PRiSM: Partial Ranking via inter-layer Semantic Measurement for Efficient Fine-tuning of Language Models

Aldrin Kabya Biswas, Md Fahim, Md. Ashraful Amin, Amin Ahsan Ali, AKM Mahbubur Rahman
Center for Computational & Data Sciences, Independent University, Bangladesh
Dhaka 1245, Bangladesh
{aldrin.kabya04, fahimcse381}@gmail.com,
{aminmdashraful, aminali, akmmrahman}@iub.edu.bd

Abstract

The growing scale of pre-trained language models poses a challenge in fine-tuning for downstream tasks, especially in resource-constrained settings. Recent studies highlight that not all layers in transformer-based language models contribute equally to downstream task performance, giving rise to various partial fine-tuning strategies. However, current methods often introduce significant training overhead or rely on simple heuristics that yield suboptimal performance and poor generalization. We propose PRiSM (**P**artial **R**anking via **i**nter-layer **S**emantic **M**easurement), a training-free approach for layer-wise partial fine-tuning that leverages the cosine similarity between pre-trained aggregate token representations across layers to identify inter-layer relationships. PRiSM comprises two stages: (i) scoring layers based on their relevance to the task via a single forward pass, and (ii) fine-tuning a subset of block-wise highest-scoring layers, while keeping others frozen. We conduct experiments on 15 diverse NLP datasets, including single-sentence and sentence-pair classification tasks. Our method achieves competitive performance compared to full fine-tuning, with an average training speedup of 1.5× and a reduction of trainable parameters by 75%, and outperforms all the comparative baselines. Additionally, our approach does not cause any notable drop in performance when the domain is changed for the evaluation tasks, demonstrating robust cross-domain generalizability.

Keywords: Layer Selection, Partial Fine-tuning, Parameter-Efficient Fine-tuning

1. Introduction

The remarkable success of large-scale Language Models (LMs) like BERT (Devlin et al., 2019) stems from their ability to learn rich, hierarchical representations of language (Arora et al., 2020; Niu et al., 2022). These models have consistently set new state-of-the-art benchmarks across a wide array of NLP tasks (Choi et al., 2021). However, their growing size presents a significant challenge: the computational cost of full fine-tuning is often prohibitive, especially in resource-constrained settings. This has spurred research into more efficient adaptation methods, grounded in the observation that not all layers contribute equally to performance on a given downstream task (Hosseini et al., 2023; Charpentier and Samuel, 2023). Consequently, various parameter-efficient and partial fine-tuning strategies have emerged, promising to match the performance of full fine-tuning with a fraction of the computational budget (Ardakani et al., 2024; Zhu et al., 2023; He et al., 2024; Shen et al., 2021).

Current approaches to partial fine-tuning largely fall into two categories, each with distinct drawbacks. The first category, dynamic layer selection, employs methods like SlimFit (Ardakani et al., 2024) and RECAP (Ilhan et al., 2024), which identify important layers during training based on iterative gradient analysis or auxiliary mechanisms. While effective, these methods introduce significant computational overhead, as importance scores must be calculated and updated for each sample in every epoch, undermining the goal of efficiency. The second category relies on static heuristics, such as freezing specific blocks of layers (Lee et al., 2019; Ingle et al., 2022), updating only bias terms (Zaken et al., 2022), or dropping layers based on input-output similarity (LayerDrop, (He et al., 2024)). Although computationally cheaper, these heuristic-based methods often yield suboptimal performance because they overlook the complex semantic interdependencies between layers (Tenney et al., 2019). Critically, both strategies can suffer from poor generalization, exhibiting performance degradation on out-of-domain data, suggesting they either overfit to the source domain or fail to capture truly transferable features.

This reveals a critical gap: the need for a method that is both computationally inexpensive like static heuristics and principled like dynamic selection. We propose that the key to selecting the most impactful layers for fine-tuning lies not in costly training-time dynamics, but in the intrinsic semantic structure of the pre-trained model itself. We propose that by measuring the semantic coherence between layers, we can identify which ones act as central "hubs" of information—those whose representations are most aligned with the rest of the model. These layers are prime candidates for adaptation, as modifying them is likely to have a harmonious

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 10313–10323
11–16 May 2026.

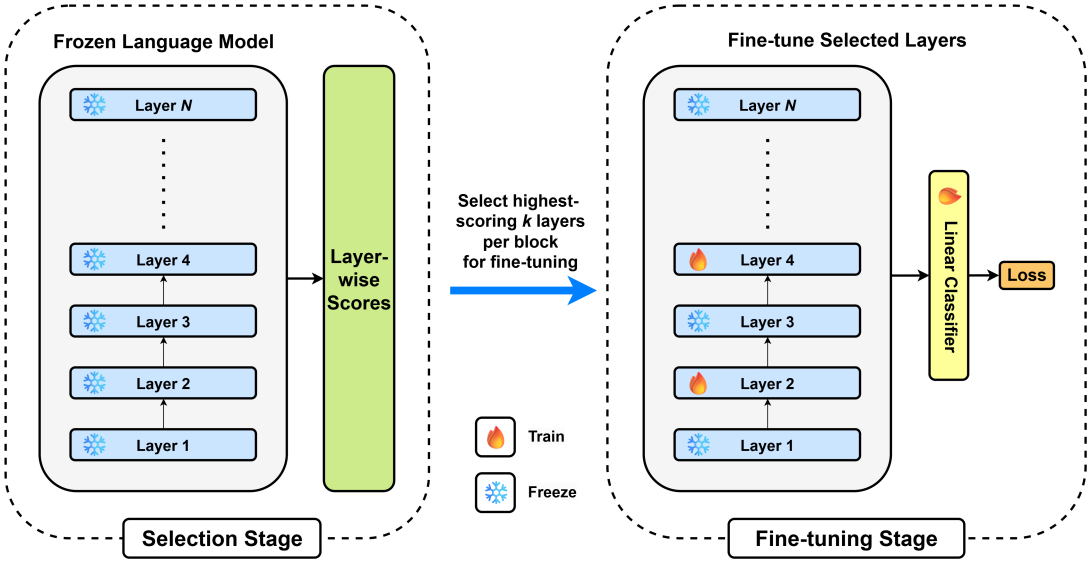

Figure 1: Overall methodology for PRiSM. We adopt a two-stage approach: a selection stage and a fine-tuning stage. In the selection stage, we calculate the cosine similarity between the aggregate token of each layer with all other layers to find layer-wise scores. In the fine-tuning stage, we select the block-wise highest-scoring k layers and fine-tune only those while keeping the remaining layers frozen.

and widespread impact on the model's overall function.

To this end, we introduce PRiSM (**P**artial **R**anking via **i**nter-layer **S**emantic **M**easurement), a novel, training-free approach for layer selection. PRiSM operates in a single, efficient forward pass before fine-tuning begins. It scores each layer by calculating the cosine similarity between its aggregate token representation (e.g., the [CLS] token) and those of all other layers. This score quantifies a layer's semantic centrality. We then fine-tune a small, block-wise subset of the highest-scoring layers. This approach directly addresses the limitations of prior work: it is training-free, avoiding the overhead of dynamic methods, while being more principled than simple heuristics by explicitly leveraging inter-layer semantic relationships.

Our contributions are threefold. First, we present a novel method that achieves a compelling trade-off, reducing trainable parameters by 75% and accelerating training by 1.5×, while performing within 0.7% of full fine-tuning accuracy. Second, through extensive experiments on 15 diverse NLP datasets, we show that PRiSM consistently outperforms existing PEFT and partial fine-tuning baselines. Finally, we demonstrate PRiSM's superior generalization, as it maintains robust performance in cross-domain evaluations where other methods falter.

2. Related Work

In this section, we discuss two lines of work on partial fine-tuning of pre-trained language models.

Heuristic Layer Selection and Freezing Strategies. This area explores static strategies for reducing trainable parameters. Approaches include freezing a fixed portion of the model, such as the initial or final layers (Lee et al., 2019; Ingle et al., 2022), or updating only bias terms (Zaken et al., 2022). Other methods employ more complex heuristics to identify layers or nodes to freeze or drop, using techniques like evolutionary search (Shen et al., 2021), node value changes during an initial tuning phase (Vucetic et al., 2022), or input-output similarity to find redundant layers (Layer Drop, (He et al., 2024)). Unlike these methods, which often overlook layer interdependencies, our approach explicitly models inter-layer semantic relationships to guide selection.

Dynamic Layer Selection and Efficient Training. This more recent line of work focuses on dynamically selecting trainable components during the fine-tuning process. Techniques include iteratively freezing layers based on gradient analysis (SlimFit, (Ardakani et al., 2024)), sequentially fine-tuning one layer at a time (LIFT, (Zhu et al., 2023)), and pruning weights using Taylor approximation (RECAP, (Ilhan et al., 2024)). Other methods focus on memory efficiency by selectively propagating gradients (Simoulin et al., 2023), randomly freezing middle layers (Pan et al., 2024), or estimating layer importance via semantic analysis of latent representations (Gu et al., 2024). Our approach deviates from these by introducing a simple, training-free selection mechanism that leverages pre-trained representations, avoiding the computational overhead of iterative updates or gradient analysis.

3. Methodology

PRiSM has two distinct stages: i) Selection Stage ii) Partial Fine-tuning Stage. In the selection stage, we select the layers that should be fine-tuned and in the fine-tuning stage, we allow those layers to be fine-tuned along with the classifier head, while keeping the other layers frozen. The selection stage doesn't require any training of parameters. The details are written in Section 3.1 and Section 3.2. The overall methodology is depicted in Fig 1.

3.1. Selection Stage

After passing a sentence, represented as a token sequence $S = (t_1, t_2, \ldots, t_n)$, into a pre-trained LM f_{LM} with L layers, we extract the hidden representations. For layer $l \in \{1, \ldots, L\}$, the model outputs a sequence of hidden state vectors, represented as a matrix $H^{(l)} \in \mathbb{R}^{N \times d}$, where N is the sequence length and d is the hidden dimension. We define the **aggregate token representation** for layer l as $h^{(l)}_{\text{agg}}$, a single vector that summarizes the sentence's contextual information at that depth. The choice of this vector depends on the model's architecture: for encoder models like BERT, it is the representation of the [CLS] token, $h^{(l)}_{\text{agg}} = H^{(l)}_{[CLS]}$; for autoregressive models, it is the representation of the final token in the sequence, $h^{(l)}_{\text{agg}} = H^{(l)}_{n}$.

To compute a selection score for each layer l, we measure the cosine similarity between its aggregate token representation $h^{(l)}_{\text{agg}}$ and those from all other layers, excluding itself. This score quantifies how semantically aligned a layer's summary representation is with the representations across the rest of the model. The selection score is obtained by averaging these similarity values. Mathematically, the score for layer l for a single input sequence is defined as:

$$\text{Score}(l) = \frac{1}{L-1} \sum_{j=1, j \neq l}^{L} \text{cos\_sim}(h^{(l)}_{\text{agg}}, h^{(j)}_{\text{agg}})$$

To obtain the final layer-wise scores for a given downstream task, we compute and aggregate these similarity scores for each batch over the entire dataset. We then calculate the global average score for each layer to ensure the selection is representative of the complete data distribution.

After calculating the averaged score for each layer, we rank them and choose k layers ($k < L$) for fine-tuning. We adopt a structured selection strategy to maximize representational diversity. The L layers are partitioned into k contiguous, non-overlapping blocks $\{B_1, B_2, \ldots, B_k\}$. Within each block B_j, we select only the single highest-scoring layer l^*_j for the fine-tuning set $\mathcal{L}_{tune}$, such that $l^*_j = \arg\max_{l \in B_j} Score(l)$ and $\mathcal{L}_{tune} = \{l^*_1, \ldots, l^*_k\}$.

This approach is twofold in its motivation. First, the block-wise division is theoretically grounded in the hierarchical nature of language models, ensuring that fine-tuning updates are distributed across the model's entire representational depth rather than being concentrated in one region. Second, selecting the highest-scoring layer is crucial because a high score signifies high average cosine similarity, identifying the layer as a "semantic hub" whose representation is most aligned with the overall model. Fine-tuning these central, stable layers is more effective, as their adjustments can harmoniously propagate throughout the network, leading to a more coherent adaptation to the downstream task. In contrast, low-scoring layers are semantic outliers, and tuning them risks localized, less impactful updates. As experiments in Section 7 verify, this structured selection is critical, with alternative strategies like non-block or lowest-score selections leading to significant performance degradation.

Why use cosine similarity to measure the importance of layers? In deep learning, similarity metrics are crucial for evaluating relationships between vector representations, with cosine similarity being favored for its focus on directional alignment rather than magnitude (Chen et al., 2020). Unlike dot product and Euclidean distance, which consider both direction and magnitude, cosine similarity measures only the angle between vectors, normalizing them to unit length. This is especially useful in Transformer models with Pre-Norm architectures, where hidden states' magnitude increases with layer depth (Liu et al., 2023; Chen et al., 2025; Xiong et al., 2020). The increase in magnitude can bias dot product similarity, while Euclidean distance may misrepresent similarity between vectors with large magnitudes but similar directions. Cosine similarity, by normalizing magnitude, provides a more reliable measure of semantic similarity, making it ideal for tasks like sentence embeddings where direction is key.

Why use pre-trained aggregate tokens? In transformer-based models, specific tokens are architecturally positioned to serve as holistic sequence representations. For encoder models like BERT, the [CLS] token is explicitly pre-trained to aggregate sentence-level semantics for classification tasks. Its representation at any layer is a distilled summary of the contextual understanding achieved at that stage of processing (Jawahar et al., 2019). Similarly, in autoregressive models, the hidden state of the final token must encapsulate the entire preceding context to predict the next word, effectively functioning as the most comprehensive summary of the input. By comparing these single aggregate vectors across layers, we directly measure the semantic stability and contribution of each layer to the final, overall representation.

This approach simplifies the similarity measurement to a direct vector comparison, avoiding the computational overhead of alternatives like mean pooling (taking the average of all token's hidden state vectors). Mean pooling can also dilute the signal by averaging over all tokens, including less informative ones, and fails to leverage the model's own learned summarization mechanism. The [CLS] or final token, by contrast, provides a representation that the model has been trained to treat as the primary carrier of sequence-level meaning. Therefore, using cosine similarity on these specific tokens offers a more theoretically grounded and computationally efficient method for identifying layers whose semantic contributions are most central to the model's foundational knowledge, making them prime targets for fine-tuning (Jiang et al., 2025).

3.2. Partial Finetuning Stage

In the partial fine-tuning stage, we update only the parameters of the k selected layers, denoted by the set $\mathcal{L}_{\text{tune}}$, while all other layers in the language model backbone remain frozen. For a given downstream task, the model performs a full forward pass to generate representations. We then take the aggregate token representation from only the final layer of the model, $h_{\text{agg}}^{(L)}$, as the holistic sentence embedding. This single vector is fed into a newly initialized linear classifier, which is trained from scratch. During the backpropagation process, gradients are computed and used to update the weights of both the linear classifier and the selected layers in $\mathcal{L}_{\text{tune}}$.

Formally, let the parameters of the language model layers be $\{\theta_1, \ldots, \theta_L\}$ and the classifier parameters be θ_c. The set of trainable parameters is $\Theta_{\text{train}} = \{\theta_l \mid l \in \mathcal{L}_{\text{tune}}\} \cup \{\theta_c\}$. For an input sentence, the model computes the final layer's aggregate token representation $h_{\text{agg}}^{(L)}$. The predicted logits are then calculated as $\hat{y} = f_{\text{classifier}}(h_{\text{agg}}^{(L)})$. A loss function, $\mathcal{L}(\hat{y}, y)$, is computed based on the ground-truth label y. The optimization step updates only the parameters within Θ_{train}, effectively adapting a small, strategically chosen subset of the model for the specific task while preserving the majority of the pre-trained knowledge.

4. Experimental Setup

Datasets. We present an extensive evaluation of our method and baselines encompassing 15 distinct NLP datasets, spanning both classification and regression problems. Following previous work (Gao et al., 2021), we selected 15 English language tasks for evaluation, consisting of 8 tasks that process individual sentences (single-sentence tasks) and 7 that handle sentence pairs (sentence-pair tasks). The dataset collection is drawn from the GLUE benchmark (Wang et al., 2018), the SNLI corpus, and six additional well-established sentence classification datasets: SST-5, MR, CR, MPQA, Subj, and TREC.

Baselines. We compare our method with several baselines, including full fine-tuning (where all layers are fine-tuned), LoRA (Hu et al., 2021), Adapters (Houlsby et al., 2019), Prefix-Tuning (Li and Liang, 2021), LIFT (Zhu et al., 2023), RECAP (Ilhan et al., 2024), and SlimFit (Ardakani et al., 2024). Additionally, we explore layer selection strategies: random selection (where k layers are chosen randomly), top selection (top k layers fine-tuned), and bottom selection (bottom k layers fine-tuned). For consistency, k is set to 3 for all selection-based baselines, including our method. We also compare our method with input-output similarity-based selection, as proposed in Layer Drop (He et al., 2024), which calculates the cosine similarity between the input and output hidden states of each transformer layer. We adopt it by fine-tuning the k layers with the lowest input-output similarity. We also experiment with different values of k in Section 7.

Models. We use publicly available transformer models through the HuggingFace library (version 4.45.1) and implement them using the PyTorch framework (version 2.3.0). Specifically, we use the BERT model in its base configuration (`bert-base-cased`) for our main results which has 12 total transformer layers. For autoregressive models used in ablation experiments, we use Gemma-2-2B (Team et al., 2024) and TinyLlama-1.1B (Zhang et al., 2024).

Hyperparameters. We conducted an extensive hyperparameter search to optimize model performance across different learning rates and training epochs. Specifically, we explored learning rates in the range of $\{1\text{e}{-5}, 2\text{e}{-5}, 5\text{e}{-5}\}$ and evaluated model performance across 3, 5, and 10 epochs. The batch size was varied between 16 and 32. We chose the optimal parameters for each of the datasets. We used the AdamW optimizer with β_1=0.9 and β_2=0.99, and `weight decay=0.01`. In the LoRA fine-tuning approach, we apply low-rank adaptation in all query and value vectors with `r=16` and α=32. We set random seeds of [0, 42, 100] for the three different runs in all our experiments.

5. Results

Single Sentence Tasks. As detailed in Table 1, PRiSM consistently achieves the highest performance among all parameter-efficient and partial fine-tuning methods on single-sentence tasks, with an average accuracy of 85.0%. It significantly

Single Sentence Tasks									
	SST-2 (acc)	**SST-5** (acc)	**MR** (acc)	**CR** (acc)	**MPQA** (acc)	**Subj** (acc)	**TREC** (acc)	**CoLA** (acc)	**Avg.**
Full Fine-tuning	91.7	53.7	87.8	88.0	88.4	96.8	96.7	82.4	85.7
PEFT Baselines									
LoRA	88.6	50.9	85.8	85.2	86.1	92.6	92.8	80.3	82.8
Adapters (Houlsby et al., 2019)	89.0	51.1	86.0	85.4	86.3	93.0	93.1	80.5	83.1
Prefix-Tuning (Li and Liang, 2021)	88.7	50.9	85.9	85.3	86.2	92.7	92.9	80.3	82.9
Partial Finetuning Baselines									
Random k layers	88.4	50.2	85.4	84.7	85.5	93.3	95.2	79.6	82.8
Bottom k layers	88.2	50.8	85.1	84.8	85.4	93.2	94.1	79.2	82.6
Top k layers	89.0	51.2	86.6	85.5	86.8	94.9	94.2	80.4	83.6
LIFT (Zhu et al., 2023)	90.4	50.5	86.8	86.1	87.0	95.9	95.8	80.5	84.1
RECAP (Ilhan et al., 2024)	90.3	51.2	86.3	86.7	87.3	96.2	96.3	81.0	84.4
SlimFit (Ardakani et al., 2024)	90.1	51.8	87.5	**87.2**	87.8	**96.3**	96.1	79.5	84.5
Layer Drop (He et al., 2024)	89.1	51.3	86.3	85.2	86.0	95.2	94.0	80.0	83.4
PRiSM (Ours)	**90.7**	**52.5**	**88.0**	86.7	**88.2**	96.2	**96.4**	**81.1**	**85.0**

Table 1: Results for Single Sentence Tasks. We compare our method with full fine-tuning, LoRA, Adapters, Prefix-Tuning, partial fine-tuning, LIFT, RECAP, SlimFit, and Layer Drop. We use k=3 for all layer selection baselines. All the results are the mean of 3 runs with different seeds. **Bold** indicates the best-performing model and underline indicates the second best.

Sentence Pair Tasks								
	MNLI (acc)	**SNLI** (acc)	**QNLI** (acc)	**RTE** (acc)	**MRPC** (F1)	**QQP** (F1)	**STS-B** (Pear.)	**Avg.**
Full Fine-tuning	84.0	90.8	90.9	62.0	81.8	90.5	86.2	83.7
PEFT Baselines								
LoRA	80.5	88.2	84.9	56.3	72.1	88.3	83.4	79.1
Adapters (Houlsby et al., 2019)	81.0	88.5	86.0	58.0	73.0	88.5	83.8	79.8
Prefix-Tuning (Li and Liang, 2021)	80.8	88.3	85.5	57.1	72.5	88.4	83.5	79.4
Partial Finetuning Baselines								
Random k layers	80.2	87.6	86.9	59.7	70.6	87.2	80.5	79.0
Bottom k layers	80.3	87.4	85.7	58.8	71.4	87.9	81.4	79.0
Top k layers	81.6	88.9	87.8	60.9	74.1	88.8	83.0	80.7
LIFT (Zhu et al., 2023)	81.5	89.3	87.6	59.2	74.2	88.4	83.7	80.6
RECAP (Ilhan et al., 2024)	81.3	89.1	87.5	61.3	75.0	89.2	84.4	81.1
SlimFit (Ardakani et al., 2024)	**82.2**	89.2	88.0	58.5	**75.9**	87.8	84.3	80.8
Layer Drop (He et al., 2024)	80.8	88.7	88.1	60.4	72.2	88.8	82.0	80.1
PRiSM (Ours)	81.8	**89.7**	**88.7**	**61.7**	75.3	**89.8**	**84.5**	**81.6**

Table 2: Results for Sentence Pair Tasks. We compare our method with full fine-tuning, LoRA, Adapters, Prefix-Tuning, partial fine-tuning, LIFT, RECAP, SlimFit, and Layer Drop. We use k=3 for all layer selection baselines. All the results are the mean of 3 runs with different seeds. **Bold** indicates the best-performing model and underline indicates the second best.

surpasses popular PEFT baselines, outperforming LoRA by 2.2%, Adapters by 1.9%, and Prefix-Tuning by 2.1%. PRiSM also shows a clear advantage over other partial fine-tuning strategies, outperforming LIFT by 0.9%, RECAP by 0.6%, and SlimFit by 0.5%. Notably, our method's performance is highly competitive with full fine-tuning (85.7%), closing the gap to just 0.7%, and outperforms the fully fine-tuned model on the MR and TREC datasets.

Sentence Pair Tasks. PRiSM continues its strong performance on the more complex sentence-pair tasks, as shown in Table 2. Achieving an average score of 81.6%, it again stands out as the best-performing efficient method. The results demonstrate a consistent pattern of improvement over the PEFT baselines, outperforming LoRA by 2.5%, Adapters by 1.8%, and Prefix-Tuning by 2.2%. It also outperforms other partial fine-tuning methods

Method	Avg Speedup ↑	#Params↓
Full Fine-tuning	-	110 M
PEFT		
LoRA	**1.63**	**1.84 M**
Adapters	1.35	3.6 M
Prefix-Tuning	1.58	2.1 M
Partial Finetuning		
Random	1.47	**28.3 M**
LIFT	0.53	29.7 M
RECAP	1.39	32.1 M
SlimFit	1.21	40.7 M
PRiSM	**1.54**	**28.3 M**

Table 3: Comparison of number of trainable parameters and average training speedup.

like LIFT by 1.0% and SlimFit by 0.8%. Although full fine-tuning (83.7%) sets the highest benchmark, PRiSM delivers the closest performance, validating the robustness of its layer selection strategy across different task formats.

Training Time and Parameters. Table 3 highlights the efficiency of PRiSM in terms of computational resources. Our method reduces the number of trainable parameters to 28.3M, a 75% reduction compared to the 110M parameters of a fully fine-tuned model. While PEFT methods like LoRA (1.84M) and Adapters (3.6M) are more parameter-frugal, PRiSM's performance gains are substantial. In terms of training efficiency, PRiSM achieves an average speedup of 1.54×, which is highly competitive with the fastest baselines, LoRA (1.63×) and Prefix-Tuning (1.58×). This demonstrates that PRiSM offers a superior trade-off, delivering accuracy near the level of full fine-tuning while providing a noteable reduction in parameters and a compelling training speedup.

6. Cross-Domain Evaluation

To evaluate PRISM's robustness against domain shifts, we conducted a series of out-of-distribution experiments, with results presented in Table 4. The setup involved training models on one dataset and evaluating them on another from the same task category but a different domain. We specifically measured generalization for sentiment analysis (transferring between SST-2 (Socher et al., 2013) and IMDb (Maas et al., 2011)) and for hate speech detection (transferring between the OLID (Zampieri et al., 2019) and Dynamically Generated Hate Speech (Vidgen et al., 2021) datasets). This setup directly tests how well each fine-tuning method's learned adaptations transfer to new, unseen data distributions.

The results highlight PRiSM's superior generalization ability. Across all cross-domain tasks, PRiSM performs exceptionally well, consistently matching the accuracy of full fine-tuning (within 1%) and even outperforming it when transferring from OLID to Dyn. Hate. This robustness stems from our core methodology. Full fine-tuning risks overfitting to the specific stylistic quirks and biases of the source domain. In contrast, PRiSM modifies only 25% of the model's layers, while preserving the vast majority of the model's pre-trained, general-purpose knowledge. This targeted approach prevents the model from over-specializing, ensuring it learns the underlying task rather than just the source domain, which is crucial for real-world deployment.

Train Set	Test Set	Method	Accuracy (%)
SST-2	IMDB	FFT	88.8
		Random	84.5
		LoRA	85.3
		LIFT	84.8
		RECAP	85.0
		SlimFit	86.2
		PRiSM	**87.6**
IMDB	SST-2	FFT	89.0
		Random	84.2
		LoRA	86.9
		LIFT	84.7
		RECAP	85.1
		SlimFit	87.0
		PRiSM	**88.5**
OLID	Dyn. Hate	FFT	52.8
		Random	46.4
		LoRA	47.8
		LIFT	46.7
		RECAP	48.2
		SlimFit	51.3
		PRiSM	**53.0**
Dyn. Hate	OLID	FFT	71.7
		Random	66.0
		LoRA	68.6
		LIFT	66.5
		RECAP	67.3
		SlimFit	69.2
		PRiSM	**71.6**

Table 4: Comparison of Cross-Domain Performance. **Bold** indicates the best-performing model; underline indicates the second best. Here FFT means Fully Fine-Tuning and Random represents Random Selection methods.

Similarity Metric	Average Score
Cosine Similarity	83.3
Euclidean Distance	77.8

Table 5: Comparison with Euclidean distance as the similarity metric.

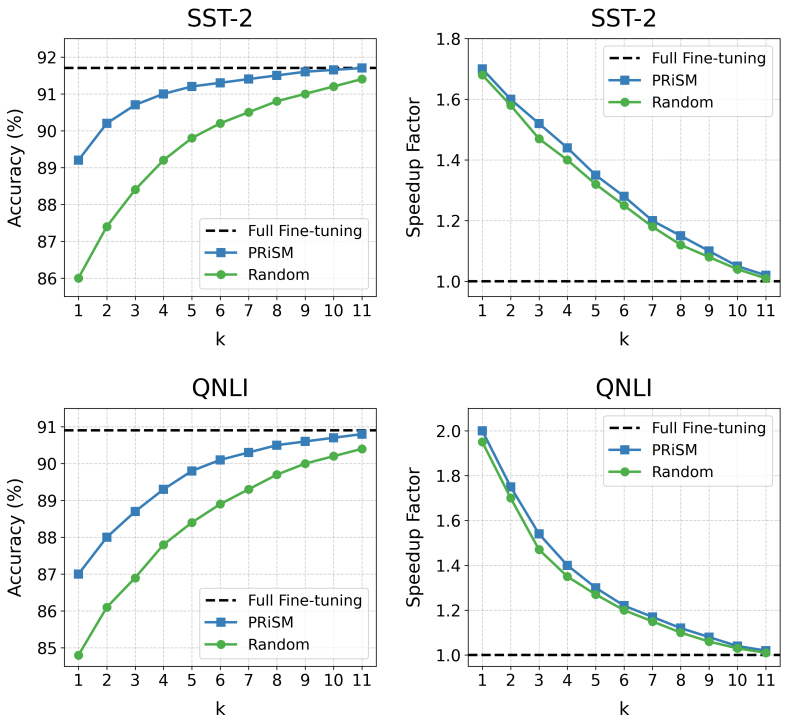

Figure 2: Comparison of accuracy and training speedup across k for SST-2 and QNLI datasets.

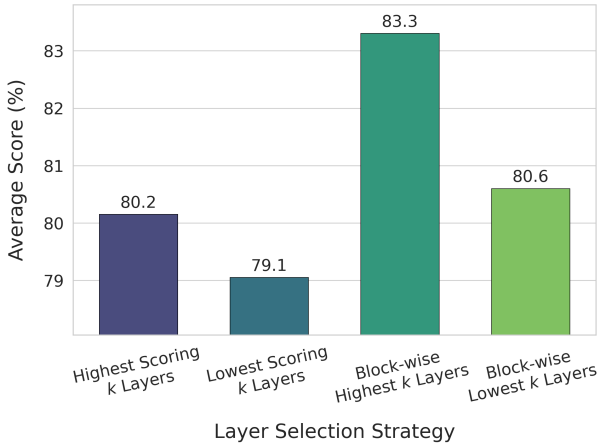

Figure 3: Comparison of performance for different layer selection strategies with the average score across all datasets. We use k=3 for all the layer selection strategies.

7. Ablation

Performance across different k values.

One of the key hyperparameters of our method is the value of k, i.e. the number of layers to select and train based on the selection score. To empirically select the optimal value of k, we perform ablation for all k values (ranging from 1 to 11) on the $\text{BERT}_{\text{BASE}}$ model, considering both the accuracy and the speedup of training time compared to full fine-tuning. We show the ablation graphs for two of the datasets, SST-2 and QNLI, in Fig 2. For both the datasets, we see that accuracy gains becomes limited as the value of k becomes larger than 6. As for the speedup in training time, we observe a continuous decline in speedup as the value of k increases. Considering the trade-off between the accuracy and speedup from these graphs, we choose k=3 to be an optimal value for all of our experiments.

Performance across different layer selection strategies.

To verify that block-wise highest-scoring layers used in PRiSM is the most effective strategy, we perform several ablations by fine-tuning without any

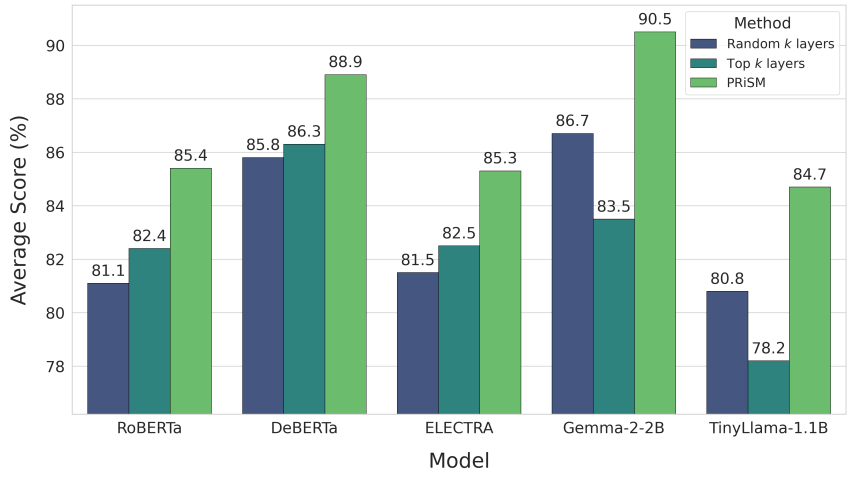

Figure 4: Comparison of performance across different encoder-only and autoregressive language models. We show the average performance score across all datasets. We use k=3 for all the layer selection methods.

block-wise selection and choose the overall highest and lowest-scoring layers instead. We also fine-tune the block-wise lowest-scoring layers to ensure the validity of selecting the most similar layers in our approach. Figure 3 shows that choosing all other strategies causes performance degradation, proving that using the block-wise highest-scoring layers is crucial for achieving high downstream task performance.

The figure compares different layer selection strategies based on their average performance scores. The results show that selecting layers using a block-wise highest-scoring strategy achieves the best performance (83.3%), indicating that informative layers tend to appear in meaningful groups or blocks rather than as isolated individual layers. In comparison, selecting the highest scoring individual k layers yields a slightly lower score (80.2%), suggesting that considering layers independently may overlook useful contextual relationships between neighboring layers. The block-wise lowest k layers still performs moderately (80.6%), while selecting the lowest scoring individual k layers leads to the weakest performance (79.1%). Overall, these findings highlight that block-wise layer selection is more effective than individual layer selection, as it better captures the structural dependencies between layers and helps identify more informative representations within the model.

Performance across different LLMs.

We also report our results for various other encoder-only models, including $\text{RoBERTa}_{\text{BASE}}$, $\text{DeBERTa}_{\text{BASE}}$, and $\text{ELECTRA}_{\text{BASE}}$, as well as two decoder-only large language models, Gemma-2-2B (Team et al., 2024), and TinyLlama-1.1B (Zhang et al., 2024), in Figure 4. We compare our method with random k, bottom k, and top k layer selection.

The results show that larger language models generally achieve higher performance, although the improvement is not strictly proportional to model size. Among the models evaluated, Gemma-2-2B, which is the largest, achieves the highest overall score (90.5%) with PRiSM, suggesting that larger LLMs can better leverage informative layer representations. In contrast, the smaller TinyLlama-1.1B performs comparatively lower across the baseline strategies but still shows a substantial improvement when PRiSM is applied. Encoder-based models such as DeBERTa, RoBERTa, and ELECTRA exhibit moderate performance, with DeBERTa performing the strongest among them. Regarding layer selection strategies, PRiSM consistently outperforms both random k layer selection and top k layer selection across all models, indicating that adaptive or principled layer selection is the most suitable strategy. The top k layers approach is not consistently effective, particularly for LLMs like Gemma-2-2B and TinyLlama-1.1B, where it sometimes performs worse than random selection. Overall, these results highlight the broad applicability of our approach to different families of LMs.

Comparison with Euclidean distance.

To empirically confirm that cosine similarity is the most effective metric for calculating the similarity between layers, as explained in Section 3, we perform our same experiment, but instead of cosine similarity, we use the Euclidean distance to calculate the similarity scores between the aggregate tokens. The result, shown in Table 5, shows conclusive evidence, as using Euclidean distance instead of cosine similarity causes a notable drop in performance.

8. Conclusion

In this paper, we propose a novel selective layer fine-tuning approach that incorporates inter-layer relationships by leveraging layer-wise semantic representations of sentences, a direction that has received limited attention in previous methods. Through experiments on fifteen different datasets, our model consistently outperformed existing approaches for layer selection and partial fine-tuning, highlighting the importance of inter-layer relationships in enhancing performance. Notably, our method achieved competitive results compared to fully fine-tuned models while reducing parameter requirements by three-fourths. Additionally, cross-domain task experiments demonstrated the robustness and generalizability of our approach, enabling reliable predictions across varied tasks. We believe our model and findings contribute to advancing the efficient and effective utilization of language models in downstream applications.

9. Limitations

While PRiSM demonstrates strong performance, its evaluation is primarily focused on sentence-level NLU classification tasks. Although our method performed well for autoregressive LLMs on these tasks, its effectiveness on token-level tasks, or on complex generative tasks like summarization, was left unexplored due to resource limitations. Besides, it is also worthwhile to explore multilingual challenges such as the transliteration problem (Fahim et al., 2024; Haider et al., 2025), as well as tasks that extend beyond standard classification baselines (Ahmed et al., 2024). Furthermore, the experiments are conducted on models up to the 2B parameter scale; how the layer-wise semantic similarity patterns and the method's efficiency translate to significantly larger models (e.g., 70B+) is an open question for future work.

Acknowledgments

We are thankful to Independent University, Bangladesh, for their support of this project. We would also like to express our gratitude to the Center for Computational & Data Sciences (CCDS) for providing computational facilities and supervising this project.

References

Fahim Ahmed, Md Fahim, Md Ashraful Amin, Amin Ahsan Ali, and AKM Rahman. 2024. Improving the performance of transformer-based models over classical baselines in multiple transliterated languages. In *ECAI 2024*, pages 4043–4050. IOS Press.

Arash Ardakani, Altan Haan, Shangyin Tan, Doru Thom Popovici, Alvin Cheung, Costin Iancu, and Koushik Sen. 2024. SlimFit: Memory-efficient fine-tuning of transformer-based models using training dynamics. In *NAACL*.

Simran Arora, Avner May, Jian Zhang, and Christopher Ré. 2020. Contextual embeddings: When are they worth it? *arXiv preprint arXiv:2005.09117*.

Lucas Georges Gabriel Charpentier and David Samuel. 2023. Not all layers are equally as important: Every layer counts bert. *arXiv preprint arXiv:2311.02265*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

Xiaodong Chen, Yuxuan Hu, Jing Zhang, Yanling Wang, Cuiping Li, and Hong Chen. 2025. Streamlining redundant layers to compress large language models. In *ICLR*.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *ICPR*, pages 5482–5487. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Md Fahim, Fariha Tanjim Shifat, Fabiha Haider, Deeparghya Dutta Barua, MD Sakib Ul Rahman Sourove, Md Farhan Ishmam, and Md Farhad Alam Bhuiyan. 2024. Banglatlit: A benchmark dataset for back-transliteration of romanized bangla. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14656–14672.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

Jian Gu, Aldeida Aleti, Chunyang Chen, and Hongyu Zhang. 2024. A semantic-based layer freezing approach to efficient fine-tuning of language models. *arXiv preprint arXiv:2406.11753*.

Fabiha Haider, Fariha Tanjim Shifat, Md Farhan Ishmam, Md Sakib Ul Rahman Sourove, Deeparghya Dutta Barua, Md Fahim, and

Md Farhad Alam Bhuiyan. 2025. Banth: A multi-label hate speech detection dataset for transliterated bangla. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7217–7236.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. 2024. What matters in transformers? not all attention is needed. *arXiv preprint arXiv:2406.15786*.

MohammadSaleh Hosseini, Munawara Munia, and Latifur Khan. 2023. Bert has more to offer: Bert layers combination yields better sentence embeddings. In *EMNLP Findings*, pages 15419–15431.

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 *Proceedings of the 36th International Conference on Machine Learning*.

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*.

Fatih Ilhan, Gong Su, Selim Furkan Tekin, Tiansheng Huang, Sihao Hu, and Ling Liu. 2024. Resource-efficient transformer pruning for finetuning of large models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Digvijay Ingle, Rishabh Kumar Tripathi, Ayush Kumar, Kevin Patel, and Jithendra Vepa. 2022. Investigating the characteristics of a transformer in a few-shot setup: Does freezing layers in RoBERTa help? In *Proceedings of the Fifth BlackboxNLP Workshop*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. 2025. Tracing representation progression: Analyzing and enhancing layer-wise similarity. In *ICLR*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *Proceedings of the 40th International Conference on Machine Learning*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *NAACL*.

Jingcheng Niu, Wenjie Lu, and Gerald Penn. 2022. Does bert rediscover a classical nlp pipeline? In *COLING*, pages 3143–3153.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*.

Zhiqiang Shen, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng. 2021. Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *Proceedings of the AAAI conference on artificial intelligence*.

Antoine Simoulin, Namyong Park, Xiaoyi Liu, and Grey Yang. 2023. Memory-efficient selective finetuning. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*.

Richard Socher, Jean Wu Alex Perelygin, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *ACL*.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J. Clark, Brett H. Meyer, and Warren J. Gross. 2022. Efficient fine-tuning of bert models on the edge. In *ISCAS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP Workshop*.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture. In *ICML*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *NAACL*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Ligeng Zhu, Lanxiang Hu, Ji Lin, and Song Han. 2023. LIFT: Efficient layer-wise fine-tuning for large model models. *Openreview Submission*.