source, it leads LATEN and SEEKING on HumanEval (17.34 vs 14.63 and 15.44), and only trails SEEKING on MBPP (18.78 vs 20.60). The advantage becomes clearer with code-specialized sources: from CodeLlama-13B-Python, SemAlign reaches 20.12 (+4.07 over SEEKING, +6.10 over LATEN) on HumanEval and 22.35 (+0.95, +4.55) on MBPP; from WizardCoder-13B-Python, it attains 19.46 (+4.42, +5.44) and 21.18 (+1.38, +2.58) on HumanEval and MBPP, respectively. These trends show that SemAlign extracts and transfers source competency more reliably, especially when the source is stronger in coding.

Models	HumanEval	MBPP
Llama2-7B-Chat	14.05	17.80
Llama2-13B-Chat	18.75	19.20
Seeking	15.44	**20.60**
LaTen	14.63	18.20
SemAlign	**17.34**	18.78
CodeLlama-13B-Python	47.56	37.80
Seeking	16.05	21.40
LaTen	14.02	17.80
SemAlign	**20.12**	**22.35**
WizardCoder-13B-Python	56.71	41.60
Seeking	15.04	19.80
LaTen	14.02	18.60
SemAlign	**19.46**	**21.18**

Table 2: Results of Parametric Knowledge Transfer from Finetuned Source Models.

Across the two coding benchmarks, all methods remain far below the finetuned sources (CodeLlama-13B-Python: 47.56 on HumanEval and 37.80 on MBPP; WizardCoder-13B-Python: 56.71 and 41.60), despite often surpassing Llama2-7B-Chat and sometimes even matching or exceeding Llama2-13B-Chat. This gap suggests that the extensive, task-specific optimization baked into code-specialized sources is difficult to reconstruct via short-horizon transfer; semantic-coordinate matching encourages conservative alignment to source representations rather than aggressive task re-optimization, limiting the attainable ceiling without longer or more targeted finetuning.

An additional observation is that SEEKING only overshoots the source on MBPP when the source is the generalist Llama2-13B-Chat (20.60 > 19.20), but not when the source is code-specialized. Meanwhile, SEMALIGN shows its largest margins over baselines precisely when transferring from code-specialized sources. This pattern hints that aggressive, task-specific optimization in SEEKING can exploit headroom left by generalist sources, whereas SemAlign's representation-faithful transfer scales better with source specialization.

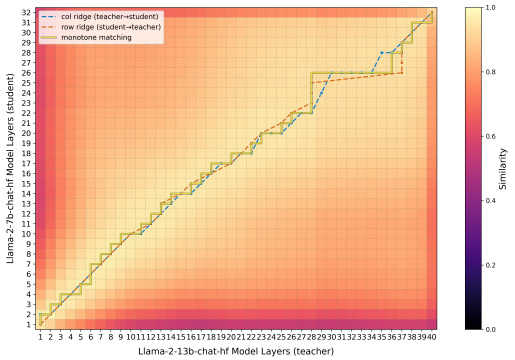
5.3 Analysis

We adopt Centered Kernel Alignment (CKA) (Kornblith et al., 2019) as the analysis tool to study the similarities between layer outputs from source and target models. We run Llama2-chat models on HumanEval data. CKA is commonly used to compute the similarities between feature representations in neural networks, which is based on Hilbert-Schmidt Independence Criterion (HSIC).

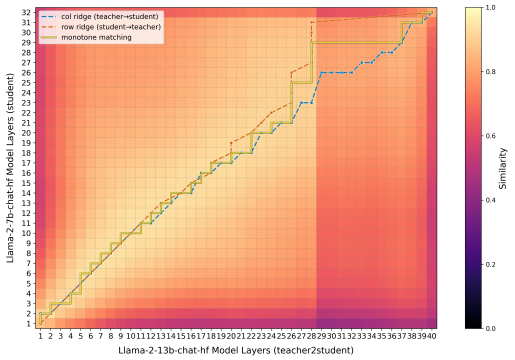
As shown in Figure 4, there are high similarities between the layer outputs from source and target models, especially along the main diagonal. This indicates that there is no strong behavioral incompatibility when layer outputs are used as the medium of parametric knowledge transfer, instead of directly using layer parameters. The highest similarities are almost layer by layer, from shallow to deep. Meanwhile, the cases before and after latent semantic alignment share a very similar pattern of similarities. This suggests that latent semantic alignment is a safe way to utilize the cross-scale similarities between layer outputs.

6 Conclusion

We studied parametric knowledge transfer across differently scaled LLMs from a *semantics-first* perspective. Rather than moving raw parameters as in prior paradigms, we use layer outputs as the medium of transfer and identify *latent semantic*



(a) source-target without alignment



(b) source-target with alignment

Figure 4: Comparison of Layer-wise Representation Similarities between LLMs.

alignment as the prerequisite for stable cross-scale transfer. Building on this view, SEMALIGN first extracts a source-only set of critical layers and then performs a merged semantic-alignment stage that pairs them with all target layers from shallow to deep, aligns their latent semantics, and steers the target stage by stage with source-side supervisory states. In the realization, the semantic-alignment stage follows a shallow-to-deep frontier schedule in which only the current target layer is trainable and the final output loss remains active throughout. This design avoids neural incompatibility, simplifies the procedure, and makes transfer efficient in both compute and data. Empirically, SemAlign improves targets over task-matched baselines and over parameter-space transfer methods on four benchmarks with Llama 2 families. The results support our central claim: treating activations as the carrier of knowledge and aligning semantics first provides a robust path to cross-scale PKT.

Limitations

Our evaluation is limited in scope. The experiments are on four benchmarks and mainly study transfer within the Llama-2 family across scales, with a few further-finetuned variants. It is still not clear how well SemAlign works for more diverse settings, such as other model families with different tokenizers or architectures, longer-context use, multilingual tasks, or safety-critical behavior.

SemAlign further assumes that the source and target share a usable semantic correspondence, so that we can decompose a representation in the source and recompose it in the target. When this correspondence is weak, such as under vocabulary mismatch or semantic drift, the alignment signal can become noisy and transfer may degrade. Extending the method to cross-family transfer will likely require additional transformations on semantic bases and explicit treatment of vocabulary mismatch, which we leave to future work. The current implementation also uses model-wise output-side semantic bases as a practical approximation to layer-wise semantics, and this approximation may be less precise on tasks that require especially sharp token-level distinctions.

More broadly, our method currently operates at the layer level and uses lightweight updates, which keep the two-stage procedure simple and stable but can cap the best achievable performance and do not fully prevent interference with other skills. The current frontier schedule reduces cross-layer interference during training, but it also makes the transfer procedure more conservative than full joint optimization. The optional LM-head readout also introduces extra vocabulary-sized computation compared with direct hidden-state matching, even though it gives a cleaner connection to standard cross-entropy and KL optimization.

References

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Xingyu Cai, Jiaji Huang, Yu-Lan Bian, and Kenneth Ward Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large

language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yuheng Chen, Pengfei Cao, Yubo Chen, Yining Wang, Shengping Liu, Kang Liu, and Jun Zhao. 2025. Cracking factual knowledge: A comprehensive analysis of degenerate knowledge neurons in large language models. In *Proceedings of ACL 2025*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. In *Proceedings of EMNLP 2024*.

Jian Gu, Aldeida Aleti, Chunyang Chen, and Hongyu Zhang. 2024. Vocabulary-defined semantics: Latent space clustering for improving in-context learning. *arXiv preprint arXiv:2401.16184*.

Jian Gu, Aldeida Aleti, Chunyang Chen, and Hongyu Zhang. 2025. A semantic-aware layer-freezing approach to computation-efficient fine-tuning of language models. In *Findings of the Association for Computational Linguistics: ACL 2025*.

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*.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch. *Preprint*, arXiv:2009.07896.

Lingkai Kong and 1 others. 2024. Aligning large language models with representation editing. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Ziyue Liu and 1 others. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *arXiv preprint arXiv:2408.07666*.

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. *arXiv preprint arXiv:2306.08568*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Yuqiao Tan, Shizhu He, Kang Liu, and Jun Zhao. 2025. Neural incompatibility: The unbridgeable gap of cross-scale parametric knowledge transfer in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 21586–21601.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Transformers: State-of-the-art natural language processing. In *Conference on Empirical Methods in Natural Language Processing*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Chuanpeng Yang, Wang Lu, Yao Zhu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *arXiv preprint arXiv:2407.01885*.

Enneng Yang and 1 others. 2025. A review of model merging approaches. *arXiv preprint arXiv:2503.08998*.

Zeping Yu and Sophia Ananiadou. 2024. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. 2024. Seeking neural nuggets:

Knowledge transfer in large language models from a parametric perspective. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Implementation Details

A.1 Stats of Language Models

The stats of language models in our experiments are shown in Table 3.

	Llama2		CodeLlama	WizardCoder
	7B	13B	13B	13B
Head Num.	32	40	40	40
Layer Num.	32	40	40	40
Dimension	4,096	5,120	5,120	5,120
Vocabulary	32,000	32,000	32,000	32,001

Table 3: Stats of Llama-2 Language Models.

A.2 Implementation Details

We follow the experimental protocol of LATEN for a fair comparison, but our refined SemAlign workflow has two stages only. The *extract* stage uses the extract split to attribute source layers on the source LM only, and it selects exactly L_{tgt} source layers for a target with L_{tgt} layers. The *semantic-alignment* stage then pairs the selected source layers with all target layers from shallow to deep and optimizes the target on the align split through a frontier schedule. At frontier stage k, only target layer k is trainable and all other target layers are frozen.

In the default implementation, we construct the aligned target-side representation in Section 4, project both the aligned supervisory states and the target hidden states through the frozen target LM-head, and optimize the resulting target-space logits with the Fisher-weighted quadratic surrogate in Section 4.3. At every frontier stage, we also keep the KL loss between the source and target final logits. The two terms are combined as a weighted sum, with λ_{layer}=0.25 and λ_{out}=1.0 in our current implementation. We update only LoRA parameters inserted into FFN (`up_proj`, `down_proj`) and MHSA (`v_proj`, `o_proj`) modules, with rank r=16, and only for the active frontier layer. Both terms are computed on the standard shifted next-token supervision positions and averaged over valid tokens. For the optimization schedule of the semantic-alignment stage, we mainly adopt the hyperparameters of SEEKING's finetuning stage. The experiments are conducted via a single run, with the global random seed 42.

Our implementation uses deep learning framework PYTORCH (Paszke et al., 2019), TRANSFORMERS (Wolf et al., 2019), and vLLM-backed LM-EVAL (Kwon et al., 2023).

	Epochs	Batch Size	Learning Rate
MMLU	5	16	3e-4
GSM8K	5	16	3e-4
HumanEval	3	16	3e-5
MBPP	5	16	3e-4

Table 4: Semantic-alignment optimization settings.

A.3 AI Use Disclosure

We use a business LLM (GPT-5 [2]) to merely aid or polish our paper writing.

B Details of Parametric Knowledge Transfer Baselines

This appendix summarizes the two PKT baselines that are most relevant to our comparison. Both methods transfer parameter-side information from a larger source model to a smaller target model, but they differ in where alignment happens. In the terminology of Tan et al. (2025), SEEKING is a representative *PostPKT* method, while LATEN is a representative *PrePKT* method. The most relevant cross-scale comparisons in these works are reported on the same four benchmarks used in our paper, namely MMLU, GSM8K, HumanEval, and MBPP.

B.1 SEEKING

SEEKING transfers knowledge through a three-step PostPKT pipeline: *extract*, *inject*, and *post-align* (Zhong et al., 2024). For a task $\mathcal{T}$ and a small seed set decoded by the source model, it scores each source parameter θ_i by sensitivity,

$$S_{i,j}^{\mathcal{T}} = \left| \theta_i^{\top} \nabla_{\theta_i} \mathcal{L}\left(x_j^{\mathcal{T}}, y_j^{\mathcal{T}} \mid \Theta\right) \right|,$$

$$S_i^{\mathcal{T}} = \sum_{j=1}^{k} S_{i,j}^{\mathcal{T}},$$

and then aggregates these scores within each layer. The selected source weights are reduced to target-compatible submatrices,

$$W_{\text{extract}}^{l} = \arg \max_{W' \subseteq W^{l}} \sum_{\theta_i \in W'} S_i,$$

[2] https://openai.com/gpt-5/

then factorized with SVD to initialize a rank-r LoRA pair for the target. After injection, the target is further optimized in a post-alignment finetuning stage.

The important point for our comparison is that SEEKING does not stop at transfer. Its post-alignment stage can provide stronger task-specific optimization than a source-faithful transfer objective, which helps explain why it sometimes overshoots the source model on individual tasks. In the broader evaluation of Zhong et al. (2024), the method is tested on several task families; the cross-scale benchmarks most directly comparable to our setting are MMLU, GSM8K, HumanEval, and MBPP.

B.2 LATEN

LATEN is proposed in the neural incompatibility study of Tan et al. (2025) as a PrePKT method. It follows three steps: *locate*, *align*, and *inject*. Given a source model M_ℓ with parameters Θ_ℓ and a target model M_s with parameters Θ_s, it first localizes informative neurons and extracts source deltas, then maps them into the target parameter space with a lightweight hypernetwork. The core pipeline can be written as three short updates:

$$\Delta\Theta_\ell \leftarrow \mathrm{Locate}(M_\ell; \mathcal{D}_{\mathrm{extract}})$$

$$\widehat{\Delta\Theta}_s \leftarrow g_\phi(\Delta\Theta_\ell)$$

$$\Theta_s^\star \leftarrow \mathrm{Inject}(\Theta_s, \widehat{\Delta\Theta}_s)$$

The locate stage uses static neuron-level attribution over FFN and MHSA modules. The align stage trains a small two-layer MLP hypernetwork on a tiny alignment set while the base weights remain frozen. After that, the aligned deltas are injected once, with no following finetuning stage.

The key conclusion of Tan et al. (2025) is that both PostPKT and PrePKT remain unstable across scales because of neural incompatibility. Their experiments on MMLU, GSM8K, HumanEval, and MBPP show that pre-alignment alone reduces some cost, but does not fully resolve the mismatch between source and target parameter spaces. This is the most direct motivation for our semantics-first alternative: we also treat alignment as the prerequisite, but we align in latent semantic space rather than in parameter space.

C Discussion

C.1 Activation-Level Transfer as the Core Design

Compared with prior PKT methods such as SEEKING and LATEN, which transfer source-side parameter information, SEMALIGN uses layer outputs as the transfer medium. This choice matches the central design in the main body: the target is optimized with supervisory hidden states constructed from the source, rather than with transplanted source parameters.

From the perspective of information transfer, layer outputs are a compact carrier. In language models, the dimensionality of a layer output is much smaller than that of the full layer parameters, and it is also much smaller than the full vocabulary-sized probability vector. This makes activation-level transfer a natural middle ground: richer than text-only supervision, but more structured and lighter than direct parameter manipulation.

From the perspective of knowledge localization, layer outputs are also a safer object to move across models. Prior work suggests that knowledge and neurons are related in a many-to-many and context-dependent way (Allen-Zhu and Li). Direct parameter manipulation can therefore introduce side effects, because the same parameters participate in many behaviors beyond the current task. By transferring aligned activations instead, SemAlign focuses on the computation induced by the current input rather than reusing source parameters directly.

This view also helps explain the result pattern in Section 5. Since the target is trained to follow supervisory hidden states and output distributions derived from the source, the method is naturally source-faithful and conservative. It narrows the gap to the source, but it is not designed to outperform a stronger source model or to match methods that add a separate finetuning stage after transfer.

C.2 Semantic Readout and Frontier Optimization

The default implementation introduces an additional readout step through the frozen LM-head, but this does not change the transfer medium from representations to logits. The transferred object is still the aligned target-side representation $\tilde{\mathbf{h}}^{\mathrm{tgt}}$ in Section 4; the LM-head only turns that representation into target-space semantic logits that are easier to optimize with standard logit-space ob-

jectives. The shallow-to-deep frontier schedule complements this design by avoiding simultaneous optimization of many layer losses. At each stage, the current frontier layer receives the local alignment signal, while the final KL term keeps the full model tied to source behavior.

This view provides a clean connection between the layer-level Fisher-weighted quadratic surrogate and the final-layer KL objective. For any source distribution $\mathbf{p}^{\mathrm{src}}$ and target distribution $\mathbf{p}^{\mathrm{tgt}}$, we have

$$\mathrm{KL}\big(\mathbf{p}^{\mathrm{src}} \,\|\, \mathbf{p}^{\mathrm{tgt}}\big) = \mathrm{CE}\big(\mathbf{p}^{\mathrm{src}}, \mathbf{p}^{\mathrm{tgt}}\big) - H(\mathbf{p}^{\mathrm{src}}),$$

so minimizing KL is equivalent to minimizing cross-entropy up to a source-only constant. At the output layer, this is the usual distillation identity. At an intermediate layer, instead of matching the induced distributions with exact KL, we use the Fisher-quadratic form induced by the aligned reference distribution $\tilde{\mathbf{p}}$. Writing

$$a = \sum_i \tilde{p}_i \, \delta_i^2,$$
$$b = \sum_i \tilde{p}_i \, \delta_i,$$
$$\delta_i = \frac{q_i - \tilde{q}_i}{\tau},$$

the layer objective becomes

$$\tfrac{1}{2}\big(a - b^2\big).$$

This is the second-order approximation of KL on logits and preserves the correct shift-invariant softmax geometry. Therefore, the layer loss can still be read as a logit-space optimization on intermediate representations after a fixed semantic readout, but with a locally quadratic surrogate that is less sensitive to noisy low-probability tails.

The LM-head thus serves two roles that connect directly to the main method. First, it provides a common semantic coordinate system that makes source-target comparison easy to implement after semantic alignment. Second, unlike a cosine-only loss, it retains coefficient magnitude and therefore stays closer to the optimization geometry of ordinary logit-space training. Adding the LM-head also introduces vocabulary-sized readout costs compared with pure hidden-state matching, but in our setting that cost remains manageable because alignment is applied only to one frontier layer at a time and on a small alignment set. In return, the optimization becomes easier to analyze and easier to relate to the final-layer KL objective used in the main training stage.