Information Sciences 740 (2026) 123204

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Training-free layer selection for partial fine-tuning of language models

Aldrin Kabya Biswas, Md Fahim, Md Tahmid Hasan Fuad, Akm Moshiur Rahman Mazumder, Amin Ahsan Ali, AKM Mahbubur Rahman*

Center for Computational & Data Sciences, Independent University, Bangladesh, Dhaka, 1245, Bangladesh

HIGHLIGHTS

- Efficient training-free layer selection utilizes token cosine similarity.
- Inter-layer token relationships identify optimal layers for fine-tuning.
- Reduces trainable parameters by 75% with a 1.5x training speedup.
- Our paper achieves better accuracy on 16 diverse datasets compared to the SOTA works.
- Layer selection strategy preserves robust cross-domain generalization.

ARTICLE INFO

Keywords:
Transfer learning
Parameter-efficient fine-tuning
Partial fine-tuning

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. We propose a training-free approach for layer-wise partial fine-tuning that leverages the cosine similarity between representative tokens across layers to identify inter-layer relationships. Our method 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 layers, either highest-scoring, lowest-scoring, or block-wise, while keeping others frozen. We conduct experiments on 16 diverse NLP datasets, including single-sentence and sentence-pair classification tasks, as well as generation 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 comparative baselines in 14 out of 16 evaluated datasets. Additionally, our approach does not cause any notable drop in performance when the domain is changed for the evaluation tasks, demonstrating a robust cross-domain performance.

1. Introduction

Language Models (LMs) play a crucial role in various NLP tasks as they can extract contextual word representations that capture semantic nuances based on their surrounding context, enabling a more accurate and nuanced understanding and generation of text [1]. Consequently, pretrained LMs like BERT [2] have set new state-of-the-art benchmarks across many NLP tasks. Several studies have explored the factors underlying the capabilities of LMs, revealing that their strength lies in capturing hierarchical representations

* Corresponding author.
Email addresses: aminali@iub.edu.bd (A.A. Ali), akmmrahman@iub.edu.bd (A.M. Rahman).

https://doi.org/10.1016/j.ins.2026.123204
Received 14 August 2025; Received in revised form 5 February 2026; Accepted 5 February 2026
Available online 6 February 2026

across different layers of the model [3]. Each layer in BERT, for example, learns to interpret language at varying levels of complexity and abstraction.

Recent research [4], however, has discovered that not all layers of Transformer models are equally important for downstream tasks [5]. As a result, several strategies have been proposed to reduce the number of learnable parameters and memory requirements. Various such partial fine-tuning and parameter-efficient training methods have achieved competitive performance compared to fully fine-tuned models.

Among recent studies, two major lines of work have been explored. The first line employs dynamic layer selection-based strategies, as demonstrated in methods such as SlimFit [6], LIFT [7], SFT [8], and RECAP [9]. These approaches select layers based on similarity scores derived from gradient analysis, selective weight updates, or attention-like mechanisms during each training epoch. The second line of work focuses on heuristic layer selection and freezing strategies. Methods such as Layer Drop [10], and those proposed by Lee et al. [11], or Ingle et al. [12] heuristically select or drop layers without considering their interrelationships with other layers within the model or layer blocks. Both these lines of work, however, suffer from several limitations. First, in dynamic layer selection-based approaches, the gradient-based scores or distances are calculated and updated in every epoch for each sample, leading to significant computational overhead. Additionally, these methods require extra resources for auxiliary networks that are updated during training, resulting in longer training times. Second, the heuristic-based layer freezing, bias-only updates, or layer-selection based on input/output similarity tends to yield less competitive performance [13] as inter-layer relations are overlooked. Third, both strategies exhibit further performance degradation in cross-domain evaluation tasks (i.e., when training occurs in one domain and testing in another), which undermines the model's generalization capability.

We address these limitations by proposing a novel training approach in which layer similarity is calculated based on representative semantic token representations (e.g., the [CLS] token in encoder-only models or the last token in causal decoder models). We focus on these token representations because they encapsulate the overall contextual information [14] of the input sequence [15]. However, to the best of our knowledge, layer similarity calculation based on such representative semantic tokens has not yet been explored. Thus, our method explicitly captures inter-layer relationships to select layers that are most relevant to the downstream task. Since these tokens are obtained during a single forward pass, calculating similarity scores across different layers is cost-effective. Moreover, the use of semantic tokens during the forward pass helps preserve the model's generalization capability.

Our solution consists of two simple stages: a training-free layer selection stage and a partial fine-tuning stage. In the selection stage, we first pass a sentence through a pre-trained language model and assign a selection score to each layer based on the cosine similarity of the representative tokens immediately after the forward pass. In the second stage, we fine-tune only a few selected layers, based on their scores, while keeping the remaining layers frozen. Therefore, our method does not require training additional parameters or designing extra networks, unlike the contemporary approaches [16]. Furthermore, our strategy maintains strong performance in cross-domain evaluation tasks since layers are selected based on their interrelationships, unlike the heuristic layer selection or freezing methods [11]. Finally, our method allows for the selection of layers across different layer blocks, ensuring diversity in representation.

Overall, the key contributions of this paper are as follows:

- We introduce a novel, training-free approach for selecting layers based on inter-layer cosine similarity of representative token representations (such as [CLS] or last token), which reduces trainable parameters by 75% and accelerates training by 1.5×, while being within 0.7% of the accuracy of full fine-tuning. Our approach is training-free because it does not require any backpropagation or parameter updates for layer selection. Instead, a single forward pass of the training samples is sufficient to compute the cosine similarity scores.
- We investigate multiple selection mechanisms, including fine-tuning the highest-scoring layers, the lowest-scoring layers, and distributing selections across different layers by grouping layers into blocks. We conduct extensive experiments on 16 diverse NLP datasets spanning single-sentence classification, sentence-pair classification, and text generation tasks, which show that our approach outperforms existing baselines.
- We demonstrate that our approach maintains strong generalization across domain shifts, as shown by improvements in out-of-domain evaluations, and exhibits scalability to larger models and datasets, ensuring that models retain their effectiveness even when applied to novel data distributions or resource-intensive scenarios.

2. Related work

In this section, we discuss different lines of work that employ parameter-efficient fine-tuning approaches to reduce the parameter count and model depth of pre-trained language models.

2.1. Heuristic layer selection and freezing strategies

Early work by Lee et al. [11] demonstrated that fine-tuning only the last 25% of layers in BERT and RoBERTa could achieve 90% of full model performance. Shen et al. [17] introduced a partial transfer approach using evolutionary search to determine optimal layer freezing patterns in few-shot learning scenarios. Vucetic et al. [18] developed Freeze And Reconfigure (FAR), which freezes feed-forward network nodes based on their value changes during an initial priming phase. Similarly, Ingle et al. [12] found that freezing the first 50% of layers in RoBERTa could improve few-shot learning performance while reducing training time. A recent work by He et al. [10] proposes Layer Drop, which uses cosine similarity between the input and output of the Transformer layers to identify redundancy, and drops the redundant layers rather than freezing them. While these methods rely on heuristic layer freezing,

bias-only updates, or complex optimization techniques, our method explicitly captures inter-layer relationships to select layers most relevant to the task.

2.2. Dynamic layer selection and efficient training

More recent approaches have focused on dynamic layer selection and efficient training strategies. Ardakani et al. [6] proposed SlimFit, which analyzes gradient contributions to iteratively freeze less important layers. Zhu et al. [7] introduced LIFT, a layer-wise fine-tuning approach that updates only one Transformer layer at a time sequentially. Simoulin et al. [8] presented a memory-efficient approach that selectively propagates gradients through a subset of input token positions. Pan et al. [19] proposed LISA, which randomly freezes a subset of the middle layers iteratively during training to save memory. Gu et al. [20] proposed a semantic analysis-based layer freezing method, using deviations between virtual and factual transitions of latent representations to estimate the fine-tuning gain for each layer. Ilhan et al. [9] presented RECAP, which uses first-order Taylor approximation to compute the less important weights to prune the model and then uses Fisher information for masking the weights that cause the least change in model predictions during fine-tuning. Our approach deviates from these by introducing a simple training-free layer selection mechanism based on cosine similarity, avoiding complex iterative updates, gradient analysis, or semantic transition estimation, and instead uses pre-trained [CLS] token representations to capture inter-layer relationships directly.

2.3. Advanced fine-tuning and selective adaptation strategies

Beyond direct layer selection, recent research has expanded into advanced parameter-efficient fine-tuning (PEFT) frameworks and architectural modifications to address specific adaptation challenges. For instance, Ding et al. [16] proposed a Hypernetwork-based PEFT (HyperPEFT) framework designed to bridge pre-trained models with continual learning. By utilizing hypernetworks to generate task-specific parameter adjustments, they effectively mitigate catastrophic forgetting while facilitating knowledge sharing across tasks. Similarly, targeting generalization in target-unspecific tasks, Cui et al. [21] introduced a Features Matrix (FM) approach. This method incorporates hand-crafted prompts to extract general knowledge from frozen models, thereby preventing overfitting and enhancing performance on novel classes and cross-dataset scenarios.

3. Methodology

Our proposed methodology for k-layer selective fine-tuning has two different stages: i) a Selection Stage, and ii) a 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 does not require any training. The details are provided in Section 3.1 and Section 3.2. The overall methodology is depicted in Fig. 1.

3.1. Selection stage

After passing a sequence S into a pre-trained LM f_{LM}, we extract the layer-wise representations for the input tokens. Specifically, for layer l, we extract the hidden representations $H^l = \{h_1^l, h_2^l, \ldots, h_n^l\}$, where n is the number of input tokens. If f_{LM} has L layers in total, we extract all layers' representations $\mathcal{H} = \{H^1, H^2 \ldots, H^L\}$.

For each layer l, we identify a *representative token*, denoted as h_{rep}^l, which encapsulates the semantic representation of the entire sequence. For encoder-only models like BERT, this corresponds to the special classification token ([CLS]). For decoder-only models

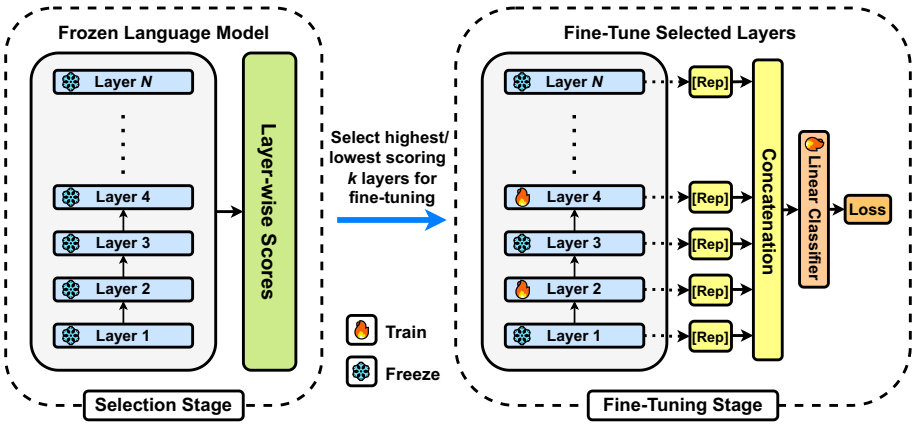

Fig. 1. Overall methodology for k-Layer selective fine-tuning. We adopt a two-stage approach: a selection stage and a fine-tuning stage. In the selection stage, we calculate the cosine similarity between representative tokens (e.g., the [CLS] token or the last token) of each layer with all other layers to find layer-wise scores. In the fine-tuning stage, we select the highest/lowest scoring k layers and fine-tune only those while keeping the remaining layers frozen. See Section 4.3 for specific details on the prediction head architecture and token extraction for encoder vs. Decoder models.

like GPT-2, we utilize the representation of the last token in the sequence, as it aggregates the context of all preceding tokens in causal attention mechanisms.

To compute the selection score for layer l, we measure the cosine similarity between h^l_{rep} and the representative token representations from all other layers, excluding h^l_{rep} itself. The selection score is obtained by averaging these cosine similarity values. Mathematically, the score for layer l is defined as:

$$\text{Score}(l) = \frac{1}{L-1} \sum_{\substack{i=1 \\ i \neq l}}^{L} \cos\_sim \left(h^l_{\text{rep}}, h^i_{\text{rep}} \right)$$

After calculating the score for each layer, we rank them and choose k layers ($k < L$) for fine-tuning in the downstream task. We consider three distinct strategies to select these k layers:

- **High Scoring k layers**: In this approach, after computing the scores for all layers, we select the k layers with the highest scores for fine-tuning. High cosine similarity often indicates that these layers function as stable information anchors, maintaining strong connectivity between raw input embeddings and final outputs. By fine-tuning these layers, we adapt the model's logical reasoning mechanisms while preserving the stability of the inference chain. This strategy is particularly effective for tasks requiring robust structural inference, where preserving the integrity of central integration layers is crucial for performance.
- **Low Scoring k layers**: This strategy is the reverse of the previous one. We select the k layers with the lowest scores for fine-tuning. Low similarity scores signify pronounced representational divergence, indicating where the model aggressively distorts the feature manifold to separate classes or predict tokens. Fine-tuning these layers allows for the optimization of specific parameters responsible for projecting generic features into the task-specific space. This strategy aligns with targeted adaptation for semantic classification and generation tasks, facilitating significant transformations without perturbing the stable feature extraction in other layers.
- **Block-wise k layers**: Here, we divide the L layers into k blocks. Within each block, we choose either the highest or lowest-scoring layer to be fine-tuned. This method ensures coverage across the model's depth, addressing scenarios where feature importance is non-monotonic or distributed with localized peaks (e.g., specific grammatical violations). Furthermore, by selecting the most robust representations within blocks, this strategy mitigates the risk of fine-tuning unstable or noisy deep layers that may exhibit low similarity due to semantic confusion rather than useful specialization. This balanced approach promotes a comprehensive adaptation that captures distributed features across various levels of abstraction.

3.1.1. 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 [22]. 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 [23]. The increase in magnitude can bias dot product similarity, while Euclidean distance may misrepresent similarity between vectors with large magnitudes but similar directions [24]. 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.

3.1.2. Representative tokens as an effective tool to capture inter-layer relationships

In Transformer-based models, specific tokens are often designated to aggregate the global context of the input sequence. For encoder-only architectures like BERT, the special [CLS] token serves this purpose, capturing a summary of the sequence for classification tasks. Similarly, in causal decoder-only architectures like GPT-2, the representation of the last token accumulates the context of all preceding tokens. As these representative tokens propagate through the network, their representations evolve, refining task-specific features from shallow syntactic patterns to deep semantic structures [25].

Cosine similarity, which measures the directional alignment between vectors, proves to be an effective metric for comparing these representative token embeddings across layers. High cosine similarity indicates minimal directional change, suggesting stability, while low similarity points to significant representational transformations, highlighting layers where the model is aggressively adapting the feature space [26].

For the purpose of fine-tuning, where the goal is to efficiently adapt the model to a new task while minimizing overfitting and computational cost, this approach highlights inter-layer relationships: selecting layers with pronounced representational changes (low similarity) targets the most influential parameters for feature projection, while layers with high similarity can be fine-tuned to adapt the model's logical reasoning mechanisms while preserving the stable information anchors. Thus, the cosine similarity of representative tokens offers a theoretically grounded method to identify inter-layer relationships and enable strategic, targeted fine-tuning to conserve computational resources.

3.2. Partial fine-tuning stage

At this stage, we train only the selected k layers, keeping the remaining layers frozen. The configuration of the prediction head depends on the nature of the downstream task:

- **Classification Tasks:** We extract the representative token embeddings (e.g., [CLS]) from all layers, both selected and unselected, and concatenate them. A linear classifier is then applied to this aggregated representation.
- **Generation Tasks:** We utilize the model's standard language modeling head. The selected k layers are optimized using the autoregressive loss, while the head and other layers remain frozen.

During training, gradients are backpropagated only through the classification/generation head and the selected k layers.

Specifically for the classification architecture used in our encoder-only experiments, let $\mathcal{H} = \{H^1, .., H^j, .., H^L\}$, where H^j (for $j \in \{1, 2, \ldots, k\}$) represents the selected layers that will be fine-tuned. We extract the representative token embedding from each layer, including both selected and unselected layers. The concatenated representation is given by:

$$H_{\text{final}} = \text{concat}\left(H^1_{\text{rep}}, H^2_{\text{rep}}, \ldots, H^L_{\text{rep}}\right)$$

where H^l_{rep} denotes the representative token embedding for layer l. The concatenated representation H_{final} is then passed through a linear classifier. Algorithm 1 outlines the complete process for our k-layer selection strategy.

4. Experimental setup

4.1. Datasets

We present an extensive evaluation of our method and baselines encompassing 16 distinct datasets, spanning single-sentence classification, sentence-pair classification, regression, and natural language generation tasks. Following previous work [27], we selected 15 English language understanding tasks drawn from the GLUE benchmark [28], the SNLI corpus, and six additional well-established sentence classification datasets: SST-5, MR, CR, MPQA, Subj, and TREC. Furthermore, we include the E2E NLG Challenge dataset [29] to evaluate generation capabilities. We categorize these datasets based on their input structure and task objective below:

Single-Sentence Tasks. We utilize 8 datasets where the model processes a single input string to predict a class label.

- *Sentiment Analysis:* We use SST-2 (Stanford Sentiment Treebank) for binary sentiment classification and SST-5 for fine-grained sentiment classification across five levels (very negative to very positive). Additionally, we employ MR (Movie Reviews) and CR (Customer Reviews) for binary sentiment detection in movie and product reviews, respectively.
- *Opinion and Subjectivity:* The MPQA (Multi-Perspective Question Answering) dataset is used for opinion polarity detection, while the Subj (Subjectivity) dataset tasks the model with classifying sentences as either subjective descriptions or objective facts.
- *Linguistic Acceptability:* We use CoLA (Corpus of Linguistic Acceptability) to determine whether a given sentence is grammatically correct.
- *Question Classification:* The TREC dataset involves classifying questions into six coarse semantic categories (e.g., person, location, numeric information).

Algorithm 1 k-Layer selective fine-tuning.

Input: Model M with L layers, layers to fine-tune k, method $f \in \{\text{'high scoring'}, \text{'low scoring'}\}$
Output: Trained model M
Initialize: Freeze all layers in M
for batch $\mathcal{B}$ **do**
  Extract representative tokens $\{\mathbf{h}^{\text{rep}}_\ell\}_{\ell=1}^{L}$ (e.g., [CLS] or last token)
  Compute cosine similarity for each layer pair:

$$\text{sim}(\mathbf{h}_{\ell_1}, \mathbf{h}_{\ell_2}) = \frac{\mathbf{h}_{\ell_1} \cdot \mathbf{h}_{\ell_2}}{\|\mathbf{h}_{\ell_1}\| \|\mathbf{h}_{\ell_2}\|}$$

end for
Compute average similarity avg_sim_ℓ for each layer ℓ
Select $\mathcal{L}_{\text{select}}$ as the k layers with:

$$\begin{cases} \text{highest avg\_sim}_\ell & \text{if } f = \text{'high scoring'} \\ \text{lowest avg\_sim}_\ell & \text{if } f = \text{'low scoring'} \end{cases}$$

for $\ell = 1$ to L **do**
  Unfreeze ℓ if $\ell \in \mathcal{L}_{\text{select}}$; else keep frozen
end for
Train M on the dataset

Sentence-Pair Tasks. We evaluate on 7 datasets requiring the model to analyze the relationship between two text segments.

- *Natural Language Inference (NLI):* These tasks require determining if a hypothesis is entailed by, contradicts, or is neutral to a premise. We use MNLI (Multi-Genre NLI), SNLI (Stanford NLI), QNLI (Question NLI; derived from SQuAD to check if a sentence contains the answer to a question), and RTE (Recognizing Textual Entailment).
- *Paraphrase and Duplicate Detection:* MRPC (Microsoft Research Paraphrase Corpus) tasks the model with identifying if two sentences are paraphrases. QQP (Quora Question Pairs) involves identifying if two questions are semantically equivalent.
- *Semantic Similarity:* STS-B (Semantic Textual Similarity Benchmark) is a regression task where the model predicts a similarity score ranging from 0 to 5 for a pair of sentences.

Generative Tasks.

- *Natural Language Generation:* To assess our method beyond classification, we use the E2E NLG Challenge dataset. This is a data-to-text generation task where the model must generate a natural language description of a restaurant based on a structured meaning representation (key-value pairs).

4.2. Baselines

We compare our method with several baselines, including full fine-tuning (where all layers are fine-tuned); PEFT baselines such as LoRA [30], Adapters [31], Prefix-Tuning [32], and $(IA)^3$ [33]; and partial fine-tuning baselines such as LIFT [7], RECAP [9], and SlimFit [6]. 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 results with input–output similarity-based selection, as proposed in Layer Drop [10], which calculates the cosine similarity between the input and output (hidden states) of each transformer layer. High similarity indicates redundancy, while low similarity suggests important layers. We fine-tune both the k low and k high similarity layers for comprehensive comparison with our method. We also experiment with different values of k in Section 7.

4.3. Network architecture and implementation details

We evaluate our approach using two distinct architectures to demonstrate versatility across task types:

Encoder-only (Classification): We utilize the pre-trained `bert-base-cased` model from the Hugging Face Transformers library, which consists of 12 transformer encoder layers with a hidden dimension size of 768. To capture the hierarchical semantic information processed by the network, we extract the hidden states corresponding to the special classification token ([CLS]) from the output of every transformer layer, excluding the initial embedding layer. Consequently, for each input sequence, we obtain a stack of 12 distinct [CLS] vector representations. For the downstream classification task, we employ a concatenation-based prediction head. The [CLS] representations from all 12 layers are concatenated into a single dense vector with a dimensionality of 12×768. This aggregated representation is passed through a dropout layer with a probability of 0.3 to mitigate overfitting before entering a specific classifier block. This classifier consists of a sequential module comprising a fully connected linear layer that projects the concatenated input down to the original hidden size of 768, followed by a Rectified Linear Unit (ReLU) activation function, and a final linear projection layer that maps the features to the number of target classes.

Decoder-only (Generation): For the text generation task, we utilize the `gpt2-small` model (124 M parameters), which also consists of 12 layers with a hidden dimension of 768. Since GPT-2 is a causal language model without a [CLS] token, we adapt our selection stage by utilizing the hidden state of the last token in the input sequence from each layer, as it aggregates the context of all preceding tokens. For the fine-tuning stage, we utilize the model's standard language modeling head without concatenation, applying gradients only to the selected k layers while keeping the head frozen.

During the fine-tuning phase for both architectures, we explicitly freeze the parameters of the unselected transformer layers. Gradient updates are only applied to the weights of the k layers identified by our selection strategy (and also to the classifier head parameters for the encoder-only model).

4.4. 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 $\{1e{-}5, 2e{-}5, 5e{-}5\}$ 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 $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and `weight decay=0.01`. In the LoRA fine-tuning approach for our main results, we applied low-rank adaptation in all query value vectors with `r=16` and α=32. We further analyzed the impact of varying r and α in Section 7. We set random seeds of [0, 42, 100] for the three different runs in all our experiments.

5. Results

5.1. Single sentence tasks

We compare our method with the baselines mentioned in Section 4.2 on single-sentence tasks in Table 1. Our layer selection strategies consistently outperform LoRA, Adapters, Prefix-Tuning, and $(IA)^3$, showing improvements of 2.2%, 1.9%, 2.1%, and 2.0%,

respectively, on average across the eight single-sentence tasks. Compared to partial fine-tuning approaches, our methods surpass the random, top, and bottom k-layer partial fine-tuning strategies by a considerable margin, achieving improvements of 2.1%, 1.3%, and 2.3%, respectively, on average.

Furthermore, our methods outperform LIFT by a notable margin of 0.9%, RECAP by 0.6%, and SlimFit by 0.5%, on average. Specifically, our `Block-wise lowest layer` strategy achieves the best performance with an average accuracy of 85.0%, surpassing other methods. Additionally, the `Low Scoring` k `layers` approach also shows competitive performance, with an average accuracy of 84.9%, exceeding other baselines.

On the other hand, our method also achieves comparable results on these datasets for full fine-tuning, being within 0.7% of the full fine-tuning accuracy on average. For the MR and TREC datasets, our method even outperforms the accuracy of fully fine-tuning the model. These results highlight the effectiveness of our layer selection strategies in single-sentence tasks.

While our approach outperforms all baselines on 7 out of 8 datasets in Table 1, we observe that on the CR dataset, our best strategy (Block-wise lowest) slightly underperforms SlimFit (86.9% vs. 87.2%). This marginal gap suggests that for certain tasks, dynamic layer selection methods like SlimFit, which update selection criteria during training, may capture transient optimization dynamics that a static, training-free selection misses. However, this comes at the cost of significantly higher training time and computational resources.

Furthermore, we note that no single selection strategy (High, Low, or Block-wise) is universally superior. As analyzed later in Section 5.5, this variance is not random but stems from the linguistic nature of the tasks: semantic tasks (like SST-2) generally benefit from fine-tuning lower-similarity deep layers (Low Scoring), while structural tasks (like MR) benefit from preserving the inference chain in the middle layers (High Scoring). Therefore, the best performing method is contingent on the downstream task type, a property common in transfer learning scenarios.

5.2. Sentence pair tasks

In Table 2, we present the results for the sentence-pair tasks of our methods along with the baselines. We also observe an almost similar pattern where our model demonstrates consistent improvements over LoRA, Adapters, Prefix-Tuning, and $(IA)^3$, with average score increases of 2.5%, 1.8%, 2.2%, and 2.0%, respectively, across the sentence pair tasks. When compared to partial fine-tuning approaches, our methods outperform the random, top, and bottom k layers-based partial fine-tuning strategies by a notable margin, achieving improvements of 2.3%, 0.9%, and 2.2%, respectively, on average.

Additionally, our approaches outperform LIFT, RECAP, and SlimFit, showing an average improvement of 0.8%, 0.3%, and 0.6%, respectively. Notably, all our methods achieve accuracies that are very close to those of full fine-tuning on these sentence-pair tasks as well.

Table 1
Results for single sentence tasks. We compare our method with full fine-tuning, LoRA, Adapters, Prefix-Tuning, $(IA)^3$, partial fine-tuning, LIFT, RECAP, SlimFit, and input–output similarity baselines. We use $k = 3$ for all layer selection baselines. All results are the mean of 3 runs with different seeds. **Bold** indicates the best-performing model, and underline indicates the second best.

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	$91.7_{\pm 0.3}$	$53.7_{\pm 0.6}$	$87.8_{\pm 0.4}$	$88.0_{\pm 0.5}$	$88.4_{\pm 0.4}$	$96.8_{\pm 0.2}$	$96.7_{\pm 0.2}$	$82.4_{\pm 0.8}$	85.7
PEFT Baselines									
LoRA	$88.6_{\pm 0.5}$	$50.9_{\pm 0.8}$	$85.8_{\pm 0.4}$	$85.2_{\pm 0.7}$	$86.1_{\pm 0.5}$	$92.6_{\pm 0.4}$	$92.8_{\pm 0.4}$	$80.3_{\pm 1.0}$	82.8
Adapters [31]	$89.0_{\pm 0.4}$	$51.1_{\pm 0.7}$	$86.0_{\pm 0.5}$	$85.4_{\pm 0.6}$	$86.3_{\pm 0.4}$	$93.0_{\pm 0.3}$	$93.1_{\pm 0.3}$	$80.5_{\pm 0.9}$	83.1
Prefix-Tuning [32]	$88.7_{\pm 0.5}$	$50.9_{\pm 0.8}$	$85.9_{\pm 0.4}$	$85.3_{\pm 0.6}$	$86.2_{\pm 0.5}$	$92.7_{\pm 0.4}$	$92.9_{\pm 0.3}$	$80.3_{\pm 1.1}$	82.9
$(IA)^3$ [33]	$88.9_{\pm 0.4}$	$51.0_{\pm 0.9}$	$85.8_{\pm 0.5}$	$85.6_{\pm 0.5}$	$86.3_{\pm 0.4}$	$92.8_{\pm 0.4}$	$93.0_{\pm 0.4}$	$80.4_{\pm 1.0}$	83.0
Partial Fine-tuning Baselines									
Random k layers	$88.4_{\pm 1.7}$	$50.2_{\pm 2.1}$	$85.4_{\pm 1.6}$	$84.7_{\pm 1.8}$	$85.5_{\pm 1.6}$	$93.3_{\pm 1.5}$	$95.2_{\pm 1.4}$	$79.6_{\pm 2.2}$	82.8
Bottom k layers	$88.2_{\pm 0.8}$	$50.8_{\pm 0.9}$	$85.1_{\pm 0.7}$	$84.8_{\pm 0.8}$	$85.4_{\pm 0.6}$	$93.2_{\pm 0.6}$	$94.1_{\pm 0.5}$	$79.2_{\pm 1.3}$	82.6
Top k layers	$89.0_{\pm 0.6}$	$51.2_{\pm 0.8}$	$86.6_{\pm 0.5}$	$85.5_{\pm 0.6}$	$86.8_{\pm 0.5}$	$94.9_{\pm 0.4}$	$94.2_{\pm 0.5}$	$80.4_{\pm 1.1}$	83.6
LIFT [7]	$90.4_{\pm 0.4}$	$50.5_{\pm 0.9}$	$86.8_{\pm 0.4}$	$86.1_{\pm 0.5}$	$87.0_{\pm 0.4}$	$95.9_{\pm 0.3}$	$95.8_{\pm 0.3}$	$80.5_{\pm 1.0}$	84.1
RECAP [9]	$90.3_{\pm 0.4}$	$51.2_{\pm 0.7}$	$86.3_{\pm 0.5}$	$86.7_{\pm 0.4}$	$87.3_{\pm 0.3}$	$96.2_{\pm 0.2}$	$96.3_{\pm 0.2}$	$81.0_{\pm 0.9}$	84.4
SlimFit [6]	$90.1_{\pm 0.5}$	$51.8_{\pm 0.6}$	$87.5_{\pm 0.4}$	$\mathbf{87.2}_{\pm 0.4}$	$87.8_{\pm 0.3}$	$\underline{96.3}_{\pm 0.2}$	$96.1_{\pm 0.3}$	$79.5_{\pm 1.2}$	84.5
I/O Similarity-based Selection									
k Low Similarity (Important) [10]	$89.1_{\pm 0.6}$	$51.3_{\pm 0.8}$	$86.3_{\pm 0.5}$	$85.2_{\pm 0.7}$	$86.0_{\pm 0.5}$	$95.2_{\pm 0.4}$	$94.0_{\pm 0.5}$	$80.0_{\pm 1.1}$	83.4
k High Similarity (Redundant) [10]	$89.8_{\pm 0.5}$	$51.8_{\pm 0.7}$	$86.8_{\pm 0.4}$	$85.5_{\pm 0.6}$	$86.8_{\pm 0.4}$	$95.1_{\pm 0.4}$	$95.4_{\pm 0.4}$	$80.6_{\pm 1.0}$	84.0
Layer Selection (Ours)									
High Scoring k layers	$90.6_{\pm 0.4}$	$\underline{52.5}_{\pm 0.7}$	$\mathbf{88.1}_{\pm 0.3}$	$86.0_{\pm 0.5}$	$87.1_{\pm 0.4}$	$96.2_{\pm 0.2}$	$96.2_{\pm 0.3}$	$81.0_{\pm 0.9}$	84.7
Low Scoring k layers	$\mathbf{90.8}_{\pm 0.3}$	$\mathbf{52.9}_{\pm 0.6}$	$87.0_{\pm 0.5}$	$86.7_{\pm 0.4}$	$87.7_{\pm 0.3}$	$96.2_{\pm 0.2}$	$\mathbf{96.8}_{\pm 0.2}$	$80.8_{\pm 1.0}$	$\underline{84.9}$
Block-wise highest layer	$\underline{90.7}_{\pm 0.3}$	$\underline{52.5}_{\pm 0.7}$	$\underline{88.0}_{\pm 0.3}$	$86.7_{\pm 0.4}$	$\mathbf{88.2}_{\pm 0.3}$	$96.2_{\pm 0.3}$	$96.4_{\pm 0.2}$	$\underline{81.1}_{\pm 0.9}$	**85.0**
Block-wise lowest layer	$90.4_{\pm 0.4}$	$52.3_{\pm 0.8}$	$87.7_{\pm 0.4}$	$\underline{86.9}_{\pm 0.4}$	$\underline{88.1}_{\pm 0.3}$	$\mathbf{96.4}_{\pm 0.2}$	$\underline{96.7}_{\pm 0.2}$	$\mathbf{81.2}_{\pm 0.8}$	**85.0**

Table 2
Results for sentence pair tasks. We compare our method with full fine-tuning, LoRA, Adapters, Prefix-Tuning, $(IA)^3$, partial fine-tuning, LIFT, RECAP, SlimFit, and input–output similarity baselines. We use $k = 3$ for all layer selection baselines. All 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	$84.0_{\pm 0.2}$	$90.8_{\pm 0.3}$	$90.9_{\pm 0.3}$	$62.0_{\pm 1.5}$	$81.8_{\pm 0.8}$	$90.5_{\pm 0.4}$	$86.2_{\pm 0.5}$	83.7
PEFT Baselines								
LoRA	$80.5_{\pm 0.4}$	$88.2_{\pm 0.5}$	$84.9_{\pm 0.6}$	$56.3_{\pm 1.8}$	$72.1_{\pm 1.2}$	$88.3_{\pm 0.6}$	$83.4_{\pm 0.7}$	79.1
Adapters [31]	$81.0_{\pm 0.4}$	$88.5_{\pm 0.4}$	$86.0_{\pm 0.5}$	$58.0_{\pm 1.7}$	$73.0_{\pm 1.1}$	$88.5_{\pm 0.5}$	$83.8_{\pm 0.6}$	79.8
Prefix-Tuning [32]	$80.8_{\pm 0.5}$	$88.3_{\pm 0.5}$	$85.5_{\pm 0.5}$	$57.1_{\pm 1.9}$	$72.5_{\pm 1.2}$	$88.4_{\pm 0.6}$	$83.5_{\pm 0.7}$	79.4
$(IA)^3$ [33]	$80.6_{\pm 0.5}$	$88.5_{\pm 0.4}$	$85.7_{\pm 0.6}$	$57.5_{\pm 1.8}$	$72.7_{\pm 1.1}$	$88.2_{\pm 0.7}$	$83.7_{\pm 0.6}$	79.6
Partial Fine-tuning Baselines								
Random k layers	$80.2_{\pm 1.7}$	$87.6_{\pm 1.8}$	$86.9_{\pm 1.6}$	$59.7_{\pm 3.0}$	$70.6_{\pm 2.5}$	$87.2_{\pm 1.8}$	$80.5_{\pm 1.9}$	79.0
Bottom k layers	$80.3_{\pm 0.8}$	$87.4_{\pm 0.8}$	$85.7_{\pm 0.7}$	$58.8_{\pm 2.1}$	$71.4_{\pm 1.4}$	$87.9_{\pm 0.7}$	$81.4_{\pm 0.8}$	79.0
Top k layers	$81.6_{\pm 0.6}$	$88.9_{\pm 0.6}$	$87.8_{\pm 0.5}$	$60.9_{\pm 1.8}$	$74.1_{\pm 1.2}$	$88.8_{\pm 0.5}$	$83.0_{\pm 0.7}$	80.7
LIFT [7]	$81.5_{\pm 0.5}$	$89.3_{\pm 0.4}$	$87.6_{\pm 0.5}$	$59.2_{\pm 1.9}$	$74.2_{\pm 1.1}$	$88.4_{\pm 0.6}$	$83.7_{\pm 0.6}$	80.6
RECAP [9]	$81.3_{\pm 0.4}$	$89.1_{\pm 0.5}$	$87.5_{\pm 0.4}$	$61.3_{\pm 1.6}$	$75.0_{\pm 1.0}$	$89.2_{\pm 0.5}$	$\underline{84.4}_{\pm 0.6}$	81.1
SlimFit [6]	$\underline{82.2}_{\pm 0.4}$	$89.2_{\pm 0.4}$	$88.0_{\pm 0.4}$	$58.5_{\pm 1.8}$	$\mathbf{75.9}_{\pm 0.9}$	$87.8_{\pm 0.7}$	$84.3_{\pm 0.6}$	80.8
I/O Similarity-based Selection								
k Low Similarity (Important) [10]	$80.8_{\pm 0.6}$	$88.7_{\pm 0.5}$	$88.1_{\pm 0.4}$	$60.4_{\pm 1.7}$	$72.2_{\pm 1.3}$	$88.8_{\pm 0.6}$	$82.0_{\pm 0.8}$	80.1
k High Similarity (Redundant) [10]	$81.3_{\pm 0.5}$	$88.9_{\pm 0.5}$	$88.5_{\pm 0.4}$	$60.8_{\pm 1.7}$	$73.4_{\pm 1.2}$	$88.6_{\pm 0.6}$	$81.2_{\pm 0.9}$	80.4
Layer Selection (Ours)								
High Scoring k layers	$82.1_{\pm 0.4}$	$\mathbf{90.1}_{\pm 0.3}$	$\mathbf{90.2}_{\pm 0.3}$	$61.0_{\pm 1.6}$	$75.0_{\pm 1.0}$	$89.0_{\pm 0.5}$	$82.4_{\pm 0.7}$	$\underline{81.4}$
Low Scoring k layers	$\mathbf{82.5}_{\pm 0.3}$	$90.0_{\pm 0.4}$	$89.4_{\pm 0.4}$	$61.4_{\pm 1.5}$	$72.5_{\pm 1.2}$	$89.0_{\pm 0.5}$	$83.3_{\pm 0.6}$	81.2
Block-wise highest layer	$81.8_{\pm 0.4}$	$89.7_{\pm 0.4}$	$88.7_{\pm 0.5}$	$\underline{61.7}_{\pm 1.5}$	$\underline{75.3}_{\pm 0.9}$	$\mathbf{89.8}_{\pm 0.4}$	$\mathbf{84.5}_{\pm 0.5}$	**81.6**
Block-wise lowest layer	$82.0_{\pm 0.4}$	$\underline{90.0}_{\pm 0.3}$	$\underline{90.1}_{\pm 0.3}$	$\mathbf{62.0}_{\pm 1.4}$	$71.6_{\pm 1.3}$	$\underline{89.7}_{\pm 0.4}$	$83.3_{\pm 0.7}$	81.2

5.3. Generation tasks

To assess the versatility of our approach beyond classification and regression tasks, we evaluate our method on the E2E NLG Challenge dataset using the GPT-2 Small model. Since GPT-2 is a decoder-only architecture and does not utilize a [CLS] token, we adapt our selection stage by using the representation of the last token in the input sequence, which encapsulates the context of the entire sequence in causal language models. Table 3 presents the performance comparison across five standard generation metrics: BLEU, NIST, METEOR (MET), ROUGE-L, and CIDEr.

Our layer selection strategies demonstrate strong capabilities in the text generation domain, consistently outperforming Full Fine-tuning. Specifically, our Low Scoring k layers strategy achieves a BLEU score of 68.9, surpassing Full Fine-tuning (67.5) by 1.4 points. Similarly, the Block-wise lowest layer strategy achieves the highest performance among all methods in terms of NIST (8.77) and ROUGE-L (71.0) scores. This suggests that by freezing the majority of the model, our approach acts as a regularizer, preventing the overfitting often observed when fully fine-tuning language models on small-to-medium generation datasets.

Compared to other partial fine-tuning baselines, our method shows clear superiority. All our proposed strategies outperform Random, Bottom k, Top k, LIFT, and RECAP across almost every metric. For instance, our Block-wise lowest layer strategy surpasses SlimFit, the strongest partial fine-tuning baseline, by notable margins in NIST ($+0.14$) and CIDEr ($+0.04$). While LoRA achieves the highest BLEU score (69.3), our method remains highly competitive (within 0.4 BLEU) while outperforming LoRA on the NIST, ROUGE-L, and CIDEr metrics, demonstrating that our training-free selection of layers preserves the critical generation capabilities of the pre-trained model.

5.4. Training time and parameters

We also compare our methods with the baseline methods in terms of both speed and parameter efficiency in Table 4. We notice that LoRA fine-tuning achieves the highest speedup factor of 1.63, as it only has 1.84 M trainable parameters. In contrast, our method provides an average speedup of 1.54, which is slightly lower than LoRA's, but still very competitive when considering the performance gains. Notably, our method uses 28.3 M trainable parameters, which is more efficient than SlimFit (40.7 M) and considerably less than full fine-tuning (116.6 M). Although LoRA is faster, our method consistently outperforms LoRA by up to 3% in terms of task performance. This shows that while LoRA offers quicker training times, our model offers a better trade-off between computational efficiency and performance, making it a superior choice when task accuracy is the priority.

5.5. Layer selection analysis

To investigate why specific layer selection strategies yield superior performance on distinct tasks, we analyze the inter-layer cosine similarity of [CLS] token representations, visualized as topological profiles in Figs. 2 and 3. This analysis reveals that the pre-trained

Table 3
Results for GPT-2 on the E2E NLG Challenge. We compare full fine-tuning against various parameter-efficient and partial fine-tuning baselines. We use $k = 3$ for all layer selection baselines. All results are the mean of 3 runs with different seeds. For all metrics, higher is better. **Bold** indicates the best performance, and underline indicates the second-best.

Method	# Trainable Parameters	BLEU	NIST	MET	ROUGE-L	CIDEr
Full Fine-tuning (FT)	124.4 M	67.5±.6	8.55±.11	45.8±.5	70.2±.6	2.41±.06
PEFT Baselines						
LoRA	1.96 M	**69.3±.4**	8.75±.06	**46.3±.4**	70.8±.5	2.47±.05
Adapters [31]	3.84 M	68.3±.5	8.61±.09	45.9±.5	70.4±.5	2.43±.05
Prefix-Tuning [32]	0.43 M	68.1±.6	8.59±.12	46.0±.4	70.3±.6	2.42±.06
$(IA)^3$ [33]	0.11 M	67.8±.8	8.57±.11	45.8±.6	70.2±.8	2.41±.08
Partial Fine-tuning Baselines						
Random k layers	21.3 M	64.1±2.5	7.25±.82	42.1±1.4	66.1±2.1	1.79±.45
Bottom k layers	21.3 M	66.4±1.4	8.39±.25	45.3±0.9	69.4±1.2	2.37±.15
Top k layers	21.3 M	67.1±1.2	8.48±.21	45.6±.8	69.9±1.0	2.39±.12
LIFT [7]	27.5 M	67.9±.9	8.58±.13	45.9±.6	70.3±.8	2.42±.08
RECAP [9]	29.0 M	68.1±.8	8.60±.11	46.0±.5	70.4±.7	2.43±.07
SlimFit [6]	35.5 M	68.3±.7	8.63±.09	46.1±.5	70.5±.6	2.44±.06
I/O Similarity-based Selection						
k Low Similarity (Important) [10]	21.3 M	67.7±1.0	8.56±.18	45.8±.8	70.1±.9	2.41±.11
k High Similarity (Redundant) [10]	21.3 M	67.3±1.1	8.51±.22	45.7±.9	70.0±1.0	2.40±.14
Layer Selection (Ours)						
High Scoring k layers	21.3 M	68.6±.7	8.67±.10	46.0±.4	70.6±.6	2.45±.06
Low Scoring k layers	21.3 M	68.9±.6	8.73±.09	46.1±.5	70.9±.5	**2.49±.04**
Block-wise highest layer	21.3 M	68.4±.7	8.65±.11	46.0±.5	70.5±.6	2.44±.07
Block-wise lowest layer	21.3 M	68.8±.5	**8.77±.08**	46.2±.5	**71.0±.4**	2.48±.05

Table 4
Comparison of number of trainable parameters and average speedup for BERT. We take the average of the speedup factor across all four of our layer selection methods. We set $k = 3$.

Method	Avg Speedup ↑	#Params↓
Full Fine-tuning	–	116.6 M
PEFT		
LoRA	1.63	1.84 M
Adapters	1.35	3.6 M
Prefix-Tuning	1.58	0.4 M
$(IA)^3$	**1.76**	**0.1 M**
Partial Fine-tuning		
LIFT	0.53	29.7 M
RECAP	1.39	32.1 M
SlimFit	1.21	40.7 M
I/O Similarity based	1.40	31.1 M
Random Selection	1.47	**28.3 M**
Our Method	**1.54**	**28.3 M**

model does not utilize a uniform processing pipeline for all inputs; rather, the semantic transformation of the hidden states varies significantly depending on the linguistic objective of the downstream task.

For semantic classification and generation tasks, specifically SST-2 (Fig. 2, top) and E2E NLG (Fig. 3, bottom), we observe a trajectory of late-stage representational divergence. As indicated by the heatmaps, these models maintain high similarity across the early and intermediate layers, implying generic linguistic processing, but exhibit a sharp reduction in similarity within the final three layers. Theoretically, this indicates that the deep layers are responsible for aggressively distorting the feature manifold [34] to separate classes or predict tokens [35]. This visualization provides a direct causal explanation for our empirical results where the Low Scoring strategy prevails for SST-2 and E2E: by targeting these low-similarity deep layers, the method optimizes the specific parameters responsible for projecting generic features into the task-specific space, thereby facilitating adaptation [36] without perturbing the stable feature extraction in the earlier layers [37].

In contrast, tasks requiring robust structural inference, such as QNLI (Fig. 2, middle), exhibit a centrality-focused topology. The visualizations demonstrate that the middle layers, typically layers 5 through 8, maintain the highest average cosine similarity, effectively functioning as information anchors that connect the raw input embeddings with the final prediction heads. Linguistically, this aligns with prior research suggesting that the intermediate layers of transformer models encode the syntactic dependencies [25] and logical implications essential for Natural Language Inference [37]. Consequently, our High Scoring strategy achieves the best

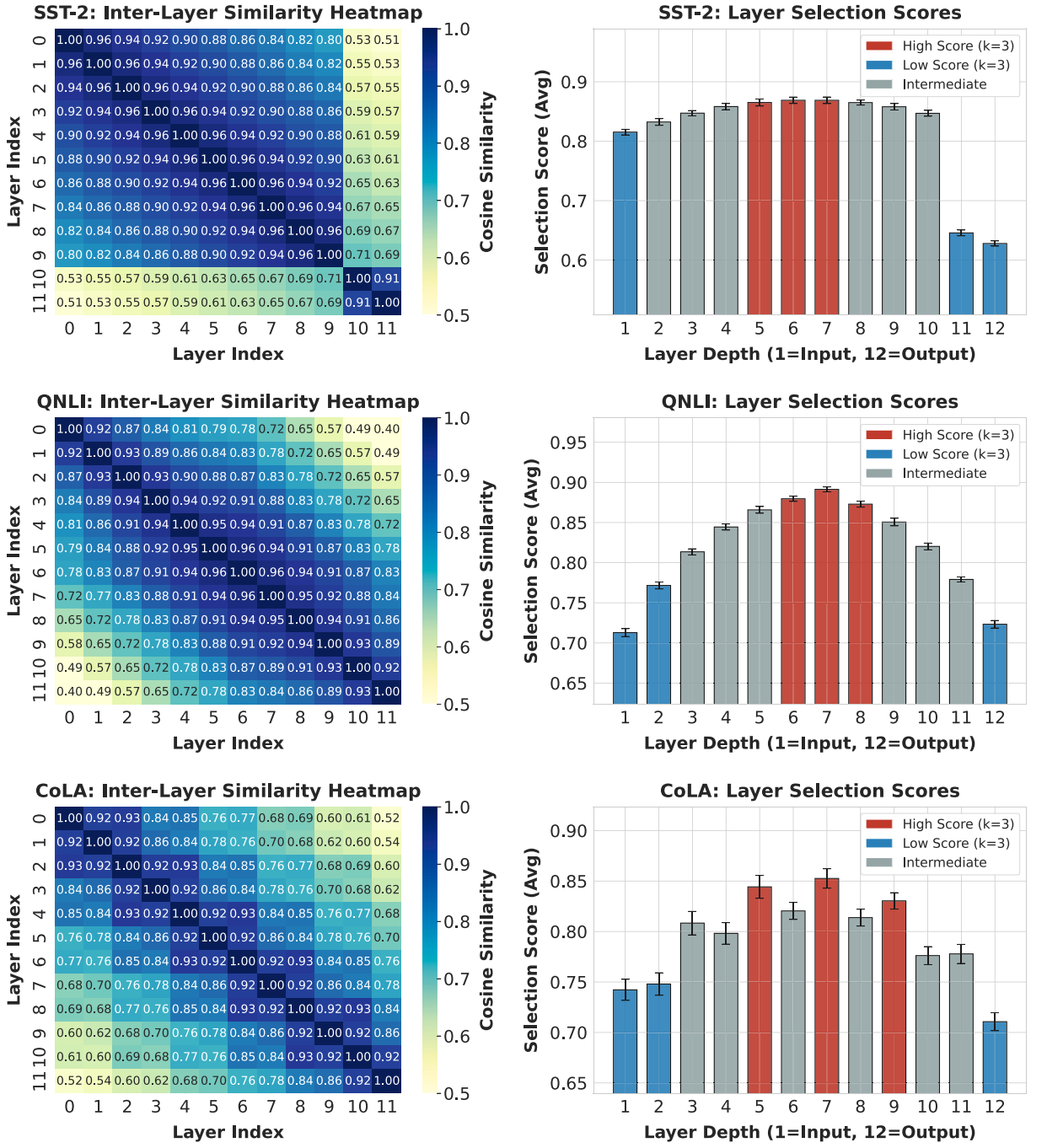

Fig. 2. Heatmaps of inter-layer cosine similarity (left) and layer selection scores (right) for SST-2, QNLI, and CoLA. Error bars show standard deviation. The distinct visual patterns, such as the sharp drop in similarity for deep layers in SST-2 versus the high similarity in the middle layers of QNLI, help explain why different selection strategies (Low Scoring vs. High Scoring) work best in our experiments.

performance for QNLI because it fine-tunes these central integration layers, allowing the model to adapt its logical reasoning mechanisms while preserving the stability of the inference chain. Conversely, selecting the low-scoring peripheral layers in QNLI disrupts the continuity of logical implications, which accounts for the inferior performance of the Low Scoring strategy on this dataset.

Furthermore, the necessity of our Block-wise strategies is theoretically justified by the non-monotonic and early-convergence profiles observed in CoLA, MRPC, and STS-B.

For CoLA (Fig. 2, bottom), the selection scores exhibit a distributed variance with localized peaks rather than a smooth gradient. This profile reflects the nature of linguistic acceptability tasks, which rely on specific grammatical feature violations distributed across

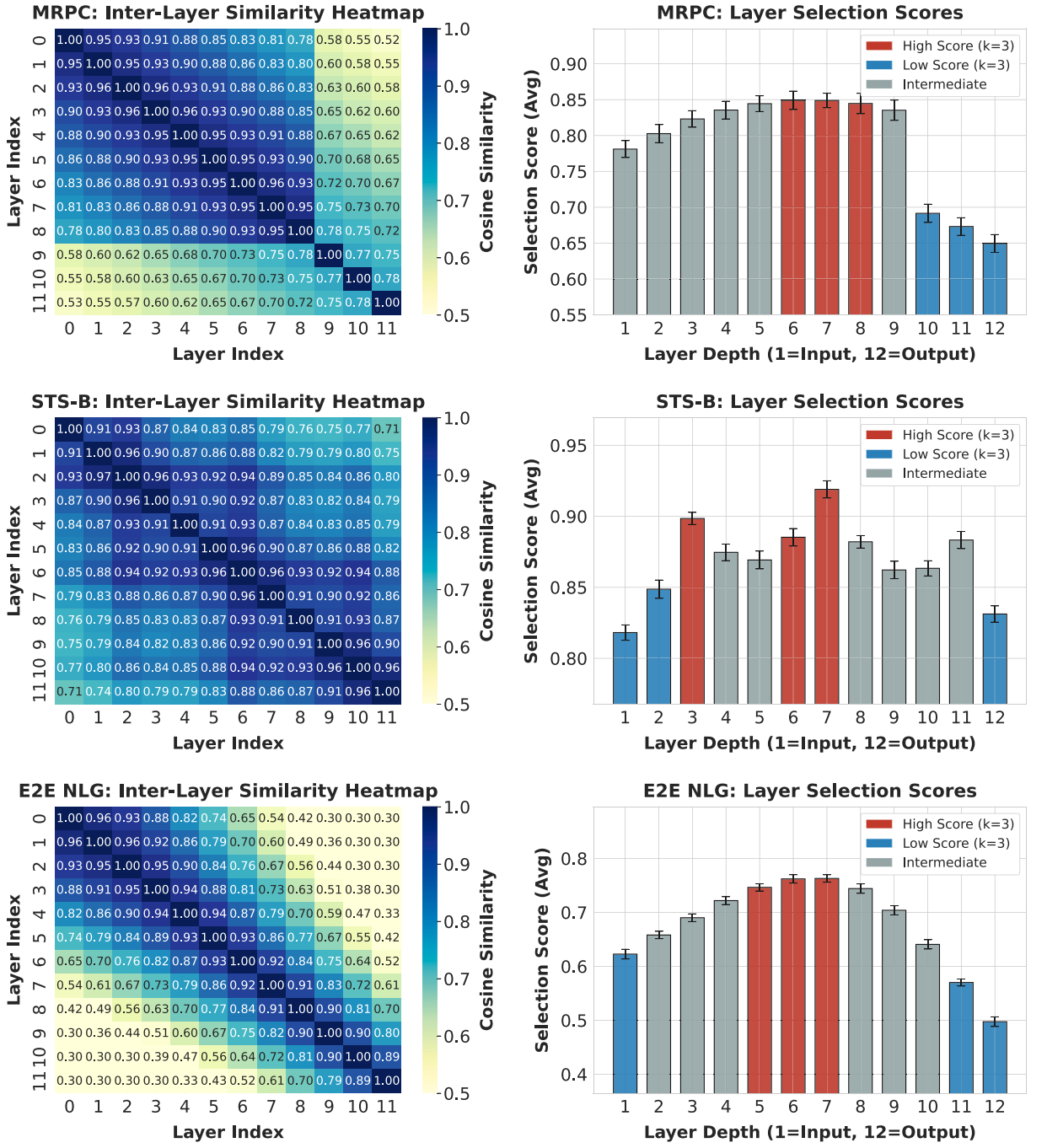

Fig. 3. Heatmaps of inter-layer cosine similarity (left) and layer selection scores (right) for MRPC, STS-B, and E2E NLG (GPT-2). Error bars show standard deviation. The differences in layer patterns, ranging from the instability in the deep layers of MRPC to the gradual change in layers for E2E, align with the effectiveness of block-wise and low scoring strategies, respectively.

the network depth [38] rather than a linear semantic accumulation [39]. A global selection strategy risks missing these disjointed local peaks, whereas our Block-wise approach ensures the coverage of distributed syntactic features.

Similarly, MRPC (Fig. 3, top) displays a profile of early-layer convergence, where shallow and intermediate layers maintain high connectivity, but deep layers show significant stochastic variance. Unlike SST-2, where deep divergence indicates useful specialization, in noisy paraphrase tasks, this variance often represents semantic confusion or over-abstraction [40]. This explains why the Low Scoring strategy performs poorly on MRPC by fine-tuning these unstable deep layers [41], while the Block-wise Highest strategy succeeds by discarding the deep noise and selecting the most robust representations from the stable early and intermediate blocks.

Finally, STS-B (Fig. 3, middle) displays a multi-stage integration profile with similarity peaks in the early, middle, and late blocks, justifying why Block-wise selection, which captures the primary integrator layer from each processing stage, outperforms global strategies [42].

6. Cross-domain evaluation

To assess the domain generalization capability and out-of-distribution performance of our partial fine-tuning approach, we perform several cross-domain experiments and compare them with full fine-tuning and other baselines in Table 5. For the experiments, we train the methods on one training set and evaluate them on the evaluation set of another similar dataset. The datasets used in these experiments are Stanford Sentiment Treebank (SST-2) Dataset [43], IMDb Movie Reviews Dataset [44], Offensive Language Identification Dataset (OLID) [45], and Dynamically Generated Hate Speech Dataset (Dyn. Hate) [46].

We see that our method performs comparably to full fine-tuning, remaining within 1% of the accuracy, especially given that our method only fine-tunes 25% of all the layers. This shows that the cross-domain performance of our method does not degrade even when the majority of the layers are not fine-tuned for the downstream task. This highlights the robustness of our approach and further demonstrates the redundancy of certain transformer layers for further fine-tuning on other datasets.

7. Ablation

7.1. 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 an ablation study 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 Figs. 4 and 5. For the SST-2 dataset, we observe that fine-tuning the lowest-scoring layers gives better accuracy than fine-tuning the highest-scoring layers for all values of k, while for the QNLI dataset, we observe the opposite. Hence, we include the results for both in our main results tables (Tables 1 and 2). As for the speedup in

Table 5
Comparison of cross-domain performance. For our method, we report the result of the best-performing strategy (High/Low/Block) for $k = 3$. **Bold** indicates the best-performing model, and underline indicates the second best.

Method	Training set	Evaluation set	Accuracy
Full Fine-tuning			88.8
Random Selection			84.5
LoRA			85.3
LIFT	SST-2	IMDB	84.8
RECAP			85.0
SlimFit			<u>86.2</u>
I/O Similarity			85.8
Our Method			**87.6**
Full Fine-tuning			89.0
Random Selection			84.2
LoRA			86.9
LIFT	IMDB	SST-2	84.7
RECAP			85.1
SlimFit			<u>87.0</u>
I/O Similarity			86.5
Our Method			**88.5**
Full Fine-tuning			52.8
Random Selection			46.4
LoRA			47.8
LIFT	OLID	Dyn. Hate	46.7
RECAP			48.2
SlimFit			<u>51.3</u>
I/O Similarity			47.6
Our Method			**53.0**
Full Fine-tuning			71.7
Random Selection			66.0
LoRA			68.6
LIFT	Dyn. Hate	OLID	66.5
RECAP			67.3
SlimFit			<u>69.2</u>
I/O Similarity			68.8
Our Method			**71.6**

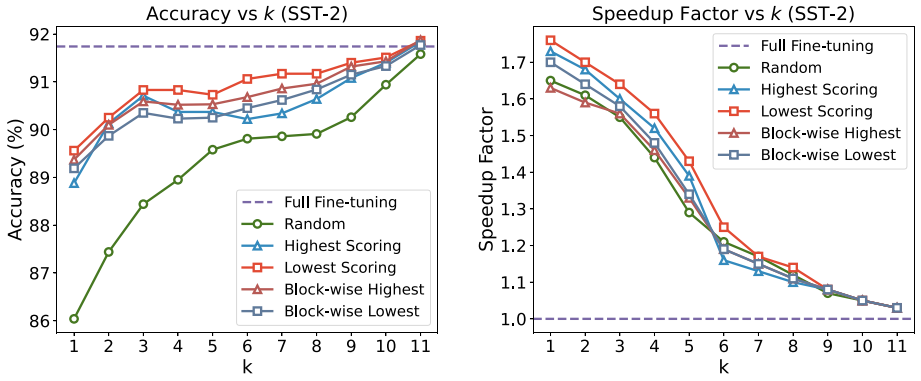

Fig. 4. Comparison of accuracy and speedup across k for the SST-2 dataset.

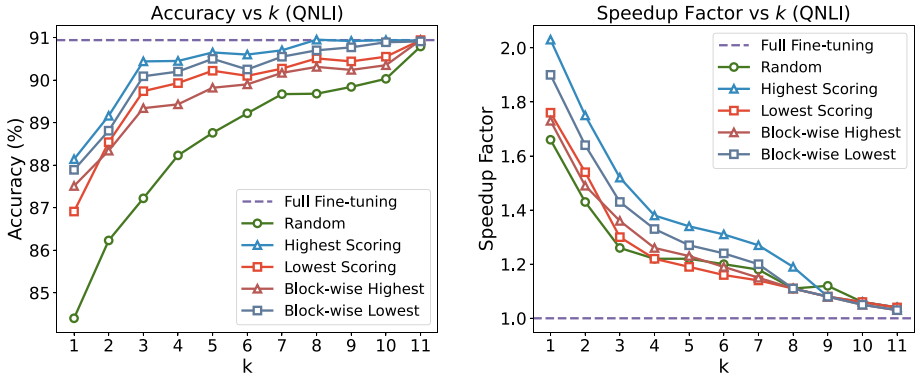

Fig. 5. Comparison of accuracy and speedup across k for the QNLI dataset.

training time, we observe a continuous decline in speedup as the value of k increases. Considering the trade-off between the accuracy and speedup, we choose $k = 3$ to be an optimal value for all of our experiments.

7.2. Performance across different LLMs

We also report our results for various LMs and LLMs, including RoBERTa-base, DeBERTa-base, ELECTRA-base, Gemma-2-2B [47], and TinyLlama-1.1B [48], in Table 6. We compare our method with random k, bottom k, and top k layer selection. As can be seen from the table, our method outperforms these baselines, achieving high performance across all tested LMs and proving to be consistent across various datasets. This highlights the broad applicability of our approach to different language models.

Scalability to Larger Models and Datasets. To explicitly evaluate the scalability of our approach, we tested billion-parameter models (Gemma-2-2B and TinyLlama-1.1B) as reported in Table 6. We observed that our method maintains its advantage over random and heuristic baselines even as model depth and width increase. Furthermore, our experiments on sentence-pair tasks (Table 2) encompass large-scale datasets, including MNLI (392k samples), QQP (363k samples), and SNLI (550k samples). The consistent performance gains observed on these large-scale benchmarks demonstrate that our training-free layer selection strategy is robust and effective not only for smaller datasets but also scales efficiently to resource-intensive training scenarios.

7.3. 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 the same experiment, but instead of cosine similarity, we use the Euclidean distance to calculate the similarity scores between the [CLS] tokens. The results in Table 7 show conclusive evidence, as using Euclidean distance instead of cosine similarity causes a notable drop in performance.

7.4. Comparison with alternative importance metrics

To further validate the effectiveness of using [CLS] token cosine similarity as a selection metric, we compared our approach against two established methods for quantifying layer importance:

Table 6
Comparison of performance across different language models. We use $k = 3$ for all layer selection methods. **Bold** indicates the best performance, and underline indicates the second-best.

Model	Method	RTE (acc)	MRPC (F1)	SST-2 (acc)	STS-B (Pear.)
RoBERTa	Random k layers	67.4	82.1	90.2	84.8
	Bottom k layers	68.3	82.9	91.4	85.3
	Top k layers	69.1	83.6	90.8	86.2
	High Scoring k layers	71.8	**85.1**	**92.2**	**88.4**
	Low Scoring k layers	**72.9**	84.6	92.0	87.3
DeBERTa	Random k layers	78.4	84.8	92.8	87.2
	Bottom k layers	79.4	85.2	93.4	88.1
	Top k layers	79.7	85.3	93.2	87.0
	High Scoring k layers	81.2	**87.4**	93.2	**89.7**
	Low Scoring k layers	**83.4**	86.2	**94.6**	88.5
ELECTRA	Random k layers	71.3	82.6	88.4	83.8
	Bottom k layers	72.0	83.8	87.3	85.6
	Top k layers	72.6	83.9	89.2	84.2
	High Scoring k layers	**74.7**	**86.3**	88.5	87.6
	Low Scoring k layers	74.0	85.8	**93.2**	**89.1**
Gemma-2-2B	Random k layers	74.4	80.2	85.8	80.3
	Bottom k layers	69.0	85.8	93.6	82.3
	Top k layers	75.8	82.3	90.0	85.8
	High Scoring k layers	**78.9**	**89.8**	95.3	**88.6**
	Low Scoring k layers	75.4	89.1	**96.8**	87.2
TinyLlama-1.1B	Random k layers	61.0	84.9	83.6	83.5
	Bottom k layers	59.9	81.5	56.4	84.9
	Top k layers	61.7	81.7	86.2	68.7
	High Scoring k layers	**63.8**	**86.4**	92.4	84.3
	Low Scoring k layers	62.5	85.7	**92.7**	**86.7**

Table 7
Comparison using Euclidean distance as the similarity metric. We use $k = 3$ for high scoring k and low scoring k layers.

	SST-2 (acc)	CoLA (acc)	QNLI (acc)	MRPC (F1)
Layer selection based on Euclidean distance				
High Scoring k layers	87.3	78.6	87.3	72.4
Low Scoring k layers	87.6	77.3	86.5	70.7
Layer selection based on cosine similarity				
High Scoring k layers	90.6	**81.0**	**90.2**	**75.0**
Low Scoring k layers	**90.8**	80.8	89.4	72.5

- **Gradient Norm-based Selection:** We compute the L_2 norm of the gradients for each layer over a single epoch of the training data. We then select the top-k layers with the highest gradient norms, under the assumption that layers with larger gradients are more sensitive to the task and thus more critical to fine-tune.
- **Feature Variance-based Selection:** We calculate the variance of the hidden state representations for each layer across the training set. We select the top-k layers with the highest variance, assuming that layers with higher statistical variability encode more discriminative features.

Table 8 presents the results on four diverse datasets. Our [CLS]-based strategies (specifically the best-performing strategy for each task) consistently outperform or match the Gradient Norm and Feature Variance baselines.

Notably, the Gradient Norm method, while competitive on QNLI (89.5%), underperforms on CoLA (78.8%) compared to our Block-wise Lowest strategy (81.2%). This suggests that the magnitude of updates (gradients) does not always correlate with the structural importance required for linguistic acceptability tasks. Furthermore, Gradient Norm selection requires a full backward pass, making it computationally more expensive than our forward-pass-only approach.

The Feature Variance method generally lags behind, particularly on MRPC (71.5 F1 vs 75.3 F1 for our method), indicating that raw statistical variance in hidden states is a noisier proxy for task relevance than the semantic trajectory captured by [CLS] cosine similarity.

Hence, our method not only achieves superior accuracy but does so with greater computational efficiency, as it is entirely training-free and does not require backpropagation for selection.

Table 8
Comparison of our [CLS] similarity-based selection against gradient norm and feature variance-based selection metrics. We use $k = 3$ for all methods. For our method, we report the result of the best-performing strategy (High/Low/Block) for that specific dataset.

Selection Metric	SST-2 (acc)	CoLA (acc)	QNLI (acc)	MRPC (F1)
Gradient Norm	90.1	78.8	89.5	73.9
Feature Variance	89.5	77.4	87.9	71.5
[CLS] Similarity (Ours)	**90.8**	**81.2**	**90.2**	**75.3**

Table 9
Ablation study on intra-block selection criteria. We compare selecting the highest and lowest scoring layers against median and random selection within blocks. We use $k = 3$ blocks for all experiments.

Intra-Block Selection Criterion	SST-2 (acc)	CoLA (acc)	QNLI (acc)	MRPC (F1)
Block-wise Random	89.9	79.9	88.5	70.2
Block-wise Median	90.1	80.1	88.8	70.8
Block-wise Highest (Ours)	**90.7**	81.1	**90.2**	**75.3**
Block-wise Lowest (Ours)	90.4	**81.2**	90.1	71.6

7.5. Influence of intra-block selection strategy

To investigate the necessity of selecting only the highest or lowest scoring layers within the block-wise setting, we experimented with two additional intra-block selection criteria:

- **Block-wise Median:** Selecting the layer with the median cosine similarity score within each defined block.
- **Block-wise Random:** Randomly selecting a single layer from within each defined block.

Table 9 compares these variations against our proposed Block-wise Highest and Block-wise Lowest strategies on four representative datasets (SST-2, CoLA, QNLI, and MRPC).

The results demonstrate that selecting the layers with extreme scores (either Highest or Lowest) consistently outperforms the Median and Random selection strategies within blocks. For instance, on CoLA, the Block-wise Lowest strategy achieves 81.2% accuracy, whereas Block-wise Median drops to 80.1%. Similarly, on QNLI, the Block-wise Highest strategy (90.2%) significantly outperforms Block-wise Median (88.8%).

This empirical evidence supports our hypothesis that the highest/lowest similarity scores serve as strong signals for layer utility. The *Highest* scoring layers act as critical information hubs (preserving stability), while the *Lowest* scoring layers represent areas of high semantic transformation (requiring adaptation). Intermediate (median) layers appear to lack this distinctiveness, resulting in suboptimal fine-tuning performance.

7.6. Influence of LoRA hyperparameters (r and α)

To ensure a comprehensive comparison with parameter-efficient baselines, we investigate the impact of the rank r and scaling factor α in LoRA. While our main experiments utilize $r = 16$ and $\alpha = 32$, we evaluate LoRA with varying ranks ($r \in \{8, 16, 32, 64\}$) while maintaining the ratio $\alpha = 2r$.

Table 10 presents the performance on four representative datasets. We observe that increasing the rank generally yields marginal performance improvements. However, even with $r = 64$, which significantly increases the number of trainable parameters, LoRA still consistently underperforms compared to our optimal layer selection strategies. This suggests that fine-tuning full, strategically selected transformer layers allows for more effective task adaptation than low-rank approximations applied across all layers, particularly for tasks requiring distinct semantic or structural adjustments.

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 studies. 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 generalization capability of our approach, enabling reliable predictions across varied tasks. We believe our method and findings contribute to advancing the efficient and effective utilization of language models in downstream applications.

Table 10
Comparison of LoRA performance across different rank (r) and alpha (α) values against our best-performing layer selection strategy for each dataset.

Method / Configuration	SST-2 (acc)	CoLA (acc)	QNLI (acc)	MRPC (F1)
LoRA ($r = 8, \alpha = 16$)	88.2	79.8	84.5	71.5
LoRA ($r = 16, \alpha = 32$)	88.6	80.3	84.9	72.1
LoRA ($r = 32, \alpha = 64$)	88.9	80.7	85.3	72.6
LoRA ($r = 64, \alpha = 128$)	89.1	80.9	85.5	73.0
Our Method (Best)	**90.8**	**81.2**	**90.2**	**75.3**

CRediT authorship contribution statement

Aldrin Kabya Biswas: Writing – original draft, Formal analysis, Data curation, Conceptualization. **Md Fahim:** Writing – review & editing, Formal analysis, Data curation. **Md Tahmid Hasan Fuad:** Validation, Methodology, Investigation. **Akm Moshiur Rahman Mazumder:** Writing – review & editing, Validation, Data curation. **Amin Ahsan Ali:** Methodology, Conceptualization. **AKM Mahbubur Rahman:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research is partially supported by Independent University, Bangladesh.

Data availability

All data are oublicly available and the links are provided in the manuscript.

References

[1] S. Arora, A. May, J. Zhang, C. Ré, Contextual embeddings: when are they worth IT? in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 2650–2663.
[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.
[3] J. Niu, W. Lu, G. Penn, Does BERT rediscover a classical NLP pipeline? in: Proceedings of the 29th International Conference on Computational Linguistics, 2022, pp. 3143–3153.
[4] M. Hosseini, M. Munia, L. Khan, BERT has more to offer: BERT layers combination yields better sentence embeddings, in: Findings of the Association for Computational Linguistics: EMNLP 2023, 2023, pp. 15419–15431.
[5] L.G.G. Charpentier, D. Samuel, Not all layers are equally as important: every layer counts BERT, in: Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning, 2023, pp. 238–252.
[6] A. Ardakani, A. Haan, S. Tan, D.T. Popovici, A. Cheung, C. Iancu, K. Sen, SlimFit: memory-efficient fine-tuning of transformer-based models using training dynamics, in: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024, pp. 6218–6236.
[7] L. Zhu, L. Hu, J. Lin, S. Han, LIFT: efficient layer-wise fine-tuning for large model models, 2024, https://openreview.net/forum?id=u0INlprg3U
[8] A. Simoulin, N. Park, X. Liu, G. Yang, Memory-efficient selective fine-tuning, in: Workshop on Efficient Systems for Foundation Models @ ICML2023, 2023.
[9] F. Ilhan, G. Su, S.F. Tekin, T. Huang, S. Hu, L. Liu, Resource-efficient transformer pruning for finetuning of large models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 16206–16215.
[10] S. He, G. Sun, Z. Shen, A. Li, What matters in transformers? not all attention is needed, arXiv preprint arXiv:2406.15786, 2024.
[11] J. Lee, R. Tang, J. Lin, What would elsa do? freezing layers during transformer fine-tuning, arXiv preprint arXiv:1911.03090, 2019.
[12] D. Ingle, R. Tripathi, A. Kumar, K. Patel, J. Vepa, Investigating the characteristics of a transformer in a few-shot setup: does freezing layers in RoBERTa help? in: Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, 2022, pp. 238–248.
[13] L. Xu, H. Xie, S.-Z.J. Qin, X. Tao, F.L. Wang, Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, arXiv preprint arXiv:2312.12148, 2023.
[14] A. Miaschi, F. Dell'Orletta, Contextual and non-contextual word embeddings: an in-depth linguistic investigation, in: Proceedings of the 5th Workshop on Representation Learning for NLP, 2020, pp. 110–119.
[15] R. Li, X. Zhao, M.-F. Moens, A brief overview of universal sentence representation methods: a linguistic view, ACM Comput. Surv. 55 (3) (2022) 1–42.
[16] F. Ding, C. Xu, H. Liu, B. Zhou, H. Zhou, Bridging pre-trained models to continual learning: a hypernetwork based framework with parameter-efficient fine-tuning techniques, Information Sciences 674 (2024) 120710.
[17] Z. Shen, Z. Liu, J. Qin, M. Savvides, K.-T. Cheng, Partial is better than all: revisiting fine-tuning strategy for few-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 9594–9602.
[18] D. Vucetic, M. Tayaranian, M. Ziaeefard, J.J. Clark, B.H. Meyer, W.J. Gross, Efficient fine-tuning of BERT models on the edge, in: 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022, pp. 1838–1842.
[19] R. Pan, X. Liu, S. Diao, R. Pi, J. Zhang, C. Han, T. Zhang, LISA: Layerwise importance sampling for memory-efficient large language model fine-tuning, arXiv preprint arXiv:2403.17919, 2024.
[20] J. Gu, A. Aleti, C. Chen, H. Zhang, A semantic-based layer freezing approach to efficient fine-tuning of language models, arXiv preprint arXiv:2406.11753, 2024.
[21] F. Cui, Y. Zhang, X. Wang, X. Tian, J. Yu, Enhancing target-unspecific tasks through a features matrix, in: Forty-Second International Conference on Machine Learning, 2025.

[22] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 1597–1607.
[23] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T.-Y. Liu, On layer normalization in the transformer architecture, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 10524–10533.
[24] X. Chen, Y. Hu, J. Zhang, Y. Wang, C. Li, H. Chen, Streamlining redundant layers to compress large language models, in: The Thirteenth International Conference on Learning Representations, 2025.
[25] G. Jawahar, B. Sagot, D. Seddah, What does BERT learn about the structure of language? in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 3651–3657.
[26] J. Jiang, J. Zhou, Z. Zhu, Tracing representation progression: analyzing and enhancing layer-wise similarity, in: The Thirteenth International Conference on Learning Representations, 2025.
[27] T. Gao, A. Fisch, D. Chen, Making pre-trained language models better few-shot learners, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 3816–3830.
[28] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. Bowman, GLUE: a multi-task benchmark and analysis platform for natural language understanding, in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, 2018, pp. 353–355.
[29] J. Novikova, O. Dušek, V. Rieser, The e2e dataset: new challenges for end-to-end generation, in: Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2017, arXiv:1706.09254.
[30] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: low-rank adaptation of large language models, in: International Conference on Learning Representations, 2022.
[31] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP, in: Proceedings of the 36th International Conference on Machine Learning, 2019.
[32] X.L. Li, P. Liang, Prefix-tuning: optimizing continuous prompts for generation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, 2021.
[33] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, C. Raffel, Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, in: Advances in Neural Information Processing Systems, vol. 35, 2022.
[34] U. Cohen, S. Chung, D.D. Lee, H. Sompolinsky, Separability and geometry of object manifolds in deep neural networks, Nat. Commun. 11 (1) (2020) 746.
[35] A. Ansuini, A. Laio, J.H. Macke, D. Zoccolan, Intrinsic dimension of data representations in deep neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).
[36] A. Merchant, E. Rahimtoroghi, E. Pavlick, I. Tenney, What happens to BERT embeddings during fine-tuning? in: Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, 2020, pp. 33–44.
[37] I. Tenney, D. Das, E. Pavlick, BERT rediscovers the classical NLP pipeline, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 4593–4601.
[38] A. Warstadt, A. Singh, S.R. Bowman, Neural network acceptability judgments, Trans. Assoc. Comput. Linguist. 7 (2019) 625–641.
[39] A. Rogers, O. Kovaleva, A. Rumshisky, A primer in BERTology: what we know about how BERT works, Trans. Assoc. Comput. Linguist. 8 (2020) 842–866.
[40] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, N. Smith, Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, arXiv preprint arXiv:2002.06305, 2020.
[41] M. Mosbach, M. Andriushchenko, D. Klakow, On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines, in: International Conference on Learning Representations, 2021.
[42] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, L. Li, On the sentence embeddings from pre-trained language models, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 9119–9130.
[43] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[44] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011, pp. 142–150.
[45] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the type and target of offensive posts in social media, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 1415–1420.
[46] B. Vidgen, T. Thrush, Z. Waseem, D. Kiela, Learning from the worst: dynamically generated datasets to improve online hate detection, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 1667–1682.
[47] M. Riviere, S. Pathak, P.G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé, et al., Gemma 2: Improving open language models at a practical size, arXiv preprint arXiv:2408.00118, 2024.
[48] P. Zhang, G. Zeng, T. Wang, W. Lu, Tinyllama: An open-source small language model, arXiv preprint arXiv:2401.02385, 2024.