

Article

A Resource-Efficient Approach to Fine-Tuning a BERT-Base Model for Sentiment Analysis

Abdullah M. Basahel ¹, Shreyanth H. Giryappa ², Furqan Alam ^{3,*} , Tahani Saleh Mohammed Alnazzawi ⁴, Saqib Qamar ^{3,*} and Adnan Ahmed Abi Sen ⁵

¹ Faculty of Economics and Administration, King Abdulaziz University, Jeddah 21589, Saudi Arabia; abasahl@kau.edu.sa

² School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool L3 3AF, UK; shreyanthhg1427@gmail.com

³ Faculty of Computing and Information Technology (FoCIT), Sohar University, Sohar 311, Oman

⁴ Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 41477, Saudi Arabia; tnazawi@taibahu.edu.sa

⁵ Hussein ElSayed Research Center, Deanship of Graduate Studies & Scientific Research, University of Prince Mugrin, Madinah 42241, Saudi Arabia; a.ahmed@upm.edu.sa

* Correspondence: falam@su.edu.om (F.A.); sqamar@su.edu.om (S.Q.)

Abstract

Fine-tuning a BERT-Base model for specific tasks, such as sentiment analysis, has become resource-intensive and often requires high computational power and memory. This paper introduces SCALE, a novel resource-efficient fine-tuning method that targets the most critical transformer layers, which reduces computational costs without sacrificing performance. By dynamically profiling transformer layers via activation magnitudes and attention entropy, SCALE selects and adapts only the most influential layers with lightweight adapter modules. The proposed method outperforms traditional fine-tuning techniques, achieving a 2.3% improvement in accuracy on the IMDB dataset and reducing training time by 56.3% compared to full-model fine-tuning. Experiments across various sentiment analysis benchmarks demonstrate SCALE's effectiveness in optimizing fine-tuning for the BERT-base model in resource-constrained environments, achieving up to 99% of the performance of full-model fine-tuning while using only 40% of the parameters. The empirical validation in this study is restricted to binary and multi-class sentiment classification. The evaluation specifically reflects effectiveness in sentiment analysis text classification tasks.

Keywords: sentiment analysis; low-rank adaptation; fine-tuning; BERT-base models



Academic Editor: Amit Kumar Mishra

Received: 3 January 2026

Revised: 15 February 2026

Accepted: 24 February 2026

Published: 3 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

BERT-base, like RoBERTa and DistilBERT, has a notable impact on natural language processing (NLP) by enabling machines to produce text that is similar to human text, intelligently respond to queries, translate languages, and more [1,2]. These models are essential components of modern AI systems, as they have demonstrated exceptional performance across a wide range of language tasks. However, optimizing these models for specific downstream tasks often requires substantial memory, high-end GPUs, and a long training period [3,4]. Small businesses, independent researchers, and students who might not have access to robust infrastructure are hampered by such requirements. Consequently, a significant amount of the international AI research community is still unable to take advantage of the tools provided by LLMs.

To address this immediate issue, this paper introduced a Selective Critical Adapter Layer Efficiency (SCALE) method to improve the resource efficiency of BERT customization. SCALE uses strategies motivated by recent advancements in parameter-efficient fine-tuning, such as Low-Rank Adaptation (LoRA) and Quantized Low-Rank Adaptation (QLoRA), which are adjusted to a model's most influential layers [5,6]. These techniques demonstrate that a small subset of parameters can be adjusted to maintain high performance while drastically reducing memory and computational requirements. SCALE is especially useful in low-resource environments, where enabling access to cutting-edge AI can be achieved by allowing inexpensive, task-specific BERT adaptation. This gives more power to researchers, students, and practitioners from diverse socioeconomic backgrounds to innovate without requiring supercomputing infrastructure, thereby promoting educational equity and inclusive technological advancement.

This research aims to reduce the computational cost of fine-tuning during BERT training. The strategy improves access to powerful AI tools by reducing processing requirements and time, thereby facilitating the adaptation of BERT to support domain-specific tasks such as sentiment analysis and query response. The objectives of this work are as follows:

- To develop a lightweight fine-tuning method that exploits activation-based scoring by selectively updating the most dominant components of a language model to secure efficient and targeted adaptation.
- To minimize computational requirements by restricting parameter updates to high-impact components, facilitating fine-tuning in low-resource environments.
- To improve the scalability of the fine-tuning process and adapt to hardware capacities.
- To enhance model interpretability by identifying and updating layers that are most influential for task performance based on internal model signals.

While prior PEFT methods, to our knowledge, do not employ a selective activation-based approach for fine-tuning BERT, SCALE targets only the most critical layers instead of performing full-model or uniform adaptation. Although parameter-efficient fine-tuning methods are typically described as task-agnostic, the present study evaluates SCALE only within sentiment classification using text classification datasets. The approach is also empirically validated only on encoder-only transformer architectures (BERT). All experiments are conducted on text classification datasets, and the approach's effectiveness for other NLP problem categories, including sequence labeling, structured prediction, and generative question answering, is outside the empirical scope of this study.

The paper is organized into eight sections. Section 2 presents a critical literature review related to the BERT-base model and its fine-tuning. Section 3 describes the experimental setup and datasets. Section 4 provides a comprehensive discussion of the proposed method. Section 5 reports the results and analysis, while Section 6 is on the Ablation study. Section 7 offers a critical discussion. Finally, Section 8 concludes the paper and outlines future research directions.

2. Literature Survey

The application domain of BERT-base models has grown significantly. Their architectures have become much more complex and advanced as a result, which has a direct impact on how efficiently they use time and resources. To make these BERT models efficient and accessible, there is an increasing need for novel fine-tuning strategies. Due to substantial computational costs, traditional full-model fine-tuning is not feasible for most users or domain-specific applications such as sentiment analysis and technical question answering. For sustainable development, BERT based on parameter-efficient techniques

such as quantization, LoRA, and adapter tuning will be important. This section critically examines the strategies that highlight the resource-conscious and targeted approaches.

Hu et al. [7] introduced low-rank adapters for each layer to implement LoRA for fine-tuning. But this leads to inefficient calculation. By carefully placing adapters only in the most critical layers, this study also increases efficiency by reducing memory consumption and processing time. Similarly, Zhang et al. [8] highlighted that AdaLoRA improves LoRA by dynamically adjusting low-rank adapter sizes based on parameter importance. To maximize the computational efficiency, reallocation and pruning techniques can be implemented. It is ideal for fine-tuning scenarios with limited resources because it outperforms standard LoRA under stringent training budgets. By utilizing 4-bit quantization and adapters, Dettmers et al. [6] created QLoRA, enabling extensive model tuning on constrained hardware. Yes, it still applies changes to all layers. Without relying entirely on quantization, this study offers improved computational efficiency and lower energy consumption by selectively tuning only high-activation layers. Houlsby et al. [9] proposed an adapter module for NLP, a lightweight network integrated into transformer layers. It enables minimal parameter changes and is effective in multitask learning. Reduced the cost of fine-tuning by training only 3–4% of the parameters per task while demonstrating outstanding performance across multiple benchmarks. Ben Zaken et al. [10] introduced the BitFit approach, a straightforward technique that adjusts only the bias terms in transformer layers and complements these strategies. It performs well on NLP benchmarks such as GLUE. Also, it provides a quick, portable, and surprisingly powerful method for fine-tuning parameters with little resource commitment. Ruckle et al. [11] proposed AdapterDrop, which randomly eliminates adapters during training to further reduce costs. Randomly ignoring leads risks omitting crucial layers. To ensure that only necessary layers are trained and maintain performance while reducing overhead, this study replaces intelligent layer scoring with random behavior. DiffPrune, a sparse fine-tuning technique that uses binary gates and L_0 regularization to prune parameter updates, was proposed by Shulman et al. [12]. Efficient models learn which weights to update, enabling lightweight training. This shows a significant reduction in parameters while maintaining comparable model performance. Delta tuning was proposed by Ding et al. [13] as a minimal parameter update technique. It provides theoretical insights and extensive empirical benchmarks by classifying methods such as prompts, adapters, and sparse tuning. Hence, this helps model designers select the best PEFT techniques for a variety of BERT tasks and domains.

He et al. [14] SensiLoRARAG integrates chain-of-thought reasoning for domain-specific BERT with sensitivity-based LoRA. It also integrates retrieval-based augmentation and dynamically selects LoRA ranks. It achieves strong performance in medical and legal QA with minimal supervision and enhanced contextual reasoning. Mao et al. [15] provide an extensive survey of LoRA and its variants. The study categorizes research based on adaptation efficiency, cross-task generalization, deployment contexts, and privacy considerations and highlights LoRA's scalability, limitations, and the evolution of low-rank fine-tuning in NLP and multimodal applications. Prottasha et al. [16] introduced Semantic Knowledge Tuning (SK-Tuning), which uses meaningful token prompts for efficient adaptation, extracts semantic representations using frozen BERT and fuses them with input embeddings, and outperforms prefix tuning in NLP tasks with fewer parameters and faster convergence. Wang et al. [17] surveyed parameter-efficient fine-tuning (PEFT) approaches, including LoRA, BitFit, prefix tuning, and delta methods. The study offers a taxonomy of methods, design choices, and empirical comparisons and identifies future directions for scalable, personalized, and privacy-preserving BERT adaptation strategies. Lastly, Zhou et al. [18] applied adapter-based tuning for Chinese medical named entity recognition. It inserts adapters in key transformer layers and combines them with learned prefix em-

beddings, achieving new state-of-the-art results while reducing trainable parameters and mitigating catastrophic forgetting. The use of BERT for sentiment analysis is increasing rapidly due to its ability to better understand context, tone, and nuanced language than deep learning methods.

Additionally, it has strong capabilities for working with multiple languages and processing large amounts of textual data, making it a perfect fit for enabling deeper emotional and opinion-based insights. Wang et al. [19] proposed a Lo-based fine-tuning approach for DeBERTa and RoBERTa models for empathy and emotion classification on social media and showed improved performance with back-translation, illustrating the usefulness of LoRA in sentiment tasks from the real world.

Building on the significance of domain-specific adaptation, Barreto et al. [20] conducted a thorough evaluation of sentiment classifiers on Twitter, comparing transformer-based and conventional models. It was discovered that contextualized models, such as BERTweet, perform significantly better than others by demonstrating the need for domain-specific adaptation. It was also found that contextualized models like BERTweet significantly outperform others, highlighting the need for domain-specific adaptation. Shen et al. [21] presented ORPO, a fine-tuning framework that combines supervised learning and alignment objectives in a single step to further the efficiency of sentiment classification. BERT is used to demonstrate increased accuracy and decreased computational cost in multi-class sentiment analysis tasks.

Addressing multilingual contexts, Đorević et al. [22] applied XLM-RoBERTa and sentiment lexicon fusion for multilingual sentiment classification. They achieved high performance in low-resource settings using cross-lingual transfer and feature combination strategies for multi-class tasks. In terms of optimization, Zhan et al. [23] analyzed LoRA and QLoRA fine-tuning strategies for sentiment analysis on LLMs. They showed that QLoRA maintains comparable accuracy while significantly reducing memory usage, validating its efficiency for deployment. In contrast, Pavlyshenko [24] evaluated and compared LoRA, AdaLoRA, and QLoRA on Twitter sentiment datasets. They found AdaLoRA superior in terms of accuracy and parameter efficiency thanks to its adaptive rank allocation, making it ideal for social media applications.

In this work, LoRA [7], AdaLoRA [8], and BitFit [10] are selected as direct experimental baselines because they can be implemented under identical training conditions and hardware constraints. Other approaches, such as Prefix-Tuning, AdapterDrop, and DiffPrune, are discussed for context but are not included in the quantitative comparison due to differences in training procedures and architectural requirements.

The literature reviewed above on existing fine-tuning methods offers various efficiency gains and an increasing use of BERT for sentiment analysis. However, most of them lack mechanisms to prioritize high-impact layers. As a result, there are redundant updates and a higher computational cost.

As per the scope and positioning of this work, the existing parameter-efficient fine-tuning methods are commonly evaluated across heterogeneous NLP tasks. However, model behavior differs substantially between classification, sequence labeling, and generative settings. Accordingly, this study restricts evaluation to sentiment classification, where supervision is explicit and measurable. The contribution of this work is therefore not a universal benchmark of PEFT methods, but rather a task-specific investigation of layer-sensitivity-guided adaptation for sentiment analysis.

3. Experimental Setup & Datasets

To perform experiments and validate the proposed method, the system uses the RunPod.io infrastructure. This cloud-based platform offers affordable access to high-

performance GPUs for Transformer-based model training. The experimental environment runs on a Linux-based system (Ubuntu 20.04.1) with an NVIDIA RTX A40 GPU. CUDA support is enabled, ensuring efficient GPU acceleration during training. Based on the run-pod image, the implementation is deployed in a container-based environment using Python 3.11.11. PyTorch 2.8.0 with CUDA 12.8.1 and cuDNN is integrated, which provides the configuration. This system configuration provides a scalable, controlled, and reproducible environment for testing large-scale sentiment analysis tasks.

All baseline methods (LoRA [7], AdaLoRA [8], BitFit [10], and SCALE) were trained using the same backbone model, tokenizer, optimizer, batch size, and hardware configuration to ensure a controlled comparison of parameter efficiency and performance. All ablation experiments were also conducted under identical environment and training conditions.

Understanding sentiment across a variety of text sources, including tweets, reviews, and movies, has become essential for real-world AI as digital expression has grown. To capture this diversity, we selected multiple datasets as mentioned in Table 1. They vary in sentiment granularity, length, and domain. The IMDb [25] Movie Reviews Dataset is appropriate for evaluating sentiment models on lengthy textual input, as it includes long-form reviews classified as either positive or negative. The Stanford Sentiment Treebank v2 (SSTV2) [26] focuses on short-form sentiment analysis and includes single-sentence reviews with binary sentiment annotations. The Yelp Polarity Reviews Dataset [27], which is a subset of the Yelp Open Dataset, isolates reviews that are neutral and concentrates on opinions that are obviously divided; therefore, it is perfect for binary sentiment modeling. A realistic challenge for social media analysis is provided by the TweetEval [28] Sentiment Dataset, which consists of brief, informal tweets classified into three classes: positive, neutral, and hostile. The TwExportly Chrome extension was used to create the XLive Dataset, which allows for real-time tweet extraction. TextBlob allocates polarity scores between -1 and $+1$. It was also used to annotate sentiment because this dataset lacked pre-existing labels. The XLive dataset is weakly supervised. Sentiment labels were generated automatically using the TextBlob polarity estimator rather than manual human annotation. Consequently, the labels represent heuristic sentiment approximations rather than verified ground truth. Therefore, results on XLive should be interpreted as performance under noisy pseudo-label supervision rather than as a direct measurement of sentiment understanding. Five sentiment classes, Negative, Less Negative, Neutral, Less Positive, and Positive, were applied to tweets, which allows more precise sentiment classification in dynamic, real-world situations.

Table 1. Details of the Sentiment Analysis Datasets.

No.	Dataset Name	Sentiment Classes	Task Type	Dataset Size
1	IMDb Movie Reviews	0: Negative, 1: Positive	Binary Class	50,000
2	Stanford Sentiment Treebank v2 (SSTV-2)	0: Negative, 1: Positive	Binary Class	87,552
3	Yelp Polarity Reviews	0: Negative, 1: Positive	Binary Class	55,000
4	TweetEval Sentiment	0: Negative, 1: Neutral, 2: Positive	Multi-class Class	75,265
5	X-Twitter Live Dataset	0: Negative, 1: Less Negative, 2: Neutral, 3: Less Positive, 4: Positive	Multi-class Class	18,018

4. Proposed Methodology

A key component of customizing natural language models for specific downstream tasks is fine-tuning models such as BERT, RoBERTa, and DistilBERT. In this paper, SCALE refers to the proposed selective adaptation framework, while SCALE-BERT denotes its specific implementation on the BERT-base encoder used in the experiments. Despite their

strength, these models have high memory and processing demands. Conventional fine-tuning methods require substantial resources and update all the transformer network's parameters, making them inappropriate for large-scale deployment or low-resource environments. To address this, we proposed a Selective Critical Adapter Layer Efficiency (SCALE) method that identifies and modifies only the most critical layers of the transformer architecture rather than using a uniform or thorough fine-tuning approach. This is accomplished by applying small, low-rank, trainable adapter modules to only the most pertinent layers and by context-sensitively profiling the transformer's internals. Without reducing the model's overall performance, this leads to significant reductions in computational costs, faster training, less overfitting, and more effective inference. Figure 1 shows the working of SCALE for sentiment analysis.

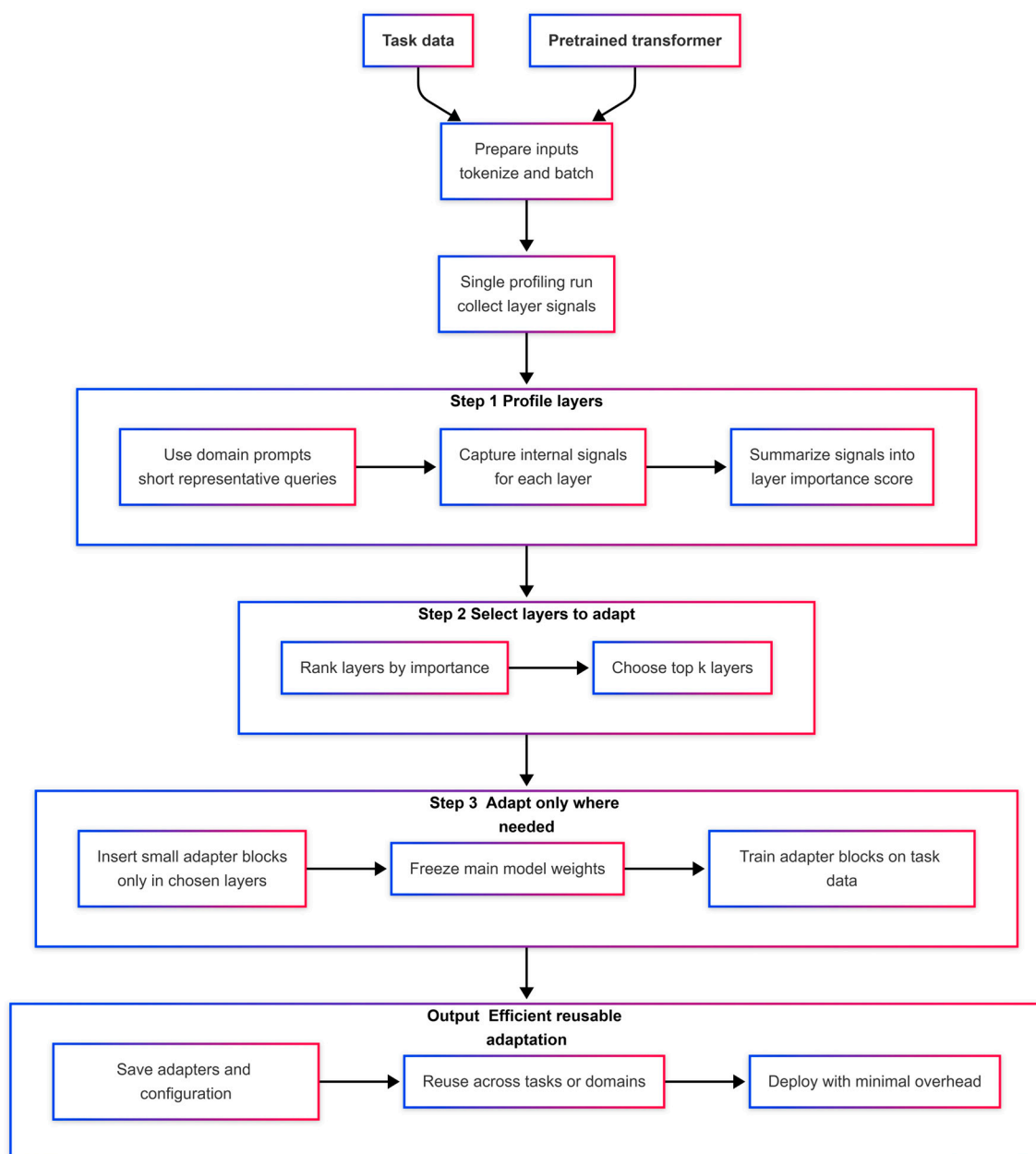


Figure 1. Illustration of the workings of the SCALE approach. First, it prepares the task data; Step 1 profiles the layers by capturing and ranking internal signals; Step 2 selects the top layers to adapt; Step 3 adapts only the necessary layers by inserting small adapter blocks; and Step 4 ensures reusable adaptation for future tasks with minimal deployment overhead.

Modern transformers have a large number of parameters, making conventional fine-tuning computationally demanding. This problem is reduced using Parameter Efficient Fine-Tuning (PEFT) techniques such as Low Rank Adaptation [7], AdaLora [8], prefix tuning, and lightweight adapters. However, they frequently lack adaptability. These approaches usually ignore the reality that not all layers contribute equally to downstream task performance by distributing parameter changes evenly across all layers. To solve this, SCALE proposes a smart and flexible method that examines the internal dynamics of each layer. Activation Magnitude (AM) and Attention Entropy (EA) are combined by SCALE to generate a composite score that assesses a layer's utility for a particular task. Adapter injection is then used to update only the top k-scoring layers.

This focused design increases training stability, improves interpretability, and reduces computational overhead by restricting modifications to the model's most significant areas. Our methodology comprises six distinct steps.

4.1. Preprocessing and Transformer Initialization

The SCALE procedure is applied and validated only for classification-style objectives where the model output corresponds to discrete label prediction. To ensure everything is configured effectively for profiling, SCALE first builds the input data for modeling before fine-tuning can begin. The 'bert-base-uncased' model from Hugging Face serves as the encoder, which is the only backbone in our experiments. However, SCALE supports encoder-decoder models, such as DistilBERT and BART, and decoder-only models, such as RoBERTa. Padding, truncation, and establishing a maximum sequence length are all part of tokenization. The system is guaranteed to be fully prepared for the next stage as a result of this preprocessing. Now that the inputs have been formatted and the model initialized, we can move on to the main innovation of SCALE: dynamic profiling. The next stage determines which transformer layers are actually worth modifying, making the way for more targeted and effective fine-tuning.

4.2. Dynamic Layer Profiling via Fast Transformer Activation Profiler

We provide a novel profiling technique that dynamically identifies Activation magnitude and attention entropy as behavioral indicators of contribution to the training objective rather than direct measurements of semantic representation. The profiling stage identifies layers that most influence sentiment classification performance. The method does not infer whether these layers encode high-level linguistic semantics. SCALE's primary innovation is its ability to go beyond uniform adaptation strategies, such as those used in LoRA [7]. Activation Magnitude (AM), which measures how strongly a layer activates, and Attention Entropy (EA), which measures how strongly a layer responds and how focused its attention is, are two internal signals that SCALE uses to evaluate each layer's contribution to the task rather than randomly inserting adapters across all layers. This provides a reliable method of estimating a layer's impact. The profiling stage is executed once before fine-tuning to estimate layer importance. It consists of a single forward pass over the training set without parameter updates and is therefore not repeated during optimization. In practice, its runtime is negligible compared with training. For example, full fine-tuning requires 17.70–31.95 min, depending on dataset complexity, whereas the profiling pass corresponds approximately to one inference pass and accounts for only a small fraction of the total training time. Consequently, profiling does not affect inference cost and introduces only a minor preprocessing overhead.

SCALE rank layers only retain the top k most useful ones by combining AM and EA importance scores. This maintains task-specific performance and interpretability while drastically reducing resource usage. However, without labeled training data, how can these

internal signals be effectively triggered? The solution lies in the next step, prompt-based evaluation, a crucial element that enables quick, accurate profiling of the model's behavior.

4.2.1. Prompt-Based Evaluation

In this step, we demonstrate a novel method that uses a sequence of preparation stages to activate transformer layers with domain-specific prompts precisely. The model can “react” internally based on its pretraining because the unlabeled prompts are related to the target task. These inputs elicit stronger responses from layers that have retained important pretrained information, thereby revealing the model's location of task-relevant knowledge. As stated in Section 4, this internal response is recorded using two crucial signals: Attention Entropy (*EA*) and Activation Magnitude (*AM*), which indicate how concentrated or dispersed the attention is across the tokens. The SCALE profiler effectively extracts the signals from the model by utilizing these simple prompts. As the foundation for our quantitative layer scoring mechanism, we now specifically define how *AM* and *EA* are calculated for each layer in the following step.

4.2.2. Layer-Wise Metric Computation

This step describes an organized approach to examine and modify each layer of the transformer architecture. Activation Magnitude (*AM*) measures the average activation strength to identify impactful transformer-layer behavior. In this step, the activation magnitude for a specific transformer layer i is calculated. The symbol AM_i refers to the average absolute value of all activations (hidden states) in that layer. The tensor represents the hidden state of each token, H_i , which has dimensions $B \times T \times d$. Here, B stands for the batch size (number of input samples), T is the sequence length (number of tokens in each input), and d is the hidden dimension (size of each token embedding). The triple summation over b , t , and j iterates over all batches, all tokens in the sequence, and all features in the embedding dimension, respectively. The absolute values of each hidden unit are summed and normalized by the total number of elements $B \cdot T \cdot d$, yielding the average activation level of the layer. The tensor represents the hidden state of each token. In the next equation, we understand Attention Entropy.

$$AM_i = \frac{1}{B \cdot T \cdot d} \sum_{b=1}^B \sum_{t=1}^T \sum_{j=1}^d |H_i[b, t, j]| \quad (1)$$

Attention Entropy (*EA*) quantifies the uncertainty or dispersion in a transformer's attention distribution. Lower entropy indicates that attention is concentrated; i.e., the model is confident and focused on specific tokens, whereas higher entropy implies that attention is spread across many tokens, reflecting uncertainty. To compute *EA* for a given transformer layer i , we first average the raw attention weights across all attention heads. Let A_i be the raw self-attention weight tensor produced by the i^{th} transformer layer, such that $A_i \in \mathbb{R}^{B \times H \times T \times T}$. The batch size, number of attention heads, and sequence length are denoted by B , H , and T , respectively. Each head provides a $T \times T$ attention matrix for each sample in the batch, A_i is the self-attention tensor of layer i , and P_i is the head-averaged attention matrix.

$$P_i[b, :, :] = \frac{1}{H} \sum_{h=1}^H A_i[b, h, :, :] \quad (2)$$

This averaging process gives a single attention distribution per token per sample, aggregating the perspectives of all attention heads. This lays the foundation for computing token-level entropy in the next step.

Compute the token-level calculation using the Shannon entropy of attention distributions. The term $P_i[b, t, k]$ denotes the attention probability from token t to token k in sample b after averaging across heads. The entropy is computed by summing over all tokens k , using the formula $-P \cdot \log(P + \epsilon)$. The constant ϵ , typically a minimal value such as 10^{-9} , ensures numerical stability by preventing the logarithm of zero. The result, $H(P_i[b, t, :])$, reflects how uncertain or spread out the attention distribution is for token t . It sets the stage for token-wise entropy computation using averaged attention distributions.

$$H(P_i[b, t, :]) = - \sum_{k=1}^T P_i[b, t, k] \cdot \log(P_i[b, t, k] + \epsilon) \quad (3)$$

To compute average attention entropy, we compute the layer-level attention entropy. It averages the token-level entropies from Equation (3) across all tokens T and all samples B . The resulting value, EA_i , reflects the overall uncertainty in the attention mechanism for layer i . Lower EA_i means the layer has focused attention (high confidence), while a higher EA_i suggests the attention is diffused (low confidence).

$$EA_i = \frac{1}{B \cdot T} \sum_{b=1}^B \sum_{t=1}^T H(P_i[b, t, :]) \quad (4)$$

Next, we combine activation and entropy metrics to derive a unified importance score. For composite scoring functions, we propose a novel function that fuses amplitude and entropy signals for precise layer importance ranking, adaptable across transformer families. The final importance score for each layer is computed as a weighted combination of activation and entropy signals. In this novel formula, S_i is calculated by combining the activation magnitude and the complement of attention entropy, where α and β are hyperparameters that weight the importance of activation and attention, respectively. $(1 - EA_i)$ implicitly assumes that EA_i is normalized to $[0, 1]$, and S_i indicates that the layer is both highly active and has focused attention, making it a strong candidate for fine-tuning.

$$S_i = \alpha \cdot AM_i + \beta \cdot (1 - EA_i) \quad (5)$$

Here, α and β are hyperparameters controlling the influence of each metric, typically set to 1.0. In the next step, we rank layers and filter the top- k based on their scores. Top- k Layer Selection selects the most critical layers to optimize adaptation and reduce computation. Once scores $\{S_1, \dots, S_N\}$ are obtained, SCALE selects the top- k layers.

$$L_k = \text{TopK}(\{(L_i, S_i)\}_{i=1}^N, k) \quad (6)$$

We introduce a top- k selection strategy, denoted as Equation (6), where L_i represents the i th transformer layer, S_i is its composite importance score (computed using activation magnitude and attention entropy), N is the total number of layers, and the TopK operator selects the indices of the top- k layers with the highest scores. This strategy uses entropy-activation profiling to rank layers and selects the top k high-salience layers most relevant for downstream adaptation, effectively reducing computation by skipping less critical layers. In the next section, we describe how to inject a lightweight adapter module across selected transformer architectural layers.

To justify the selection of the hyperparameters α and β in Equation (5), we conducted a sensitivity study on the IMDB sentiment classification dataset. This analysis evaluates whether the importance ranking S_i depends critically on a specific ratio of Activation Magnitude (AM) to Attention Entropy (EA). We tested five configurations spanning from AM-dominant to EA-dominant weighting schemes, as mentioned in Table 2:

Table 2. Sensitivity Analysis of Weighting Coefficients on IMDB Dataset.

$\alpha:\beta$	Strategy	Accuracy (Mean $\pm$ Std)	F1-Score (Mean $\pm$ Std)
2.0:0.0	AM-only	0.879 $\pm$ 0.0060	0.879 $\pm$ 0.0061
1.5:0.5	High-AM	0.882 $\pm$ 0.0070	0.882 $\pm$ 0.0070
1.0:1.0	Balanced	0.880 $\pm$ 0.0065	0.880 $\pm$ 0.0065
0.5:1.5	High-EA	0.880 $\pm$ 0.0065	0.880 $\pm$ 0.0065
0.0:2.0	EA-only	0.880 $\pm$ 0.0065	0.880 $\pm$ 0.0065

The results indicate that model performance remains remarkably stable regardless of the coefficient selection. Accuracy varied only within a narrow range of 0.879 to 0.882, while the F1-score exhibited a similarly negligible variance. The low standard deviations across all five-fold iterations confirm that SCALE's training behavior is robust regarding the relative weighting of internal signals. The coefficients α and β serve as stable selection mechanisms rather than sensitive hyperparameters, allowing the use of fixed values (typically 1.0) without introducing performance bias.

This analysis confirms that the proposed AM-EA criterion reliably identifies task-relevant layers for sentiment classification without requiring exhaustive hyperparameter optimization.

4.3. Selective Adapter Injection

After identifying high-impact layers, we propose a novel approach to selectively injecting lightweight adapter modules into those selected top layers. This deviates from conventional full-model tuning or uniform adapter placement by achieving both parameter efficiency and interpretability. This stage introduces task-specific adaptability to the transformer model by inserting lightweight trainable modules, or adapters, into the influential layers. The next step involves defining the internal architectures and mathematical equations of the selective adapter modules used in SCALE to execute this strategy.

Adapter Architecture and Mathematical Formulation

Based on the relevance of influential layers, we propose a unique residual adapter framework that is selectively introduced into top- k layers. This method enables targeted adaptation while preserving pretrained representations. Let $x \in R^d$ be the input to a transformer sub-layer. The adapter function $A(x)$ is defined as a bottleneck architecture with the following components; Equations (7)–(10) describing the normalization, projection, and residual operations.

Layer Normalization stabilizes and standardizes the input before projection, ensuring smooth gradient flow and faster convergence. Before injecting the adapter module, we take the input $x \in R^d$ and input token embedding, where d is the hidden dimension, z is the normalized representation, and LayerNorm ($\cdot$) is the standard layer normalization operation. The output of this operation ensures stable training and prevents internal covariate shift. Next, we reduce the normalized vector to a compact form using down-projection.

$$z = \text{LayerNorm}(x) \quad (7)$$

Down-Projection reduces the input dimensionality to a smaller bottleneck d_a , significantly lowering the number of trainable parameters. Equation (8) defines the down-projection step of the adapter module; $W_d \in R^{d_a \times d}$ is the down-projection weight matrix, $b_d \in R^{d_a}$ is the bias term, and $\sigma(\cdot)$ denotes a non-linear activation function applied to the affine transformation $W_d z + b_d$, where W_d projects the normalized input z into a lower-dimensional bottleneck space and b_d is the corresponding bias term. Now we restore the representation to the original dimension using an Up-Projection layer. Up-Projection

restores the reduced representation to its original dimension d , enabling it to integrate seamlessly with the main model.

$$h = \sigma(W_d z + b_d), \quad W_d \in \mathbb{R}^{d_a \times d}, \quad b_d \in \mathbb{R}^{d_a} \quad (8)$$

In Equation (9), $\hat{x}$ is the reconstructed output representation. The variable d refers to the original hidden size of the transformer layer, while d_a is the bottleneck dimension of the adapter, typically much smaller than d . The matrix $W_u \in \mathbb{R}^{d \times d_a}$ denotes the up-projection weight matrix that maps the low-dimensional vector h back to the original dimension. The term $b_u \in \mathbb{R}^d$ is the bias associated with this transformation. The result of this projection is the reconstructed representation $\hat{x} \in \mathbb{R}^d$, which aligns with the transformer's native hidden space. To integrate with the transformer, we add a residual connection on top of this projection in the next step.

$$\hat{x} = W_u h + b_u, \quad W_u \in \mathbb{R}^{d \times d_a}, \quad b_u \in \mathbb{R}^d \quad (9)$$

Residual Connection preserves the original transformer output while allowing the adapter to efficiently inject task-specific information. In this context, Equation (10) is used, where $\hat{x}$ is the final output of the adapter module and has the same dimensionality as the transformer's hidden state. The function $L_i(x)$ represents the output of the i^{th} transformer layer before adapter injection. After the adapter output is added, the modified layer output is denoted as $L'_i(x)$. The adapter function is written as $A(x)$, and, in this case, it is defined to be equal to $\hat{x}$. In the next step, we quantify the parameter complexity of these adapters to analyze their memory and computational efficiency.

$$A(x) = \hat{x}, \quad L'_i(x) = L_i(x) + A(x) \quad (10)$$

Parameterization and Complexity: We propose a novel parameter budgeting formula to quantify the memory footprint of selective adapter injection, allowing users to control model size and training cost analytically. The number of trainable parameters in a single adapter block is:

$$\text{Params}_{\text{adapter}} = 2d \cdot d_a + d + d_a \quad (11)$$

In Equation (11), d stands for the transformer's original hidden size, and d_a is the bottleneck size used in the adapter module. The expression $2d \cdot d_a$ accounts for the number of trainable parameters in the two linear projection matrices; W_d for down-projection and W_u for up-projection. The additional term $d + d_a$ comes from the biases b_d and b_u associated with each projection. Together, these define the complete parameter count for one adapter block.

$$\phi = \bigcup_{i=1}^k \{W_d^{(i)}, W_u^{(i)}, b_d^{(i)}, b_u^{(i)}\} \quad (12)$$

In the above Equation (12), the complete set of trainable parameters ϕ spans all selected transformer layers. Here, k denotes the number of layers where adapters are injected. For each layer i , the set includes the down-projection weight $W_d^{(i)}$, up-projection weight $W_u^{(i)}$, and their respective biases $b_d^{(i)}$ and $b_u^{(i)}$. The union over $i = 1 \dots k$ aggregates all adapter components into the global parameter set ϕ , which is used for optimization during training.

In the next step, Equation (13), we introduce the training objective that updates only the adapter parameters while keeping the rest of the model frozen.

Training Objective; We propose a novel fine-tuning objective that isolates adaptation to injected adapter modules, freezing the rest of the model and enabling efficient specialization without catastrophic forgetting.

$$\min_{\phi} \mathcal{L}(\phi) = - \sum_{j=1}^M y_j^T \log f_{\theta, \phi}(x_j) \quad (13)$$

This equation defines the training objective used to fine-tune the adapters. To minimize the loss $\mathcal{L}$, update only the adapter parameters ϕ . The base model parameters θ are kept frozen for each training sample x_j ; the model produces a softmax-normalized class probability vector output $f_{\theta, \phi}(x_j)$, which is compared to the true label y_j using the cross-entropy loss to pick the log-probability. Here, M is the total number of training samples. The above objective guarantees that the modifications are effective and lightweight without changing the pretrained core model. In the next step, we evaluate how this injection mechanism generalizes across different transformer architectures.

4.4. K-Fold 5 Cross-Validation Strategy Was Employed

To ensure robust performance estimation, we fine-tuned BERT-base models with cross-validation. Because cross-validation is expensive and not a standard practice, we used K-fold cross-validation on the test set solely to ensure stability and consistency of trained BERT models. Instead of using it during training, each of the five test folds was passed through the saved models for LoRA [7], AdaLoRA [8], BitFit [10], and SCALE. At every fold, model weights remained fixed, and no hyperparameter tuning was performed during testing. This enabled us to assess the stability and generalization of predictions across diverse test subsets. The evaluation was conducted on three binary classification datasets, IMDB, Stanford Sentiment Treebank (SSTV-2), and Yelp, and two multi-class classification datasets, XLive and Twitter, providing a comprehensive view of each model's performance under distributional shifts. The average performance across the five folds was considered as the final result for all three models. With model performance validated, we now explain how the final model is serialized, structured, and prepared for deployment, as discussed in the next section.

4.5. Model Serialization and Inference Deployment

Once fine-tuning is complete, SCALE enters the serialization and deployment phase, packaging the model with selective adapters (Equations (11) and (12)) for efficient inference and reuse. It stores the model weights with adapters, configuration details, including architecture and adapter metadata, and tokenizer files to ensure consistent text processing. This process ensures platform compatibility and streamlines model loading. The lightweight design allows fast, portable deployment while preserving adapter flexibility for future fine-tuning or domain adaptation without retraining the whole model.. The complete execution flow of the SCALE framework, integrating dynamic profiling, selection, and adapter injection, is summarized in Algorithm 1. In the next section, we discuss the results of our novel methodology

Algorithm 1: Proposed SCALE Method

Require: Pretrained transformer model $\mathcal{M}$ with N layers, downstream task data $\mathcal{D}$

Ensure: Model with selective adapter injection

1. **Step 1: Preprocessing**
2. Prepare dataset $\mathcal{D}$ and model $\mathcal{M}$
3. Tokenize input data with padding and truncation
4. Convert to tensors: input_ids, attention_mask
5. Move $\mathcal{M}$ and data to GPU
6. **Step 2: Layer Profiling**
7. Identify high-impact transformer layers using activation-entropy profiling.
8. Use domain-specific prompt set ($\mathcal{P}$)
9. **For each layer $i = 1, \dots, N$ do**
 Compute hidden states $H_i \in R^{B \times T \times d}$, where B : batch size,
 T : sequence length, d : hidden dimension
 Compute Attention $A_i \in R^{B \times H \times T \times T}$, where H : number of heads
10. Activation Magnitude as given in Equation (1):

$$AM_i = \frac{1}{B \cdot T \cdot d} \sum_{b=1}^B \sum_{t=1}^T \sum_{j=1}^d |H_i[b, t, j]|$$
11. Head-Averaged Token-Level Attention Entropy (Equations (2)–(4)):

$$P_i[b, :, :] = \frac{1}{H} \sum_{h=1}^H A_i[b, h, :, :]$$

$$H(P_i[b, t, :]) = - \sum_{k=1}^T P_i[b, t, k] \log(P_i[b, t, k] + \varepsilon)$$

$$EA_i = \frac{1}{B \cdot T} \sum_{b=1}^B \sum_{t=1}^T H(P_i[b, t, :])$$
12. Composite Score as given in Equation (5)

$$S_i = \alpha \cdot AM_i + \beta \cdot (1 - EA_i), \alpha = 1.0, \beta = 1.0$$
13. **End for**
14. Select top- (k) layers as given in Proposed novel Equation (6)

$$L_k = \text{TopK}(\{(L_i, S_i)\}_{i=1}^N, k)$$
15. **Step 3: Adapter Injection**
16. Insert adapters and define parameters
17. **For each layer $i \in L_k$ do**
 LayerNorm as given in Equation (7)
 Down-projection as given in Equation (8)
 Up-projection as given in Equation (9)
 Residual output as given in Equation (10)
 Adapter parameters as given in Equation (11)
 Total trainable parameters as given in Equation (12)
 Training objective as given in Equation (13)
18. **End for**

5. Results and Analysis

SCALE has been benchmarked across five datasets, as stated in Table 1. Three binary classification datasets are used, IMDB, Stanford Sentiment Treebank (SSTV-2), and Yelp, and two multi-class classification datasets are used, XLive and Twitter. Since XLive labels originate from TextBlob, improved accuracy on this dataset may partly reflect a closer approximation of TextBlob's decision boundaries rather than improved semantic sentiment reasoning. For this reason, conclusions about sentiment understanding are drawn primarily from manually annotated benchmark datasets, while XLive is treated as a robustness test

under weak supervision. SCALE is compared with the state-of-the-art baseline methods, including LoRA [7], Adalora [8], and BitFit [10]. Five performance evaluation metrics are used: Accuracy, Precision, Recall, F1-score, and Area-Under-the-Curve (AUC). Accuracy reflects the overall correct prediction rate, while precision highlights the model's ability to avoid false positives. Recall highlights the capacity to identify all relevant instances, basically minimizing false negatives. Lastly, the F1-score, which is the harmonic mean of Precision and Recall, provides a balanced view, especially in cases of class imbalance. All methods were trained under an identical experimental protocol. The same bert-base-uncased backbone, tokenizer, sequence length (128), stratified data split, and random seed were used. Hyperparameters were matched, including epochs, learning rate 3×10^{-5} , effective batch size 64 (via gradient accumulation), AdamW optimizer with weight decay 0.01, and cosine scheduler with 0.1 warmup. The only difference between baselines is which parameters were updated, and trainable parameter counts were explicitly measured and reported. Together, these metrics provide a comprehensive evaluation of how well the model is performing from various perspectives, which is vital for robust performance evaluation.

5.1. Binary Datasets

To evaluate the proposed SCALE-BERT method, we used three well-known binary classification datasets: IMDB, SSTV-2, and YELP. The IMDB dataset contains long-form reviews, whereas the SSTV-2 dataset contains sentiment-polarized sentences. Lastly, the YELP dataset contains real-world reviews, both formal and informal.

5.1.1. Accuracy

The evaluation of prediction accuracy for SCALE-BERT, LoRA-BERT, AdaLoRA-BERT, and BitFit-BERT across the IMDB, SSTV-2, and YELP sentiment classification datasets reveals distinct performance. The IMDB dataset contains long and context-rich movie reviews. As depicted in Figure 2, SCALE-BERT produced an accuracy of 87.7% and outperforms LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT by 2.3%, 11.8% and 8.3%, respectively. BitFit-BERT achieved an accuracy of 75.9% on this dataset. This indicates that SCALE-BERT maintains long-range contextual dependencies, whereas the other two models struggle with this due to the vanishing gradient problem. Finding the correct sentiments at times depends on long contextual text, and SCALE-BERT excels in understanding and balancing the representation of global and local sentiment cues. On the SSTV-2 dataset, which consists of short, sentiment-polarized sentences, SCALE-BERT showed the same trend and achieved the highest accuracy of 98.2%; LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT fell short by 1.7%, 9.9%, and 7.9%, respectively. BitFit-BERT produced an accuracy of 88.3% on SSTV2. This highlights, in text of limited size, that SCALE-BERT can efficiently capture fine-grained sentiment variations. Lastly, for the YELP dataset, which consists of formal and informal expressions with variable review lengths, SCALE-BERT produced 94.1% accuracy. LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT fall short by 1.3%, 3.9%, and 2.5%, respectively. BitFit-BERT achieved a competitive accuracy of 90.2% on the YELP dataset. In conclusion, the sentiment prediction accuracy highlights that SCALE-BERT offers the most balanced combination of precision and generalization across diverse datasets of varying complexity and length. Furthermore, the confusion matrices are illustrated in Figure 3 for a more detailed view.

Comparative Performance of Models Across Datasets

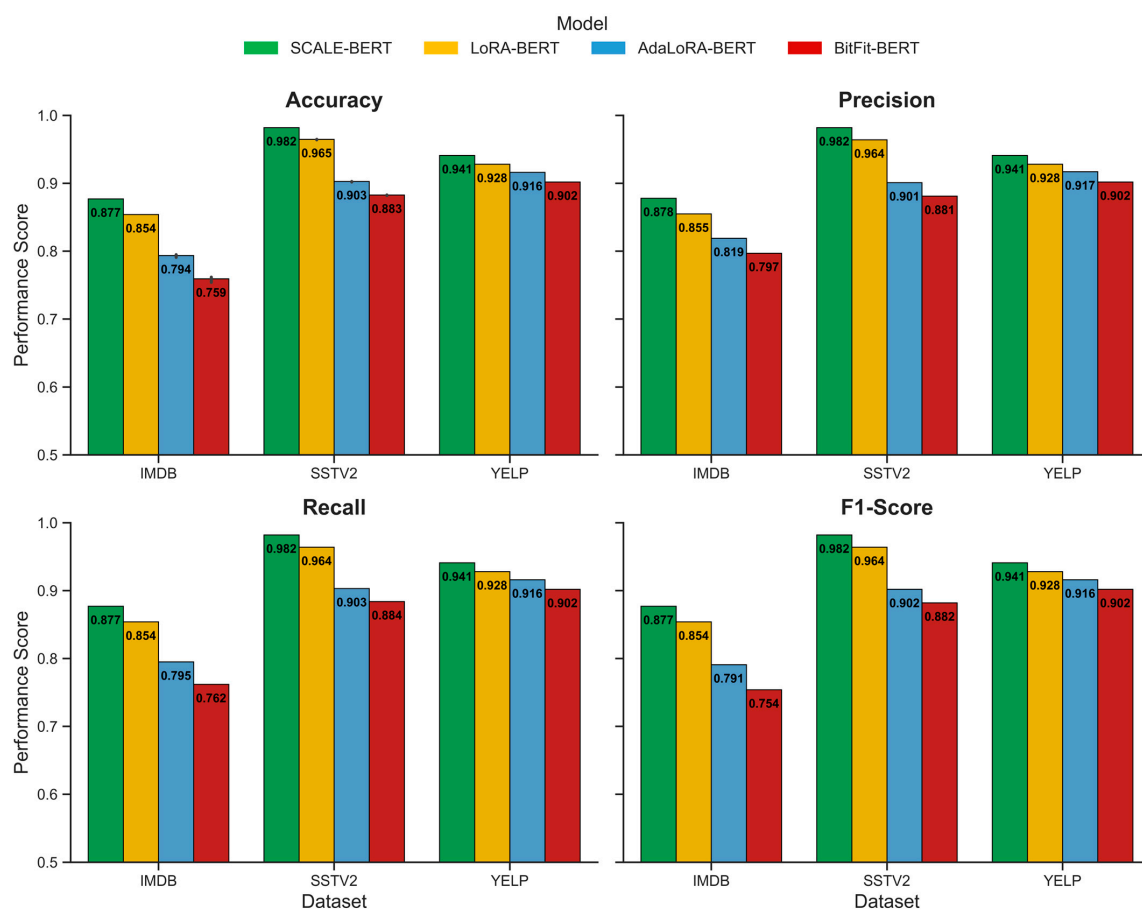


Figure 2. Comparative performance of SCALE-BERT, LoRA-BERT, and AdaLoRA-BERT BitFit-BERT across IMDB, SSTV2, and Yelp datasets based on Accuracy, Precision, Recall, and F1-Score.

Confusion Matrices of Binary Classification Datasets

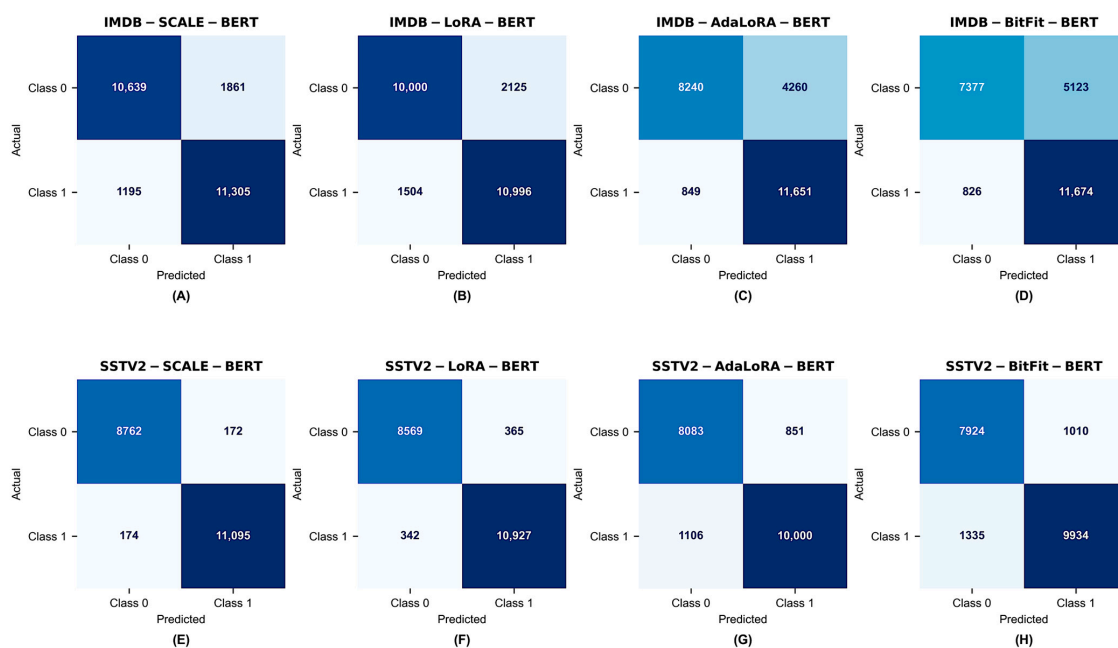


Figure 3. Cont.

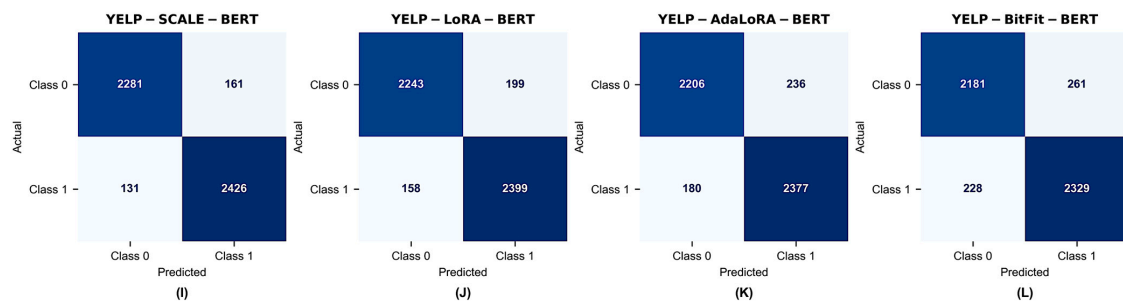


Figure 3. Confusion matrices for binary classification on IMDb, SSTV2, and Yelp datasets. (A) IMDb-SCALE-BERT strong correct classification; (B) IMDb-LoRA-BERT slightly higher false positives; (C) IMDb-AdaLoRA-BERT increased Class 0 misclassification; (D) IMDb-BitFit-BERT higher errors. (E) SSTV2-SCALE-BERT clear class separation; (F) SSTV2-LoRA-BERT moderate false positives; (G) SSTV2-AdaLoRA-BERT increased Class 1 misclassification; (H) SSTV2-BitFit-BERT higher overall errors. (I) Yelp-SCALE-BERT good separation; (J) Yelp-LoRA-BERT moderate overlap; (K) Yelp-AdaLoRA-BERT slight misclassification increase; (L) Yelp-BitFit-BERT largest overlap.

5.1.2. F1-Scores

The F1-score gives a balanced measure of both precision and recall. It is a measure of how effectively each model identifies positive sentiment while minimizing both false positives and false negatives. As depicted in Figure 2, for the IMDB dataset, SCALE-BERT achieved an F1-score of 0.877, outperforming LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT, which achieved F1-scores of 0.854, 0.754, and 0.791, respectively. BitFit-BERT produced an F1-score of 0.754 on the IMDB dataset. F1-score of SCALE-BERT highlights that it consistently performed well across extended text. Further, on the SSTV-2 dataset, SCALE-BERT achieved a near-perfect F1-score of 0.982, considerably surpassing LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT, which achieved F1-scores of 0.964, 0.882, and 0.902, respectively. BitFit-BERT achieved an F1-score of 0.882 on SSTV2. This proves the optimal performance of SCALE-BERT in handling short and sentiment-polarized text. Lastly, for the YELP dataset, which contains a mix of formal and informal reviews, SCALE-BERT achieved an F1-score of 0.941, slightly higher than LoRA-BERT and significantly higher than BitFit-BERT and AdaLoRA-BERT, which achieved F1-scores of 0.928, 0.902, and 0.916, respectively. BitFit-BERT reached an F1-score of 0.902 on the YELP dataset. Importantly, averaging across all datasets, SCALE-BERT achieves an overall F1-score of 0.933, which is 0.018 higher than LoRA-BERT and 0.063 higher than AdaLoRA-BERT. In conclusion, SCALE-BERT has demonstrated consistent performance across short and long texts and is the most balanced of the three models.

5.1.3. Area-Under-the-Curve (AUC)

The Area-Under-the-Curve (AUC) quantifies how well the trained model distinguishes between the positive and negative classes. The AUC measure is between (0–1). An AUC closer to 1 indicates that a model with high confidence can classify the classes with greater discriminative power across varying decision thresholds, as visualized in the Receiver Operating Characteristic (ROC) curve. SCALE-BERT produced superior and consistent ROC performance across the IMDB, SSTV-2, and YELP datasets for binary text classification tasks, as depicted in Figure 4. On the IMDB dataset, which contains lengthy and noisy reviews, SCALE-BERT produced an AUC of 0.949, whereas LoRA-BERT, AdaLoRA-BERT, and BitFit-BERT fell behind with AUCs of 0.912, 0.913, and 0.897, respectively. BitFit-BERT recorded the lowest discriminative power on this dataset with an AUC of 0.897. This demonstrates how well SCALE-BERT manages long-range dependencies and contextual variability. In heterogeneous text conditions, AdaLoRA-BERT, which employs adaptive rank reallocation, fails to discriminate between classes.

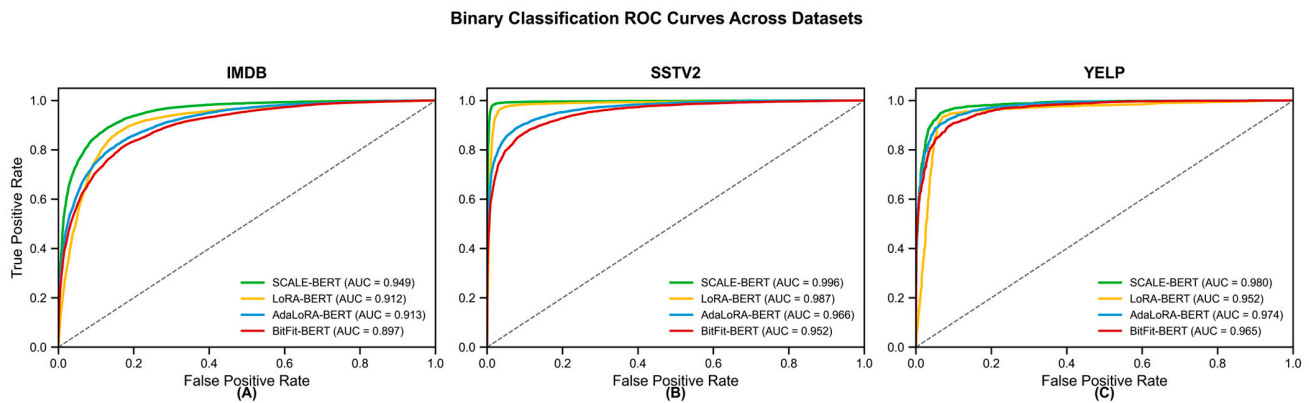


Figure 4. Binary ROC curves for sentiment classification on three datasets. (A) IMDB—highest AUC observed; (B) SSTV2—near-perfect discrimination; (C) Yelp—strong separation with slightly lower performance than SSTV2.

In comparison, SCALE-BERT yielded an AUC of 0.996 on the SSTV-2 dataset, with both conciseness and syntactic meaning. This shows the sensitivity–specificity balance of SCALE-BERT; whereas, lagging slightly behind with 0.987 and 0.966, BitFit-BERT produced an AUC of 0.952. This confirms that while low-rank adaptation can perform competitively on structured, low-noise data, SCALE-BERT maintains a clear performance edge. Additionally, this affirms that low-rank adaptation can perform optimally on structured, low-noise SSTV-2 data. Lastly, on the YELP dataset, which is a collection of short and long reviews with sentiment cues, SCALE-BERT achieved the highest AUC of 0.980. LoRA-BERT and AdaLoRA-BERT followed with AUCs of 0.952 and 0.974, respectively, while BitFit-BERT achieved an AUC of 0.965. BitFit-BERT performed competitively on the YELP dataset, outperforming LoRA-BERT in terms of AUC. In conclusion, SCALEBERT performed consistently and achieved the most stable AUC for all three cases.

5.2. Multi-Class Datasets

The evaluation of SCALE-BERT, LoRA-BERT, AdaLoRA-BERT, and BitFit-BERT on multi-class classification highlights the distinct performance differences in how each model generalizes across diverse datasets. Compared to binary classification, multi-class classification introduces more variability due to overlapping decision boundaries. Two multi-class datasets are used: the TweetEval dataset with three sentiment classes, and the XLive dataset with five sentiment classes. The multi-class datasets help to evaluate the model and prove its generalization.

5.2.1. Accuracy

The evaluation of SCALE-BERT, LoRA-BERT, AdaLoRA-BERT, and BitFit-BERT on the XLive and TweetEval multi-class classification datasets shows SCALE-BERT as the winner, as depicted in Figure 5. On the XLive dataset, SCALE-BERT achieved an accuracy of 83.1%, which is 1.9% higher than LoRA-BERT, 40.9% higher than BitFit-BERT, and 38.9% higher than AdaLoRA-BERT, as depicted in Figure 5. BitFit-BERT achieved an accuracy of 42.2% on the XLive dataset. Similar performance is seen with the TweetEval dataset. SCALE-BERT achieved 89.2% accuracy, far exceeding LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT, by 8.6%, 30.2%, and 25.2%, respectively. BitFit-BERT reached an accuracy of 59.0% on the TweetEval dataset. In conclusion, these results confirm that SCALE-BERT produced optimal and decisive edges in a multi-class classification problem. It performed consistently, achieving higher accuracy across binary and multi-class datasets of variable length. Scalability and contextual fidelity are supported by its architectural design. Additionally, the class-wise performance details are shown in Figure 6 as a confusion matrix.

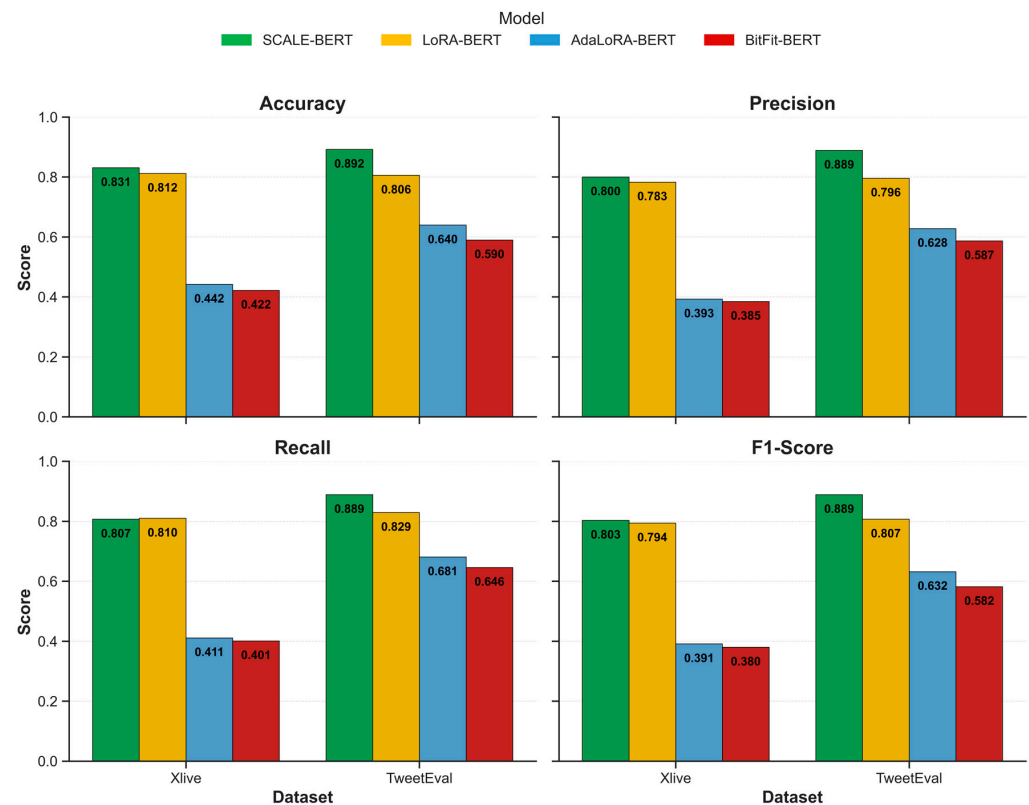


Figure 5. Comparative performance of SCALE-BERT, LoRA-BERT, AdaLoRA-BERT, and BiFit-BERT on XLive and TweetEval multi-class datasets across Accuracy, Precision, Recall, and F1-Score metrics.

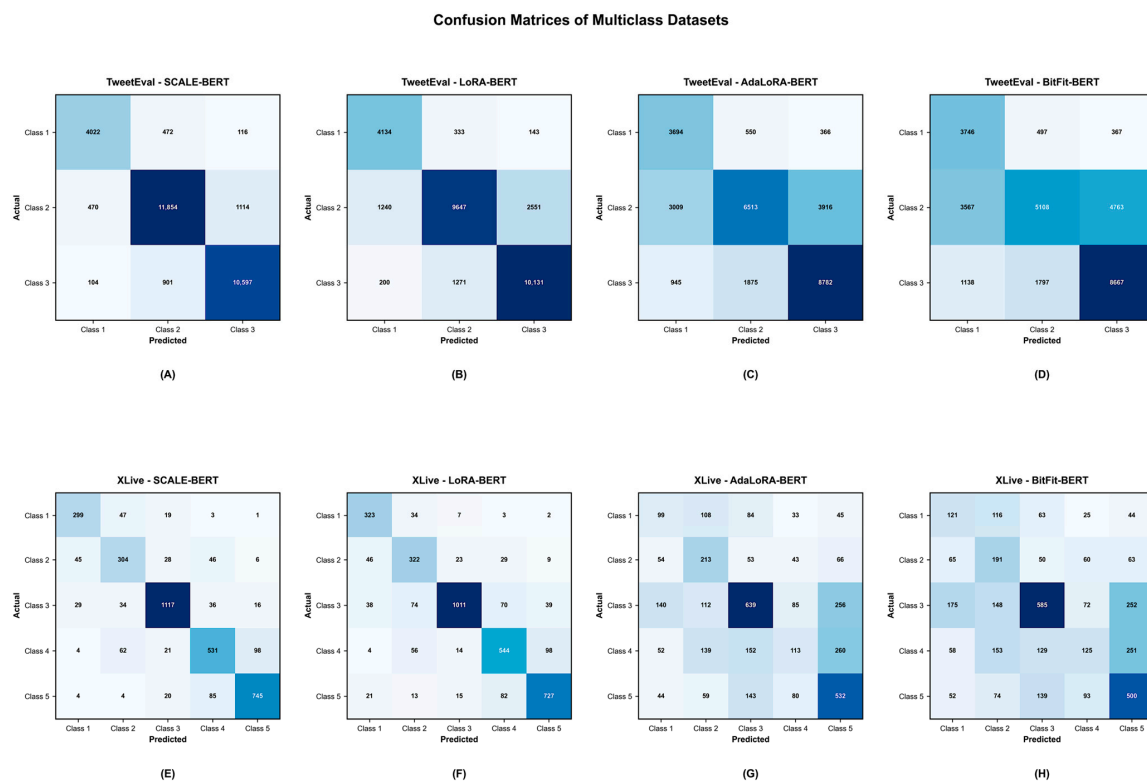


Figure 6. Confusion matrices for TweetEval and XLive multiclass classification. (A) TweetEval strong diagonal concentration; (B) TweetEval moderate class confusion; (C) TweetEval increased misclassification; (D) TweetEval highest errors. (E) XLive clear class separation; (F) XLive moderate overlap; (G) XLive substantial confusion; (H) XLive largest overlap.

5.2.2. F1-Scores

The F1-score produces a comprehensive, well-balanced picture of recall and precision across several sentiment classes. As shown in Figure 5, SCALE-BERT achieved an F1-score of 0.803 on the XLive dataset, marginally higher than those of LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT, by 0.009, 0.423, and 0.412, respectively. BitFit-BERT produced an F1-score of 0.380 on the XLive dataset. Similarly, on the TweetEval dataset, SCALE-BERT produced an F1-score of 0.889, as depicted in Figure 5, outperforming LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT by 0.082, 0.307, and 0.257, respectively. BitFit-BERT achieved an F1-score of 0.582 on the TweetEval dataset. Overall, SCALE-BERT shows consistent performance, achieving higher F1 Scores across both datasets. This also highlights SCALE-BERT's ability to handle complex, multi-label datasets, which are more challenging. The scalable parameter adaptation mechanism employed by SCALE-BERT does not produce higher accuracy and balances accuracy across the classes, making it a more reliable and generalizable model.

5.2.3. Area-Under-the-Curve (AUC)

The ROC–AUC analysis is depicted in Figure 7. It highlights how well SCALE-BERT, LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT can distinguish multiple sentiments in both the TweetEval and XLive datasets. In multi-class classification problems, micro-AUC and macro-AUC are computed. Micro AUC measures a model's discriminative ability by providing a single summary of predictions across all classes, thereby focusing on the main categories. In contrast, macro-AUC computes the average AUC across each class independently. High micro-AUC means strong overall performance. A high macro-AUC indicates balanced effectiveness of major and minor sentiment classes. On the TweetEval dataset, SCALE-BERT exhibits exceptional performance, with class-wise AUC values ranging from 0.95 to 0.97 and a micro-AUC and macro-AUC of 0.96 each, indicating optimal performance. LoRA-BERT yields a lower micro-AUC value of 0.94 and a macro-AUC value of 0.94. This means partial class confusion. This is further evident in Class 2, with a significantly lower AUC of 0.90, indicating a loss of sensitivity in detecting subtle sentiments. BitFit-BERT achieved micro and macro-AUC values of 0.80 each on the TweetEval dataset, with class-wise performance significantly lower than SCALE-BERT, ranging between 0.71 and 0.89. Lastly, AdaLoRA-BERT performed poorly, with a micro-AUC of 0.83 and macro-AUC of 0.84. The low AUCs of LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT indicate that their adaptive ranking mechanisms struggle to establish stable decision boundaries in highly variable text in tweets. A similar pattern is observed in the XLive dataset. The SCALE-BERT produces the highest AUC across all classes, ranging between 0.92 and 0.98. At the same time, it produces micro-AUC and macro-AUC values of 0.96 and 0.95, respectively. This shows its consistent performance for all five sentiment classes. However, LoRA-BERT's AUC performance falls short compared to SCALE-BERT, and BitFit-BERT recorded micro- and macro-AUC values of 0.73 and 0.72, respectively, on the XLive dataset. BitFit-BERT's class-wise performance on XLive ranged from 0.65 to 0.77, suggesting minor generalization loss in complex multi-class environments. Additionally, AdaLoRA-BERT's AUC is significantly lower, with micro-AUC and macro-AUC of 0.75 and 0.73, respectively. Overall AUC results show SCALE-BERT's robustness and adaptability in handling multiple classes and understanding their complex inter-class dependencies.

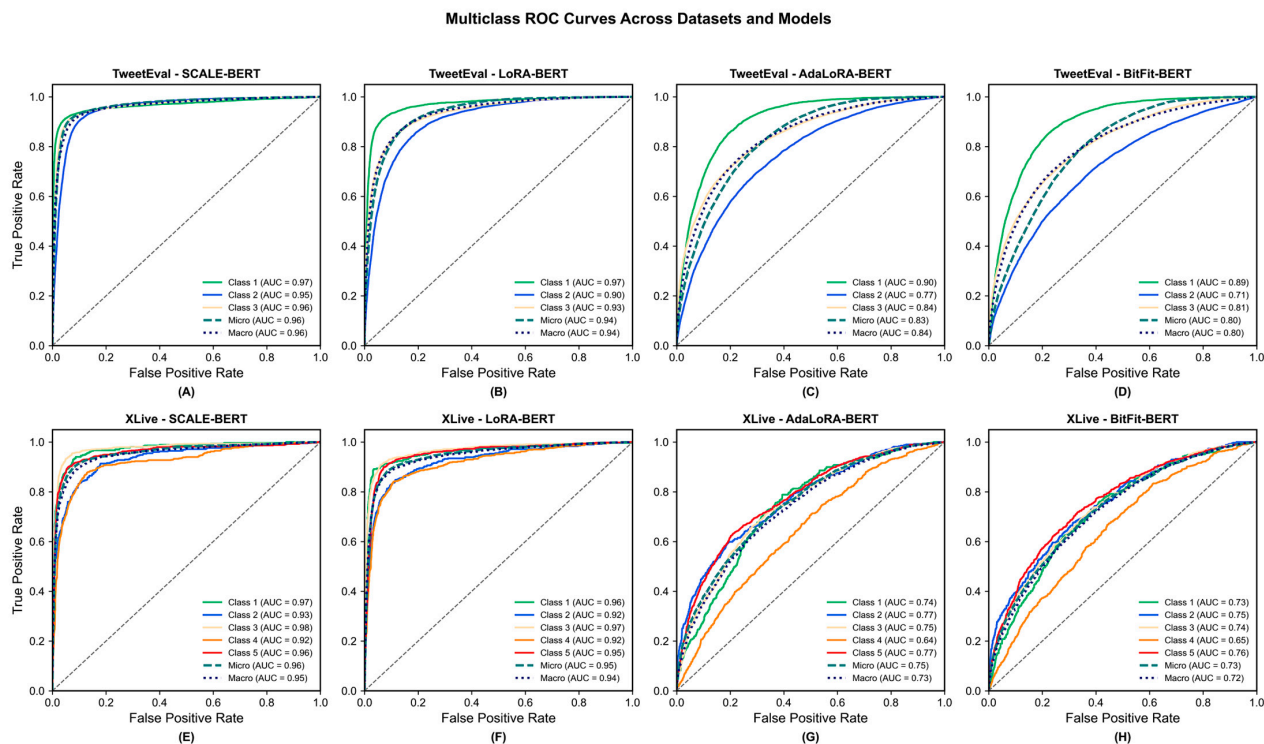


Figure 7. Multiclass ROC curves on TweetEval and XLive comparing SCALE-BERT, LoRA-BERT, AdaLoRA-BERT, and BitFit-BERT. (A) TweetEval-SCALE-BERT shows the highest separability (macro-AUC ≈ 0.96); (B) LoRA-BERT slightly lower (≈ 0.94); (C) AdaLoRA-BERT reduced discrimination (≈ 0.84); (D) BitFit-BERT lowest performance (≈ 0.80). (E) XLive-SCALE-BERT maintains strong separation (≈ 0.95); (F) LoRA-BERT marginally lower (≈ 0.94); (G) AdaLoRA-BERT degraded performance (≈ 0.73); (H) BitFit-BERT lowest separability (≈ 0.72).

5.3. Training Time

The comparative model training time analysis of the SCALE-BERT, LoRA-BERT, BitFit-BERT, and AdaLoRA-BERT on five different datasets, IMDB, SSTV2, YELP, TweetEval, and XLive, shows the computational superiority of the proposed approach. As shown in Figure 8, SCALE-BERT consistently achieves the lowest training time, indicating superior convergence stability. On IMDB, SCALE-BERT completed training in 7.94 min, compared to 8.63 min for BitFit-BERT, 11.01 min for LoRA-BERT, and 17.06 min for AdaLoRA-BERT. This represents an 8.0% reduction in training time compared to BitFit-BERT, and a 27.9% and 53.5% reduction compared to LoRA-BERT and AdaLoRA-BERT, respectively. On SSTV2, SCALE-BERT required 20.99 min, which is 5.7% faster than BitFit-BERT (22.27 min), 28.9% faster than LoRA-BERT (29.53 min), and 55.7% faster than AdaLoRA-BERT (47.36 min). For YELP, SCALE-BERT trained in 15.22 min, outperforming BitFit-BERT (18.88 min) by 19.4%, LoRA-BERT (22.19 min) by 31.4%, and AdaLoRA-BERT (34.64 min) by 56.0%. In the case of the TweetEval dataset, SCALE-BERT completed training within 14.00 min, showing a 17.1% reduction over BitFit-BERT (16.88 min), 30.3% over LoRA-BERT (20.1 min), and 54.8% over AdaLoRA-BERT (30.98 min). Finally, on XLive, SCALE-BERT achieved 9.72 min, which is 14.1% faster than BitFit-BERT (11.32 min), 24.1% faster than LoRA-BERT (12.8 min), and 50.0% faster than AdaLoRA-BERT (19.46 min). The gradual drop in training time across datasets of varying complexity demonstrates SCALE-BERT's scalability and low-rank efficiency. These results confirm that the profiling-based adapter injection and selective freezing mechanism in SCALE-BERT fine-tune the model without wasting extra computation, making it the most time-efficient alternative to adaptive and uniform parameter-efficient fine-tuning methods, including BitFit.

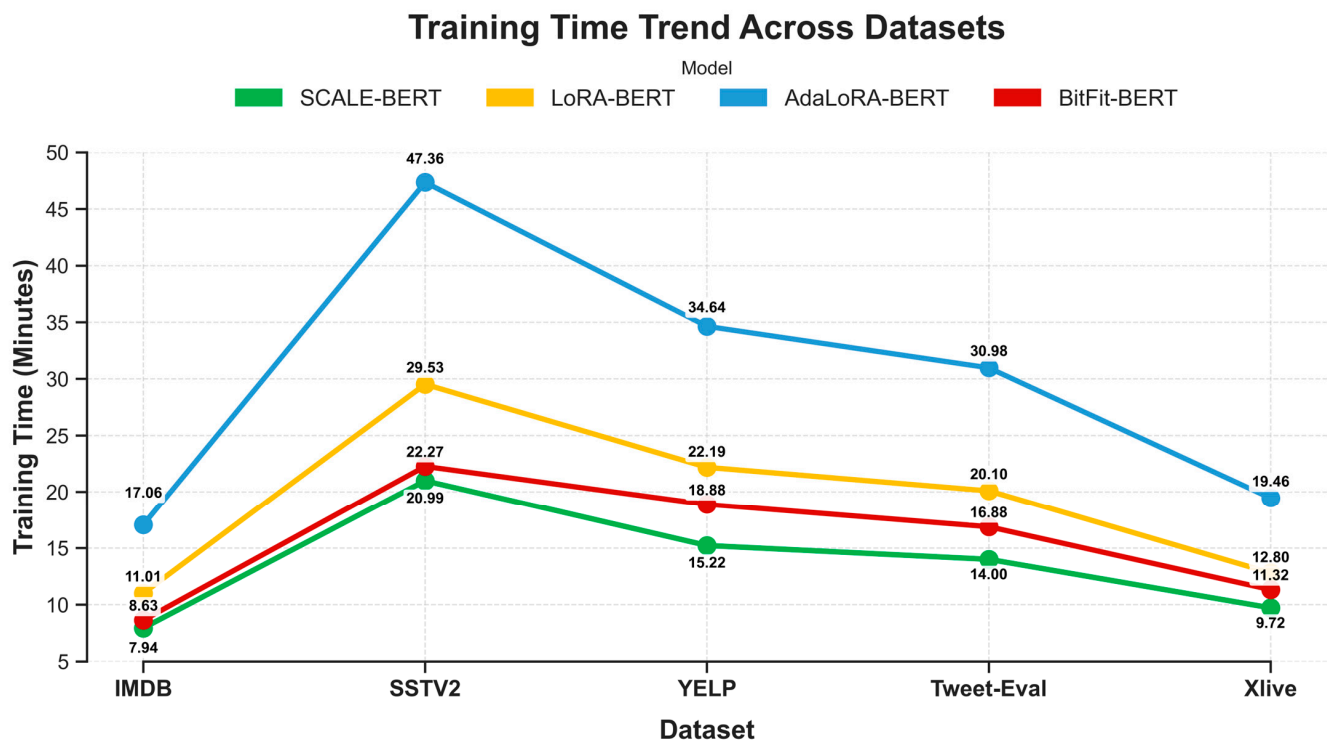


Figure 8. Training time trends showing that SCALE-BERT consistently achieves the lowest training time compared to BitFit-BERT, LoRA-BERT, and AdaLoRA-BERT across all datasets.

6. Ablation Study

To further validate the effectiveness of the SCALE, an ablation study was conducted to analyze the influence of selective layer injection on performance, training time, and parameter utilization. Four configurations were compared: Full Model Fine-Tuning, which updates all weights, and the proposed selective layer selection methods, Uniform Layer Selection and Adaptive Layer Selection. The evaluation was conducted on both the binary-class dataset IMDB and the multi-class dataset Tweepval for sentiment classification, and the comparative results are presented in Table 3.

Table 3. Ablation study comparing performance, training efficiency, and parameter utilization across Full Fine-Tuning, Uniform, Adaptive, and SCALE methods.

Method	Accuracy (Binary)	Accuracy (Multi-Class)	Training Time (Binary, min)	Training Time (Multi-Class, min)	Trainable Parameters
Full-Model Fine Tuning	0.882	0.910	17.70	31.95	109 M
SCALE-Proposed	0.877	0.892	7.73	13.96	42 M
Selective Layer Selection	0.854	0.806	11.00	19.82	42 M
Uniform Layer Selection	0.795	0.640	17.01	30.72	42 M
Adaptive Layer Selection					

The Uniform Layer Selection takes a simple approach: layers are selected at equal intervals for model parameter updates, regardless of their contribution to learning. This uniform spacing reduces the number of trainable parameters but often makes adaptation inefficient, since all layers are treated equally.

On the other hand, the Adaptive Layer Selection dynamically decides which layers to fine-tune based on adaptive criteria, such as changes in gradients or loss values. Although this method aims to improve flexibility, in practice, it is sometimes unstable during training and inconsistent in convergence due to frequent rank and layer changes during optimization. The accuracy of the proposed SCALE Selective Layer Selection for binary

classification was 0.877, which was slightly lower than Full Fine-Tuning, 0.882, but higher than Uniform Layer Selection, 0.854, and Adaptive Layer Selection, 0.795, indicating gains of 2.7% and 10.3%, respectively. This demonstrates how SCALE preserves accuracy near the whole model while requiring less computation. With gains of 10.6% and 25.2%, respectively, SCALE outperformed Uniform, 0.806, and Adaptive, 0.640, in multi-class classification, achieving 0.892, just below Full Fine-Tuning, 0.910. These findings demonstrate that, through profiler-guided selective layer optimization, SCALE effectively uses parameters while offering balanced accuracy across both binary and multi-class tasks. Comparing the four configurations' efficiency highlights the trade-offs between computation and performance. Since it modifies all 109 million parameters, the Full-model fine-tuning serves as an initial starting point, accomplishing the highest efficiency but requiring a significant amount of computation, 17.70 min for binary classification and 31.95 min for classification with multiple classes. This setup turns out to be the most time- and memory-intensive setup.

In contrast, SCALE-proposed selective layer selection achieves the lowest time, with only 7.73 min for binary and 13.96 min for multi-class classification, providing an average 56.3% reduction in training time. It also fine-tunes only 42 million parameters, resulting in a 61.4% reduction in trainable parameters compared to the full model while retaining over 99% of the full model's accuracy. Uniform Layer Selection performs moderately, completing training in 11.00 min for binary and 19.82 min for multi-class tasks, approximately 38% faster than full fine-tuning. However, its accuracy drops by 3.1% in binary and 9.6% in multi-class classification, as it treats all layers equally without accounting for their contributions to learning.

Adaptive layer selection, though designed to adjust layers dynamically, suffers from unstable optimization. It takes 17.01 min for binary and 30.72 min for multi-class tasks, only about 4% faster than full fine-tuning, yet its accuracy drops by 9.8% and 27.6%, respectively. Overall, among all compared methods, SCALE clearly shows the best performance balance. It maintains almost the same accuracy as full-model fine-tuning while using less than half the training time and under 40% of the parameters. This strongly indicates that the profiler-guided selective layer injection in SCALE effectively identifies and updates only the most critical layers, thereby maintaining an effective trade-off between accuracy, faster convergence, and better parameter efficiency across both binary and multi-class tasks. Furthermore, we will compare efficiency in the Section 7.

7. Discussion

In this part, we address a critical discussion of the suggested SCALE method for efficient fine-tuning of BERT-base models in the context of sentiment analysis. The findings presented in Figure 8 indicate that SCALE is a cost-effective and performance-consistent alternative to traditional fine-tuning and modern PEFT methods. SCALE, on the other hand, only updates the most effective active transformer layers based on activation size and attention-entropy profiling. It differs from full-model fine-tuning and thorough search-based adaptation. This selective process enables a good balance between performance and efficiency. To put these findings in the context of other research, SCALE is compared with a few recent fine-tuning techniques that focus on computational scalability, parameter efficiency, and robustness regarding dataset distribution in sentiment analysis. The present study does not evaluate cross-task or cross-domain transfer, such as zero-shot or few-shot adaptation to unrelated NLP tasks.

Hu et al. [7] proposed LoRA, which makes adaptation easier by learning low-rank modifications for each layer of a trained model. LoRA still spreads updates across all layers, so it does not work well in areas with little effect. This significantly reduced the number of parameters that can be trained. In terms of SCALE, this is better because it profiles internal

actions to limit adaptation to the top-k layers that are most responsive and influential. This results in better efficiency while maintaining equivalent or superior performance, as demonstrated for both binary and multi-class operations. Zhang et al. [8] also came up with AdaLoRA, and this dynamically reallocates prioritized budgets based on how important each parameter is. This gives you more options when you have few resources. But AdaLoRA's adaptive score variation often makes optimization unstable, especially when there are long text dependencies. SCALE navigates this problem by keeping a static, rather than data-driven, selective injection enabled, which ensures that the gradient flow remains constant and convergence remains stable across evaluated sentiment datasets.

Dettmers et al. [6] created QLoRA, which enables memory-efficient fine-tuning of transformer models through low-rank adapters combined with 4-bit quantization. QLoRA uses memory very well, but it uses adapters precisely and does not account for the sensitivity of transformer components to different layers. SCALE's activation entropy evaluation addresses this gap by demonstrating where modifications yield the most information. This is why SCALE achieves the same level of accuracy with fewer resources and trains faster. In a similar way, Houlsby et al. [9] invented adapter tuning as a less heavy version of full fine-tuning. They used small neural components that were placed after each level of attention and feed-forward. Placing adapters in the same spot across all layers might not work, even though it does. SCALE goes a step further by allowing adapters to operate only on layers that will have the greatest impact on the dataset's characteristics. This reduces extra work while maintaining the task's accuracy. Other sparse and selective tuning methods, like DiffPrune [12] and Delta Tuning [13], also want to lower the cost of fine-tuning. While BitFit [10] achieves high efficiency by updating only bias terms, our results (Section 5) show it struggles with discriminative power in context-rich datasets like IMDB and multi-class tasks like XLive. Using binary constraints and L0 regularization, DiffPrune removes unnecessary weights. Delta Tuning, on the other hand, sorts parameter-efficient methods into structured groups, such as prompts and adapters. SCALE can be seen as a mix of these two ideas. It obtains structural sparsity by selectively freezing layers, and it keeps a strong learning capacity in a few layers that are relevant to the task at hand. Because of this, SCALE maintains more than 99% of the full-model predictive power within the evaluated sentiment classification datasets (Section 6) while using just over half the parameter values and training time, as shown in Section 6.

He et al.'s recent work (SensiLoRA-RAG) combines sensitivity-aware LoRA via retrieval-augmented generation to improve fine-tuning in specialized areas. SensiLoRA focuses on adjusting ranks based on sensitivity, while SCALE focuses on selecting layers based on profiling. This makes SCALE independent of layer depth within the evaluated BERT classification architecture. Mao et al. [15] surveyed LoRA variants focusing on task adaptability; however, none of the techniques employed include quantitative inner profiling for significance estimation. SCALE's use of activation strength and attention energy gives layer prioritization a clear and repeatable base, which makes it easier to understand something that is often missed in PEFT research. In the realm of implementation, Wang et al. [19] and Barreto et al. [20] assessed LoRA-tuned transformers for sentiment and emotion classification in social media datasets. These works indicate the growing popularity of parameter-efficient methods in sentiment modeling. But their dependence on identical or rank-based adjustments still makes it hard to adapt. SCALE's selective approach not only reduces computation but also ensures consistent performance across both structured datasets (e.g., SSTV-2 and Yelp) and unstructured, noisy datasets (e.g., TweetEval and XLive). The XLive corpus is weakly supervised, and its sentiment labels are generated automatically using the TextBlob polarity estimator rather than human annotation. Consequently, the labels represent heuristic sentiment approximations rather than verified ground truth.

Performance improvements on XLive may therefore partly reflect a closer approximation of TextBlob decision boundaries rather than deeper semantic sentiment understanding. For this reason, XLive is interpreted only as an evaluation under noisy pseudo-label conditions and not as an independent benchmark of sentiment comprehension. This strength comes from the data-aware profiling system, which identifies transformer regions relevant to the context without updating all parameters. The comparative performance analysis shows that SCALE had better accuracy, F1-score, and AUC across all datasets. Specifically, SCALE outperformed BitFit-BERT in accuracy by 11.8% on IMDB and 30.2% on TweetEval. It also reduced training time by up to 56.3% compared with full fine-tuning and by an average of 40–50% compared with LoRA and AdaLoRA. While BitFit-BERT was competitive in terms of training time, SCALE still achieved an average reduction of 5–19% in training time across the five evaluated datasets (Section 5.3). This shows that profiling-based selective injection can improve computational efficiency without hurting representational strength. Also, SCALE's stable performance across a 5-fold validation set shows that it generalizes across evaluated sentiment datasets with different distributions. In particular, results on XLive should be interpreted as robustness to weak supervision rather than evidence of improved sentiment reasoning, and conclusions about sentiment understanding are based primarily on human-annotated datasets, even when the dataset is unbalanced or when text lengths vary. From a technological and market standpoint, the increasing use of large-scale transformer designs in consumer-grade systems requires ongoing fine-tuning methodologies.

As noted in [17], the next generation of customized resource-efficient BERT adaptation for sentiment classification will be based on efficient fine-tuning frameworks. SCALE aligns with this goal by making it easier for everyone to use BERT fine-tuning by better leveraging resources. Because it uses little memory and has an easy-to-understand profiling mechanism, it can be used on the edge, in educational research settings, or by small businesses without access to high-end GPU clusters. In short, SCALE is better than current methods such as LoRA [7], AdaLoRA [8], and BitFit [10] because it combines transformer interpretability with selective fine-tuning for sentiment classification applications. Its profiling-guided phase selection ensures shorter training time, higher accuracy, and lower energy use, while maintaining or exceeding the starting point. The SCALE method is a balanced and scalable step forward in parameter-efficient fine-tuning for sentiment analysis tasks. It closes the gap between high-performance modifications and makes them easy to use for real-world BERT applications. The experiments in this study evaluate robustness across datasets within sentiment classification. Evaluation of selective layer adaptation under true cross-domain transfer, including zero-shot or few-shot settings, is outside the scope of the present work and remains future research.

8. Conclusions

Over the last decade, BERT has advanced our understanding of natural languages, but optimizing these massive and complex architectures remains a significant challenge, including high computational costs, high memory requirements, and limited access to hardware in low-resource environments. These issues are addressed by SCALE, which uses a profiling-based selective fine-tuning approach that concentrates only on the most significant transformer layers, as determined by activation strength and attention entropy. This technique enables effective adaptation while largely retaining the pretrained model's strong representational power. Despite using only half the trainable parameters and requiring significantly less training time than LoRA and AdaLoRA, SCALE demonstrated significant improvements in accuracy, F1 Scores, and AUC across extensive studies in both binary and multi-class sentiment analysis tasks. Results obtained on the XLive dataset

should be interpreted cautiously, as its labels are automatically generated using TextBlob; thus, it assesses stability under weak supervision rather than true ground-truth sentiment prediction. According to the results, better efficiency and more stable convergence during training are achieved by carefully selecting which layers to fine-tune rather than updating all layers at once.

Additionally, SCALE's interpretability makes it easier to see which layers are most important, increasing visibility into the fine-tuning procedure. It is a more practical option for accessible and useful AI deployment since it also works well in real-world situations where hardware constraints are an issue. In the future, SCALE can be expanded to include encoder–decoder and decoder-only architectures, creating opportunities for broader tasks such as dialogue systems, reasoning, and summarization. Future studies will examine flexible profiling for real-time layer adaptation and combine SCALE with numerical or mixed-precision training to further increase energy efficiency. The experimental results indicate that profiling-guided adapter placement improves parameter efficiency for sentiment classification tasks. The current evaluation is limited to sentiment text classification datasets, and the applicability of SCALE to other NLP task categories, including sequence labeling, question answering, and generative text modeling, has not been investigated in this study. In conclusion, SCALE lays a strong foundation for resource-conscious, interpretable, and sustainable fine-tuning of BERT-base models, paving the way for scalable AI that operates reliably even with limited computational resources.

Author Contributions: For conceptualization, S.H.G. and A.A.A.S.; methodology, S.H.G. and A.M.B.; software, S.H.G.; validation, A.A.A.S., A.M.B., T.S.M.A. and F.A.; formal analysis, S.H.G.; investigation, T.S.M.A.; resources, A.M.B. and T.S.M.A.; writing—original draft preparation, A.M.B. and S.H.G.; writing—review and editing, F.A. and A.A.A.S.; visualization, A.A.A.S. and S.Q.; supervision, S.Q. and F.A.; project administration, A.M.B. and A.A.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets IMDB, SST-2, Yelp, and TweetEval are publicly available and the XLive dataset is a private dataset and is not publicly available due to privacy and ethical restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kalyan, K.S. A Survey of GPT-3 Family Large Language Models Including ChatGPT and GPT-4. *Nat. Lang. Process. J.* **2024**, *6*, 100048. [\[CrossRef\]](#)
2. Kumar, P. Large Language Models (LLMs): Survey, Technical Frameworks, and Future Challenges. *Artif. Intell. Rev.* **2024**, *57*, 260. [\[CrossRef\]](#)
3. Parthasarathy, V.B.; Zafar, A.; Khan, A.; Shahid, A. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. *arXiv* **2024**, arXiv:2408.13296. [\[CrossRef\]](#)
4. Zhang, S.; Dong, L.; Li, X.; Zhang, S.; Sun, X.; Wang, S.; Li, J.; Hu, R.; Zhang, T.; Wu, F.; et al. Instruction Tuning for Large Language Models: A Survey. *arXiv* **2023**, arXiv:2308.10792. [\[CrossRef\]](#)
5. Golnari, P.A.; Wang, S. LoRA-Enhanced Distillation on Guided Diffusion Models. *arXiv* **2023**, arXiv:2312.06899. [\[CrossRef\]](#)
6. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Fine-tuning of Quantized LLMs. *arXiv* **2023**, arXiv:2305.14314. [\[CrossRef\]](#)
7. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**, arXiv:2106.09685. [\[CrossRef\]](#)
8. Zhang, Q.; Chen, M.; Bukharin, A.; Karampatziakis, N.; He, P.; Cheng, Y.; Chen, W.; Zhao, T. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. *arXiv* **2023**, arXiv:2303.10512. [\[CrossRef\]](#)

9. Houshy, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; de Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA, 9–15 June 2019; pp. 4944–4953. Available online: <https://proceedings.mlr.press/v97/houshy19a.html> (accessed on 12 June 2025).
10. Ben Zaken, E.; Ravfogel, S.; Goldberg, Y. BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language Models. *arXiv* **2021**, arXiv:2106.10199. [[CrossRef](#)]
11. Rücklé, A.; Geigle, G.; Glockner, M.; Beck, T.; Pfeiffer, J.; Reimers, N.; Gurevych, I. AdapterDrop: On the Efficiency of Adapters in Transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), Punta Cana, Dominican Republic, 7–11 November 2021; pp. 7930–7946. [[CrossRef](#)]
12. Shulman, Y. DiffPrune: Neural Network Pruning with Deterministic Approximate Binary Gates and L_0 Regularization. *arXiv* **2020**, arXiv:2012.03653. [[CrossRef](#)]
13. Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.M.; Chen, W.; et al. Parameter-Efficient Fine-Tuning of Large-Scale Pre-Trained Language Models. *Nat. Mach. Intell.* **2023**, *5*, 220–235. [[CrossRef](#)]
14. He, Y.; Zhu, X.; Li, D.; Wang, H. Enhancing Large Language Models for Specialized Domains: A Two-Stage Framework with Parameter-Sensitive LoRA Fine-Tuning and Chain-of-Thought RAG. *Electronics* **2025**, *14*, 1961. [[CrossRef](#)]
15. Mao, Y.; Ge, Y.; Fan, Y.; Xu, W.; Mi, Y.; Hu, Z.; Gao, Y. A Survey on LoRA of Large Language Models. *Front. Comput. Sci.* **2024**, *19*, 197605. [[CrossRef](#)]
16. Prottasha, N.J.; Mahmud, A.; Sobuj, M.S.I.; Bhat, P.; Kowsher, M.; Yousefi, N.; Garibay, O.O. Parameter-Efficient Fine-Tuning of Large Language Models Using Semantic Knowledge Tuning. *Sci. Rep.* **2024**, *14*, 30667. [[CrossRef](#)] [[PubMed](#)]
17. Wang, L.; Chen, S.; Jiang, L.; Pan, S.; Cai, R.; Yang, S.; Yang, F. Parameter-Efficient Fine-Tuning in Large Language Models: A Survey of Methodologies. *Artif. Intell. Rev.* **2025**, *58*, 227. [[CrossRef](#)]
18. Zhou, L.; Chen, Y.; Li, X.; Li, Y.; Li, N.; Wang, X.; Zhang, R. A New Adapter Tuning of Large Language Model for Chinese Medical Named Entity Recognition. *Appl. Artif. Intell.* **2024**, *38*, 2385268. [[CrossRef](#)]
19. Wang, Y.; Wang, J.; Zhang, X. YNU-HPCC at WASSA-2023 Shared Task 1: Large-Scale Language Model with LoRA Fine-Tuning for Empathy Detection and Emotion Classification. In Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA 2023), Toronto, Canada, 13 July 2023; pp. 526–530. [[CrossRef](#)]
20. Barreto, S.; Moura, R.; Carvalho, J.; Paes, A.; Plastino, A. Sentiment Analysis in Tweets: An Assessment Study from Classical to Modern Word Representation Models. *Data Min. Knowl. Discov.* **2023**, *37*, 318–380. [[CrossRef](#)] [[PubMed](#)]
21. Shen, J.C.; Su, N.J.; Lin, Y.B. Effective Multi-Class Sentiment Analysis Using Fine-Tuned Large Language Model with KNIME Analytics Platform. *Systems* **2025**, *13*, 523. [[CrossRef](#)]
22. Dordevic, N.; Stojkovic, S. Traditional and Parameter-Efficient Fine-Tuning of LLMs for Sentiment Analysis in the English and Serbian Language. In Proceedings of the 2024 11th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2024), Niš, Serbia, 24–27 September 2024. [[CrossRef](#)]
23. Zhan, T.; Shi, C.; Shi, Y.; Li, H.; Lin, Y. Optimization Techniques for Sentiment Analysis Based on LLM (GPT-3). *arXiv* **2024**, arXiv:2401.08673. [[CrossRef](#)]
24. Pavlyshenko, B.M. Analysis of Disinformation and Fake News Detection Using Fine-Tuned Large Language Model. *arXiv* **2023**, arXiv:2311.09731. [[CrossRef](#)]
25. Stanfordnlp/Imdb. *Datasets at Hugging Face*. Available online: <https://huggingface.co/datasets/stanfordnlp/imdb> (accessed on 8 December 2025).
26. Stanfordnlp/Sst2. *Datasets at Hugging Face*. Available online: <https://huggingface.co/datasets/stanfordnlp/sst2> (accessed on 8 December 2025).
27. Yelp_Polarity_Reviews TensorFlow Datasets. Available online: https://www.tensorflow.org/datasets/catalog/yelp_polarity_reviews (accessed on 8 December 2025).
28. Cardiffnlp/Tweet_eval. *Datasets at Hugging Face*. Available online: https://huggingface.co/datasets/cardiffnlp/tweet_eval/viewer/sentiment (accessed on 8 December 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.