GradES: Significantly Faster Training in Transformers with Gradient-Based Early Stopping

Anonymous authors
Paper under double-blind review

Abstract

Early stopping monitors global validation loss and halts all parameter updates simultaneously, which is computationally costly for large transformers due to the extended time required for validation inference. We propose *GradES*, a novel gradient-based early stopping approach that operates within transformer components (attention projections and Feed-Forward layer matrices). We found that different components converge at varying rates during fine-tuning for both language and vision-language models. *GradES* tracks the magnitude of gradient changes in backpropagation for these matrices during training. When a projection matrix's magnitude of gradient changes fall below a convergence threshold τ, we exclude that projection matrix from further updates individually, eliminating costly validation passes while allowing slow converging matrices to continue learning. *GradES* speeds up training time by 1.57–7.22$\times$ while simultaneously enhancing generalization through early prevention of overfitting, resulting in 1.2% higher average accuracy in language tasks and 3.88% on multimodal benchmarks.

1 Introduction

Large language models (LLMs) have remarkable capabilities across diverse tasks, but their training and deployment require substantial computational costs that scale with model size and inference frequency. Due to the high cost of training models with billions of parameters, any effort toward reducing the training cost improves the development of LLMs. This challenge extends to vision-language models (VLMs), where we can observe similar differences in visual and textual modalities.

Fine-tuning LLMs requires balancing computational efficiency against downstream task performance. As transformer architectures scale to billions of parameters, the computation becomes increasingly expensive Dettmers et al. (2023); Malladi et al. (2023). While optimizing for memory and computational efficiency remains important Rawassizadeh (2025), one key issue is often overlooked, i.e., the common practice of extensive fine-tuning assumes that more gradient updates always improve performance Zhang et al. (2021). Research has shown that training for excessive epochs leads to overfitting, where models overfit training data and fail to generalize Mosbach et al. (2021).

Conventional early stopping, which is determined based on loss score, is computationally expensive for large language models, as each validation step requires full forward passes through all transformer layers for every sample in the validation set. This overhead scales linearly with both model size and validation set size, forcing practitioners to validate infrequently, typically every few thousand training steps Tirumala et al. (2022), thereby creating a fundamental trade-off between computational cost and the risk of overfitting. In vision transformers(ViT), the same problem is present because ViT process both image and text inputs through separate transformer encoders. We detail this overhead in 4.

Furthermore, our experiment shows that Transformer's components have different convergence patterns, as shown in Figure 1. This difference in convergence is particularly pronounced in ViT, where vision transformers typically converge slower than their language counterparts. and the binary decision of classic early stopping fails to exploit the diverse convergence patterns, where attention and MLP components exhibit fundamentally different learning dynamics during the fine-tuning process Yao et al. (2025).

Through analysis of gradient dynamics across transformer architectures, we identified a critical inefficiency in current fine-tuning practices, i.e., varied convergence across the Transformer's components. By tracking magnitude of gradient changes for attention projections matrix: $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ that compute queries, keys, and values respectively, and $\mathbf{W}_o$ or output projection. As well as MLP network matrix $\mathbf{W}_{\text{up}}$ for dimension expansion and $\mathbf{W}_{\text{down}}$ for dimension reduction. We observe that some components reach convergence with magnitude of gradient changes below 10^{-3}, while others maintain substantial magnitude of gradient changes throughout, as shown in Figure 1. This pattern holds across both language models and ViT architectures.

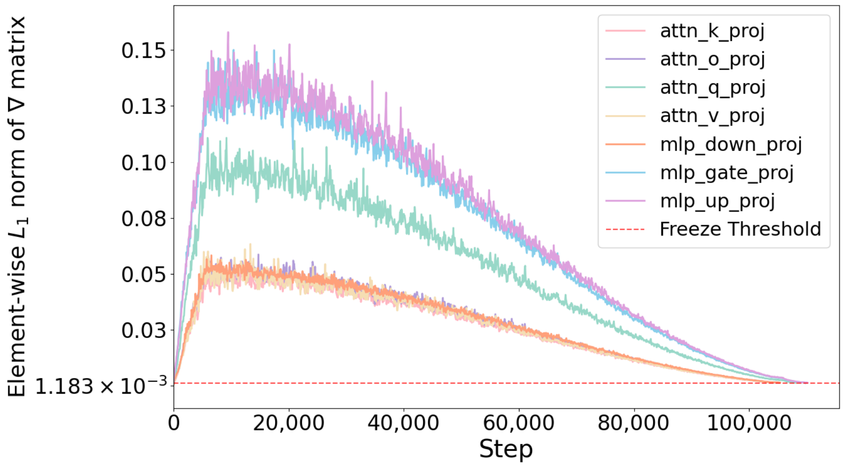


Figure 1: Element-wise L_1 norms for the gradient matrix of components in layer 7 for Qwen3-0.6B Qwen et al. (2025). Each step consists of processing one training batch through the complete forward pass, loss computation, backpropagation, and parameter update cycle. The seven tracked matrices comprise attention projections ($\mathbf{W}_q^{(7)}$, $\mathbf{W}_k^{(7)}$, $\mathbf{W}_v^{(7)}$, $\mathbf{W}_o^{(7)}$) and MLP components ($\mathbf{W}_{\text{gate}}^{(7)}$, $\mathbf{W}_{\text{up}}^{(7)}$, $\mathbf{W}_{\text{down}}^{(7)}$). MLP projections exhibit 2 to 3$\times$ higher gradient magnitudes than attention projections throughout training, with $\mathbf{W}_{\text{up}}^{(7)}$ and $\mathbf{W}_{\text{down}}^{(7)}$ maintaining the largest magnitude of gradient changes. The red dotted line indicates our convergence threshold $\tau = 1.183 \times 10^{-3}$.

This disparity motivates *GradES*, our gradient-based early stopping strategy that operates at the matrix level. Rather than requiring expensive validation passes, *GradES* leverages gradient information already computed during backpropagation to monitor each matrix $\mathbf{W}^{(l)} \in \{\mathbf{W}_q^{(l)}, \mathbf{W}_k^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{W}_o^{(l)}, \mathbf{W}_{\text{gate}}^{(l)}, \mathbf{W}_{\text{up}}^{(l)}, \mathbf{W}_{\text{down}}^{(l)}\}$ independently, where l denotes the layer. As illustrated in Figure 2, our architecture monitors the L1 norm changes for each component within the transformer layers. When a matrix's magnitude of gradient changes falls below threshold τ, the component is frozen (marked with lock icons in Figure 2), we stop its training while maintaining gradient flow for proper backpropagation, transforming early stopping from a binary termination decision into a continuous regularization mechanism. This selective stopping allows components with higher gradient activity to continue learning while converged components remain fixed. The selection of threshold τ is a critical hyperparameter whose configuration is detailed in Section A.7.

Our experiments across five LLMs, varied from 0.6B to 14B parameters, demonstrate that *GradES* reduces fine-tuning time by 50% while maintaining or improving accuracy across eight benchmarks. Attention projections consistently stabilize 2–3 times faster than MLP components, with key and value projections stopping earliest which is a pattern that validates our component specific (Attention or MLP) approach over early stopping. *GradES* can be seamlessly integrated with optimizers such as Adam Kingma & Ba (2017), SGD Robbins & Monro (1951), and parameter-efficient fine-tuning methods such as LoRA Hu et al. (2022), while complementing weight decay by preventing overfitting in converged components as weight decay continues, regularizing active components.

Contributions. We make the following contributions:

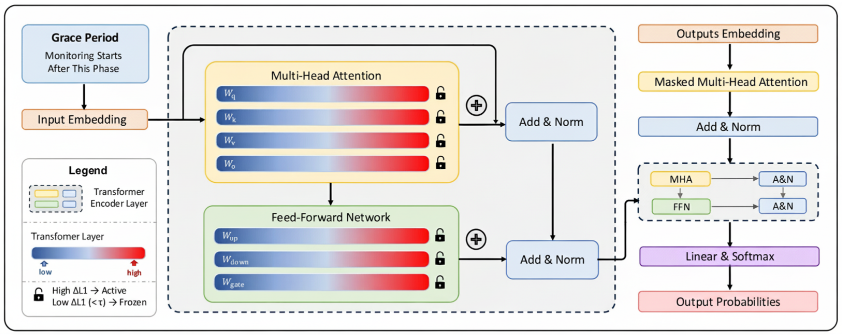


Figure 2: *GradES* architecture. Color indicate gradient magnitude changes for each weight matrix. Components with low gradient changes (below threshold τ) are stopped (lock icons) while others continue training.

- We propose *GradES*, a gradient-based early stopping method designed specifically for transformer and vision transformer architectures, eliminating expensive validation inference used for classic early stopping.
- We identified that gradient-based early stopping serves as an effective regularization technique, preventing overfitting in fast-converging Transformers' components while maintaining learning capacity in not yet converged completely, achieving improved accuracy up to 1.2% on language tasks and 3.88% on multimodal benchmarks and training time speed up of 1.57–7.22×.
- We validate the compatibility of *GradES* with diverse optimization algorithms (e.g., Adam, SGD) and parameter-efficient fine-tuning methods (e.g., LoRA), showing consistent acceleration across training paradigms while preserving their respective computational and memory efficiency advantages.
- We release our implementation as an open-source repository and PyPi package. It requires minimal modifications to existing training pipelines.

2 Related Work

2.1 Transformer and VLM

The Transformer architecture Vaswani et al. (2017) consists of distinct weight matrices for queries, keys, values, output projections, and MLP components. Vision-Language Models (VLMs) have evolved from contrastive learning frameworks such as CLIP Radford et al. (2021), which learns joint embeddings for zero-shot visual recognition, to architectures that enable complex multimodal reasoning. LLaVA Liu et al. (2023) integrated frozen vision encoders with large language models (LLMs) through learnable projection layers, refined by state-of-the-art models like Qwen2.5-VL-7B Bai et al. (2025). Concurrently, efforts toward computational efficiency have produced frameworks like nanoVLM Wiedmann et al. (2025). Interestingly, language encoders converge faster compared to vision layers which require extended optimization. Even within the language layers, attention and FFN layers converge at disparate rates. This variability motivates our *GradES* algorithm.

2.2 Parameter-Efficient Fine-tuning and Adaptive Training

LoRA Hu et al. (2022) employs low-rank decomposition $\mathbf{BA}$ where $r \ll \min(d_{\text{in}}, d_{\text{out}})$ (10,000× parameter reduction). A group of promising theoretical works established connections between gradient convergence patterns and optimization dynamics in neural networks. Arora et al. Arora et al. (2019) analyzes gradient descent convergence in deep linear networks, while Chatterjee et al. Chatterjee (2022) extends these results to general deep architectures, demonstrating that different components exhibit distinct convergence trajectories. Nguegnang et al. Nguegnang et al. (2021) characterizes convergence rates for gradient descent in linear networks, providing theoretical foundations for component convergence monitoring. The role of gradient-based stopping criteria has been formalized by Patel et al. Patel (2021), who establish strong convergence guarantees when magnitude of gradient changes fall below thresholds. Additionally, Gess et al. Gess & Kassing (2024) demonstrate exponential convergence rates for momentum methods in over parameterized settings, suggesting that convergence patterns are robust across different optimizers.

Building on these theoretical insights, several methods exploit convergence patterns for training efficiency. AutoFreeze Liu et al. (2021) accelerates BERT fine-tuning by 2.55× through automatic layer freezing patterns, though it operates only at layer granularity. AFLoRA Liu et al. (2024) introduces gradual freezing of LoRA adapters using convergence scores, achieving 9.5× parameter reduction and 1.86× speedup. LoRAF Zhang et al. (2025) prevents catastrophic forgetting by selectively freezing LoRA matrices based on their convergence state, using 95% fewer parameters than standard LoRA. However, these approaches either require coarse layer level decisions (AutoFreeze), additional architectural components like routers or masks (AFLoRA, LoRAF), or coupling with specific fine-tuning methods.

GradES advances this line of work by directly leveraging the heterogeneous convergence phenomenon Foster et al. (2018) at the granularity of individual weight matrices. *GradES* uses magnitude of gradient changes to detect when components converge Gao et al. (2024). It then stops individual components as they reach convergence, adapting to each component's learning rate without requiring schedules.

2.3 Early Stopping

Early stopping represents an established regularization technique with extensive theoretical and practical foundations. Prechelt et al. Prechelt (2012) provided practical guidelines for validation-based stopping criteria, while theoretical analyses by Yao et al. Yao et al. (2007) demonstrated that early stopping acts as implicit regularization in gradient descent learning. Recent theoretical advances by Ji et al. Ji et al. (2021) have proven the consistency of early stopping in neural networks, establishing rigorous convergence guarantees. Beyond traditional validation-based approaches, gradient-based early stopping by Mahsereci et al. Mahsereci et al. (2017) and Pflug et al. Pflug (1990) monitors magnitude of gradient changes to detect convergence without requiring validation data. Recent work has explored more sophisticated stopping strategies: instance dependent early stopping by Yuan et al. Yuan et al. (2025b) adapts termination per training example to reduce overfitting on easy samples, while correlation based methods by Ferro et al. Ferro et al. (2024) combine multiple online indicators for more robust stopping decisions. Methods for noisy settings have also emerged, including strategies for learning with label noise by Bai et al. Bai et al. (2021) and stopping without validation data by Yuan et al. Yuan et al. (2025a).

All these approaches apply stopping decisions globally, either terminating all training simultaneously or operating at the instance level. *GradES* differs fundamentally by applying gradient-based convergence detection at the individual weight matrix level within transformers, allowing different components to stop at their natural convergence points while others continue learning. This approach will effectively bridge early stopping with fine-grained regularization.

3 Method

3.1 GradES Algorithm

We analyzed the magnitude of gradient changes of weight matrices across Transformer layers during fine-tuning and discovered different convergence rates for different components within Transformers' Layers. For each weight matrix $\mathbf{W}^{(l)}$ in layer $l \in \{1, \dots, L\}$, we track the element-wise L_1 norm of gradients at training step t, as it is presented in Equation 1.

$$G_{\mathbf{W}}^{(l)}(t) = \|\nabla \mathbf{W}_t^{(l)} - \nabla \mathbf{W}_{t-1}^{(l)}\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |(\nabla \mathbf{W}_t^{(l)})_{ij} - (\nabla \mathbf{W}_{t-1}^{(l)})_{ij}| \qquad (1)$$

where $\mathbf{W}^{(l)} \in \{\mathbf{W}_q^{(l)}, \mathbf{W}_k^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{W}_o^{(l)}, \mathbf{W}_{\text{gate}}^{(l)}, \mathbf{W}_{\text{up}}^{(l)}, \mathbf{W}_{\text{down}}^{(l)}\}$ represents the attention projection matrices ($\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, $\mathbf{W}_o$) and MLP weight matrices ($\mathbf{W}_{\text{gate}}$, $\mathbf{W}_{\text{up}}$, $\mathbf{W}_{\text{down}}$) within each Transformer layer. The metric $G_{\mathbf{W}}^{(l)}(t)$ quantifies the total gradient flow through each component, revealing distinct convergence trajectories that motivate our gradient-based component level early stopping strategy.

Algorithm 1 *GradES*: Early Stopping based on Absolute Change of Gradient Matrix

```
Input: Pre-trained model $\mathcal{M}$ with $L$ layers, dataset $\mathcal{D}$, total steps $T$, grace period ratio $\alpha$, threshold
    $\tau$, learning rate $\eta$
Output: Fine-tuned model $\mathcal{M}'$
 1: Initialize: All parameters trainable, frozen set $\mathcal{F} \leftarrow \emptyset$
 2: Initialize: Previous gradients $\nabla \mathbf{W}_{t-1} \leftarrow 0$ for all $\mathbf{W}$
 3: $t_{\text{grace period}} \leftarrow \lceil \alpha \cdot T \rceil$                ▷ Start monitoring after grace period
 4: The grace period is computed as a fraction $\alpha$ of the total training steps $T$.
 5: for training step $t = 1$ to $T$ do
 6:     Sample $\mathcal{B} \sim \mathcal{D}$; compute loss $\mathcal{L}(\mathcal{B})$ and gradients
 7:     if $t > t_{\text{grace period}}$ then                              ▷ Monitor after grace period
 8:         for each $\mathbf{W} \in$ all layers : $\mathbf{W} \notin \mathcal{F}$ do
 9:             $G_{\mathbf{W}}(t) = \sum_{i,j} |(\nabla \mathbf{W}_t)_{ij} - (\nabla \mathbf{W}_{t-1})_{ij}|$
10:             if $G_{\mathbf{W}}(t) < \tau$ then $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{W}\}$
11:     for each projection matrix $\mathbf{W}$ do
12:         if $\mathbf{W} \notin \mathcal{F}$ then                             ▷ Update only active parameters
13:             $\mathbf{W} \leftarrow \mathbf{W} - \eta \cdot \nabla \mathbf{W}$
14:         else
15:             Skip update (but gradient still flows through)
16:     for each $\mathbf{W} \in$ all layers do                          ▷ Store gradients for next step
17:         $\nabla \mathbf{W}_{t-1} \leftarrow \nabla \mathbf{W}_t$
18:     if all parameters frozen then                              ▷ i.e., all $\mathbf{W} \in \mathcal{F}$
19:         break                                                   ▷ Early stopping
20: $\mathcal{M}' \leftarrow$ Update model with modified projection matrices $\mathbf{W}$
21: return Fine-tuned model $\mathcal{M}'$
```

Algorithm 1 formalizes, *GardES*, our gradient-based early stopping procedure. *GardES* algorithm introduces three key innovations that distinguish it from traditional early stopping approaches:

Component-level convergence detection. Unlike conventional methods that monitor global validation accuracy, *GradES* tracks individual weight matrices $\mathbf{W}^{(l)} \in \{\mathbf{W}_q^{(l)}, \mathbf{W}_k^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{W}_o^{(l)}, \mathbf{W}_{\text{gate}}^{(l)}, \mathbf{W}_{\text{up}}^{(l)}, \mathbf{W}_{\text{down}}^{(l)}\}$ within each layer l. We employ the L_1 gradient norm $G_{\mathbf{W}}(t) = \|\nabla \mathbf{W}\|_1$ as our convergence metric, chosen for its computational efficiency compared to L_2 norms. When $G_{\mathbf{W}}(t) < \tau$, we consider the component converged stop updating the weight matrix (lines 8-14).

Adaptive grace period strategy. The initial $t_{\text{grace period}} = \lceil \alpha T \rceil$ steps (with $\alpha = 0.5$ in our experiments) allow all components to escape their pre-trained initialization before convergence monitoring begins. This prevents terminating training of components prematurely that may appear initially converged but require substantial adaptation for the downstream task. The grace period duration depends on the total training examples and scales proportionally with the total training budget T.

Gradient flow preservation. A critical design choice is maintaining gradient computation through Converged matrices (line 12). While converged components do not receive parameter updates (lines 17-22), they continue to propagate gradients to earlier layers. This ensures that active components receive proper gradient signals throughout training, preventing the gradient flow disruption that would occur with complete component removal, like pruning.

The algorithm terminates when all components satisfy the convergence criterion (Line 24), eliminating unnecessary computation on converged parameters. We provide formal convergence guarantees and theoretical analysis in Appendix A.2.

4 Results

4.1 Accuracy on Benchmarks

We evaluate *GradES* against standard full-parameter (FP) fine-tuning and LoRA across five language models ranging from 0.6B to 14B parameters on eight commonsense reasoning benchmarks. As shown in Table 5, *GradES* consistently improves upon the baseline early stopping (ES) method across both fine-tuning methods (FP and LoRA). For full-parameter fine-tuning on larger models (14B), full-parameter with *GradES* achieves the highest average accuracy on Qwen3 (90.81%) and Phi4 (91.94%), demonstrating consistent improvements over full-parameter fine-tuning (90.80% and 91.93% respectively). Its impact is more significant on smaller models, where LoRA (ES) and LoRA (GradES) on Qwen3 0.6B achieve 67.37% and 67.30% average accuracy, substantially outperforming standard full-parameter methods (∼66.5%). Notably, the choice of base fine-tuning method (Full-parameter vs. LoRA) exhibits its model-independent behavior, while full-parameter fine-tuning methods perform competitively on larger models (Qwen3 14B, Phi4 14B), LoRA variants showed higher accuracy on mid-sized models, with LoRA (ES) achieving 86.27% on Mistral-7B compared to 75.80% for standard full-parameter fine-tuning, a remarkable 10.47 percentage point improvement. *GradES* shows particular strength on specific benchmarks, achieving best results on Winograde compared to other methods (Qwen3 14B: 84.77%), PIQA (Phi4 14B: 92.60%).

Table 3 presents *GradES* performance on vision-language tasks using Qwen2.5-VL-7B. *GradES* consistently improves both full-parameter and LoRA fine-tuning, with LoRA+GradES achieving the highest average accuracy (70.6%). LoRA methods substantially outperform full-parameter approaches on image captioning (54.44% vs. 41.61% for COCO Captions), while LoRA+GradES attains best performance on VQAv2 (81.24%). These results demonstrate that *GradES* effectively enhances vision-language model performance, particularly when combined with PEFT methods.

Table 4 demonstrates *GradES*'s effectiveness on nanoVLM training across diverse multimodal benchmarks. Training with *GradES* achieves substantial improvements over standard training, with an average accuracy increase of 3.88 percentage points (34.70% vs. 30.82%). Most notably, *GradES* yields significant gains on Fine-grained Perception (+7.42% and Logical Reasoning (+8.35%), while maintaining consistent improvements across all evaluated domains. These results confirm *GradES*'s ability to enhance vision-language model training, particularly for tasks requiring detailed visual understanding and complex reasoning.

4.2 Training Efficiency

Tables 6 and 7 present computational efficiency metrics across language and vision-language models. *GradES* demonstrates consistent efficiency improvements over both baseline and traditional early stopping (ES) approaches. For language models, FP+GradES achieves speedups of 1.32–1.64× while reducing FLOPs by 29–45%, with the most significant gains on larger models (Qwen3-14B: 1.51× speedup, 0.55× FLOPs). In contrast, traditional ES paradoxically slows training (0.59–0.76×) despite FLOPs reduction, highlighting the overhead of frequent validation and convergence monitoring. Vision-language models show similar trends, with FP+GradES achieving 1.17× speedup and 12% FLOPs reduction on Qwen2.5-VL-7B.

The efficiency benefits extend to parameter-efficient methods, where LoRA+GradES consistently outperforms standard LoRA across all models. For language models, LoRA+GradES achieves speedups of 2.66–2.87× over full-parameter baselines while reducing LoRA's computational overhead from 2.34–2.43× to 1.88–2.29× FLOPs. On vision-language tasks, LoRA+GradES delivers the highest speedup (1.80×), reducing training time by 44%. Remarkably, these computational savings preserve model quality—*GradES* variants achieve the highest accuracies while requiring substantially fewer FLOPs than standard fine-tuning. These results establish gradient-guided early stopping as a practical solution for resource-constrained deployment, demonstrating consistent efficiency gains across model architectures (Qwen, Phi, Llama, Mistral), scales (0.6B–14B parameters), and modalities (language and vision-language).

5 Discussion

5.1 Convergence Patterns Across Different Models

Figure 3 shows the progression of the matrix that converged across different model scales during training. After a warm-up period of $\alpha = 1000$ steps, our method begins stopping converged components based on magnitude of gradient changes thresholds τ (model-specific values detailed in Appendix A.7).

The difference in convergence rate reflects distinct architectural behaviors. Larger models (7B-14B) exhibit rapid convergence, with the majority of components converged by step 1400—approximately 40% through training. In contrast, the smaller Qwen-0.6B model demonstrates delayed convergence, with no components meeting the stopping criteria until step 1600.

Notably, the threshold τ varies significantly between training methods, and full fine-tuning requires larger thresholds compared to LoRA adaptation.

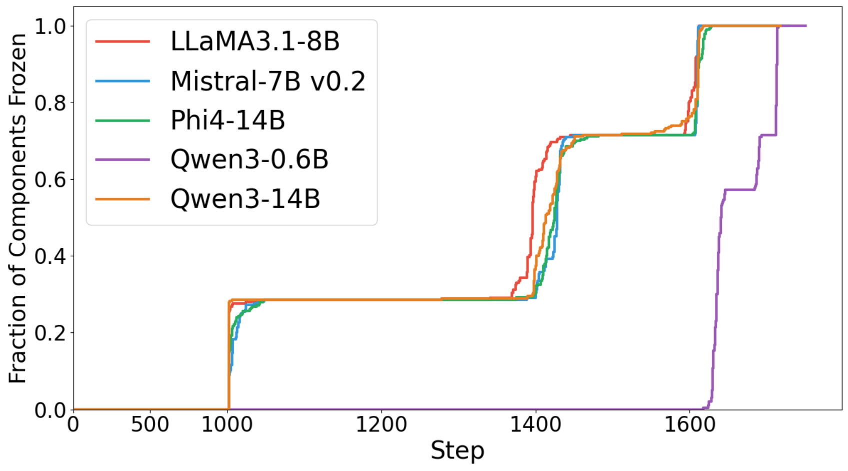


Figure 3: Cumulative converged components during training across model scales. Fraction of weight matrices converged over time for five different LLMs.

5.2 Attention versus MLP components

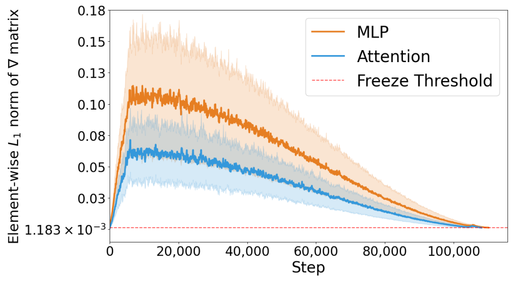
Figure 4a presents the gradient norms $|\nabla W|$ across all weight matrices during fine-tuning of the Qwen-0.6B model. We compute the element-wise L1 norm for each weight matrix and report averaged values for two architectural components: MLP matrices ($\mathbf{W}_{\text{up}}$, $\mathbf{W}_{\text{down}}$; orange) and attention projection matrices ($\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$, $\mathbf{W}_o$; blue).

We have two key observations. First, the gradient trend reflects the cosine learning rate schedule: initial warmup drives increasing magnitude of gradient changes as the model adapts to the task distribution, while the subsequent cosine decay produces monotonically decreasing magnitude of gradient changes, enabling smooth convergence from exploration to refinement.

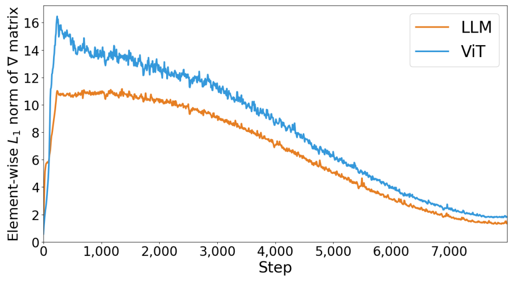
Second, and more critically for our method, MLP weight matrices consistently maintain larger magnitude of gradient changes compared to attention projection matrices throughout training. This persistent gap indicates that MLP parameters require more steps to converge, suggesting inefficiency under uniform training strategies. This observation directly motivates our approach: by dynamically allocating computational resources proportional to magnitude of gradient changes, we can accelerate convergence of slower learning components while maintaining overall model accuracy.

5.3 LLMs vs VLMs

Perhaps a similarly interesting discovery we made is with the vision language model. Figure 4 illustrates the training dynamics of the vision and language transformer components separately. Our



(a) Magnitude of gradient changes during Qwen-0.6B fine-tuning.



(b) Magnitude of gradient changes evolution during NanoVLM training.

Figure 4: Comparison of magnitude of gradient changes patterns across different model architectures during fine-tuning. Element-wise L1 norms of weight gradients averaged across layers are shown for (a) Qwen-0.6B with MLP matrices (orange) and attention projections (blue), where MLP matrices consistently exhibit larger magnitude of gradient changes throughout training, indicating slower convergence and motivating targeted computational allocation; and (b) NanoVLM with Vision layers matrices (blue) and attention projections (orange), where ViT matrices consistently exhibit larger magnitude of gradient changes throughout training.

analysis reveals an difference in convergence pattern like that between attention and MLP components: while the language transformer reaches convergence early in training, the vision transformer converges slower. However, extending training with a decreasing learning rate schedule leads to overfitting and subsequent accuracy degradation.

Our approach demonstrates significant accuracy improvements over full parameter fine-tuning. With only 8,000 training examples, we achieve 34.7% accuracy—representing a 3.92% absolute improvement over the baseline full fine-tuning approach. Remarkably, the original Nano-VLM requires approximately 17,000 examples to reach comparable performance levelsWiedmann et al. (2025), indicating that our method reduces the required training data by more than 50% while maintaining competitive accuracy.

5.4 Combination with Parameter-Efficient Fine-Tuning Methods

GradES combines particularly well with parameter-efficient fine-tuning methods such as LoRA. As shown in Table 6, LoRA+GradES achieves the fastest training times across all models, reducing training time to just 0.14–0.38× of standard full-parameter fine-tuning. This dramatic speedup comes from optimizing two independent dimensions, (i) LoRA reduces the parameter space through low-rank weight decomposition, (ii) while *GradES* reduces training iterations by detecting convergence through gradient monitoring. For example, on Qwen3 0.6B, LoRA+GradES completes training in just 907 seconds compared to 6,550 seconds for standard fine-tuning, a 7.22× speedup, while achieving better accuracy (67.30% vs. 66.53%).

The benefits of combining *GradES* with LoRA become especially clear when compared to standard early stopping. While LoRA+ES actually increases training time by 2.97–6.90× due to validation overhead, LoRA+GradES maintains or improves LoRA's efficiency. This difference suggests that gradient-based early stopping criteria work particularly well in LoRA's constrained optimization landscape, where the reduced parameter space provides clearer convergence signals. Although LoRA methods inherently require more FLOPs per iteration than full-parameter training (2.07–2.43×), *GradES* compensates by reducing the total number of iterations needed. The result is a practical for deployment due to the extremely fast training times with reasonable computational costs, making LoRA+GradES the optimal choice for fine-tuning.

5.5 GradES versus Classic Early Stopping

A fundamental limitation of classic early stopping is the computational expense of requiring complete forward passes for every validation step. In contrast, *GradES* reuses gradient information from backpropagation, yielding substantial computational savings. As shown in Table 6, it achieves up to 6.79× speedup compared to classic early stopping on Qwen3-0.6B (907s vs 6,155s for LoRA fine-tuning), while maintaining comparable accuracy. The minimal accuracy difference—67.30% for *GradES* versus 67.37% for classic early stopping as shown in Table 5—does not justify the significant computational overhead. Across all five models tested, *GradES* consistently reduces training time by 35–66% while achieving a 45–71% reduction in FLOPs compared to baseline full fine-tuning, making it particularly valuable for limited resource settings.

Furthermore, classic early stopping employs a model-wide convergence criterion that is not suitable for Transformer architectures. As demonstrated in Figure 1, different weight matrices within transformer layers exhibit varying convergence rates. Model-wide early stopping fails to account for this heterogeneity, potentially leading to overfitting in some parameters while underfitting in others. *GradES* addresses this limitation by enabling component-specific convergence criteria, allowing MLP and attention components to be trained independently until each reaches its optimal stopping point. This ensures all weight matrices converge according to appropriate criteria rather than being halted or unnecessarily extended based on global validation metrics.

6 Limitation

While *GradES* demonstrates substantial efficiency gains, some limitations exists. First, gradient monitoring incurs approximately 3% computational overhead, though this is negligible compared to the 1.57–7.22× training time speed up achieved. Second, the convergence threshold τ requires manual tuning across different models and tasks, with full-parameter fine-tuning requiring larger thresholds than LoRA and model scale affecting optimal values. For VLMs, we observe that vision and language components convergence at a different rate, motivating component specific thresholds for optimal performance. Third, our current implementation is achieved without patience mechanisms common in early stopping, potentially leading to premature convergence decisions. Additionally, while we validate *GradES* on transformer architectures, its applicability to other neural architectures, such as convolutional networks, graph neural networks, and emerging architectures like state space models, remains unexplored.

7 Conclusion and Future work

We presented *GradES*, a gradient-based early stopping strategy that addresses the computational inefficiencies of validation-based approaches for fine-tuning transformers. By monitoring magnitude of gradient changes at the component level and selectively halting updates for converged components, *GradES* achieves up to 7.22× speed up in training time and 45% reduction in FLOPs while improving model accuracy across eight commonsense reasoning benchmarks. Our experiments across five model architectures (0.6B–14B parameters) demonstrate that *GradES* seamlessly integrates with both full parameter fine-tuning and parameter-efficient methods like LoRA, with the combination achieving remarkable efficiency, completing fine-tuning in as little as 14% of the baseline time. The key insight underlying our approach is that different transformer components exhibit distinct convergence behaviors during fine-tuning. The method's ability to eliminate costly validation passes while providing precise convergence control represents a practical advancement for deploying large language models with limited resources.

Several promising directions emerge for future work. Automatic threshold selection through gradient statistics or meta learning could eliminate manual tuning, while incorporating patience parameters would allow components to temporarily violate convergence criteria before early stopping. Most ambitiously, applying early stopping to pretraining could significantly reduce the massive computational costs of foundation model development, while theoretical analysis of convergence criteria would provide principled guidelines for threshold selection and accuracy guarantees. As the scale of language models continues to grow, optimization strategies like *GradES* will become increasingly critical for making these powerful models accessible to the broader research community.

Ethics Statement

We have read and adhered to the ICLR Code of Ethics. This work reduces computational costs for fine-tuning transformers, with no negative societal impacts anticipated. All experiments used publicly available datasets and models with no human subjects involved.

Reproducibility Statement

We provide full implementation details in Section A.7 (threshold selection) and Section A.5 (experimental setup). All datasets are publicly available. Source code for GradES, including integration with Adam, SGD, and LoRA, will be released as supplementary materials upon acceptance.

Acknowledgments

We used large language models to assist with the writing and editing of this paper. The authors declare no conflicts of interest.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, and Others. Qwen2.5-vl technical report, 2025. URL `https://arxiv.org/abs/2502.13923`.

Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels. *arXiv preprint arXiv:2106.15853*, 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Sourav Chatterjee. Convergence of gradient descent for deep neural networks. *arXiv preprint arXiv:2203.16462*, 2022.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server, 2015. URL `https://arxiv.org/abs/1504.00325`.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Manuel Vilares Ferro, Yerai Doval Mosquera, Francisco J. Ribadas Pena, and Victor M. Darriba Bilbao. Early stopping by correlating online indicators in neural networks. *arXiv preprint arXiv:2402.02513*, 2024.

Dylan J. Foster, Ayush Sekhari, and Karthik Sridharan. Uniform convergence of gradients for non-convex learning and optimization. *arXiv preprint arXiv:1810.11059*, 2018.

Wenzhi Gao, Ya-Chi Chu, Yinyu Ye, and Madeleine Udell. Gradient methods with online scaling. *arXiv preprint arXiv:2411.01803*, 2024.

Benjamin Gess and Sebastian Kassing. Exponential convergence rates for momentum stochastic gradient descent in the overparametrized setting. *arXiv preprint arXiv:2302.03550*, 2024.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering, 2017. URL `https://arxiv.org/abs/1612.00837`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019. URL `https://arxiv.org/abs/1902.09506`.

HuggingFace. Hugging face hub, 2020. URL `https://huggingface.co/`. AI model and dataset repository platform.

Ziwei Ji, Justin D. Li, and Matus Telgarsky. Early-stopped neural networks are consistent. In *Advances in Neural Information Processing Systems*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

Bo Li, Peiyuan Zhang, Kaichen Zhang, Fanyi Pu, Xinrun Du, Yuhao Dong, Haotian Liu, Yuanhan Zhang, Ge Zhang, Chunyuan Li, and Ziwei Liu. LMMs-Eval: Accelerating the development of large multimodal models, March 2024. URL `https://github.com/EvolvingLMMs-Lab/lmms-eval`.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL `https://arxiv.org/abs/2304.08485`.

Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*, 2021.

Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Anthony Beerel. AFLoRA: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models. *arXiv preprint arXiv:2403.13269*, 2024.

Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*, 2017.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*, 2021.

Gabin Maxime Nguegnang, Holger Rauhut, and Ulrich Terstiege. Convergence of gradient descent for learning linear neural networks. *arXiv preprint arXiv:2108.02040*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, et al. PyTorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Vivak Patel. Stopping criteria for, and strong convergence of, stochastic gradient descent on bottou-curtis-nocedal functions. *arXiv preprint arXiv:2004.00475*, 2021.

Georg Ch Pflug. Non-asymptotic confidence bounds for stochastic approximation algorithms with constant step size. *Monatshefte für Mathematik*, 110(3-4):297–314, 1990.

Lutz Prechelt. Early stopping — but when? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*. Springer, 2012.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, and Others. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL `https://arxiv.org/abs/2103.00020`.

Reza Rawassizadeh. *Machine Learning and Artificial Intelligence: Concepts, Algorithms and Models*. 1 edition, March 2025.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *arXiv preprint arXiv:2205.10770*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Luis Wiedmann, Aritra Roy Gosthipaty, and Andrés Marafioti. nanovlm. `https://github.com/huggingface/nanoVLM`, 2025.

Xinhao Yao, Hongjin Qian, Xiaolin Hu, Gengze Xu, Wei Liu, Jian Luan, Bin Wang, and Yong Liu. Theoretical insights into fine-tuning attention mechanism: Generalization and optimization. *arXiv preprint arXiv:2410.02247*, 2025.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Suqin Yuan, Lei Feng, and Tongliang Liu. Early stopping against label noise without validation data. *arXiv preprint arXiv:2502.07551*, 2025a.

Suqin Yuan, Runqi Lin, Lei Feng, Bo Han, and Tongliang Liu. Instance-dependent early stopping. *arXiv preprint arXiv:2502.07547*, 2025b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. LoRA without forgetting: Freezing and sparse masking for low-rank adaptation. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. LMMs-Eval: Reality check on the evaluation of large multimodal models, 2024. URL `https://arxiv.org/abs/2407.12772`.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. Revisiting few-sample BERT fine-tuning. *arXiv preprint arXiv:2006.05987*, 2021.

A Appendix

A.1 Theoretical Analysis: Selection of Norm

The choice of norm for gradient monitoring significantly impacts both computational efficiency and convergence detection reliability. For a gradient matrix $G \in \mathbb{R}^{m\times n}$, we consider four matrix norm candidates: element-wise L_1 ($\|G\|_{1,1} = \sum_{i,j} |g_{ij}|$), Frobenius ($\|G\|_F = \sqrt{\sum_{i,j} g_{ij}^2}$), spectral ($\|G\|_2 = \sigma_{\max}(G)$), and subordinate L_∞ ($\|G\|_\infty = \max_i \sum_j |g_{ij}|$).

We select the element-wise L_1 norm based on computational efficiency and convergence properties. The L_1 norm requires only $O(mn)$ operations through element-wise summation, avoiding expensive computations such as square roots (Frobenius) or singular value decomposition (spectral, $O(mn\min(m,n))$). Furthermore, the L_1 norm provides a stronger convergence criterion through the following theorem:

Theorem 1. *For any matrix $A \in \mathbb{R}^{m\times n}$, the elementwise L_1 norm provides an upper bound for commonly used matrix norms:*

$$\|A\|_2 \le \|A\|_{1,1} \tag{2}$$
$$\|A\|_F \le \|A\|_{1,1} \tag{3}$$
$$\|A\|_\infty \le \|A\|_{1,1} \tag{4}$$
$$\|A\|_1 \le \|A\|_{1,1} \tag{5}$$

where $\|A\|_1 = \max_j \sum_i |a_{ij}|$ denotes the subordinate L_1 norm (maximum column sum).

Proof. For equation 2, the spectral norm satisfies $\|A\|_2 \le \sqrt{\|A\|_1 \cdot \|A\|_\infty}$ by the well-known inequality for induced norms. Since $\|A\|_1 = \max_j \sum_i |a_{ij}| \le \sum_{i,j} |a_{ij}| = \|A\|_{1,1}$ and similarly $\|A\|_\infty \le \|A\|_{1,1}$, we have:

$$\|A\|_2 \le \sqrt{\|A\|_1 \cdot \|A\|_\infty} \le \sqrt{\|A\|_{1,1} \cdot \|A\|_{1,1}} = \|A\|_{1,1}$$

For equation 3, note that $a_{ij}^2 \le |a_{ij}| \cdot \max_{k,l} |a_{kl}| \le |a_{ij}| \cdot \|A\|_{1,1}$ for any element. Summing over all indices:

$$\|A\|_F^2 = \sum_{i,j} a_{ij}^2 \le \sum_{i,j} |a_{ij}| \cdot \|A\|_{1,1} = \|A\|_{1,1}^2$$

Therefore, $\|A\|_F \le \|A\|_{1,1}$.

For equation 4, the subordinate L_∞ norm is the maximum row sum: $\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^n |a_{ij}|$. Since the maximum row sum cannot exceed the sum of all elements: $\|A\|_\infty \le \sum_{i=1}^m \sum_{j=1}^n |a_{ij}| = \|A\|_{1,1}$.

For equation 5, similarly, the subordinate L_1 norm is the maximum column sum: $\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^m |a_{ij}| \le \sum_{i=1}^m \sum_{j=1}^n |a_{ij}| = \|A\|_{1,1}$. □

These relationships establish that monitoring the L_1 norm provides a universal upper bound for convergence detection. When $\|G\|_{1,1} < \tau$, we guarantee that all other standard matrix norms are also bounded by τ, ensuring robust convergence detection across multiple norm perspectives while maintaining linear computational complexity.

A.2 Convergence Properties

We provide theoretical guarantees for the convergence of Algorithm 1. Our analysis demonstrates that *GradES* converges to a stationary point of the loss function while ensuring computational efficiency through gradient based early stopping

Theorem 2 (Convergence of GradES)**.** *Consider Algorithm 1 applied to a loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ that is L-smooth and lower bounded by $\mathcal{L}^*$. With threshold $\tau > 0$, the algorithm satisfies:*

1. *The loss sequence $\{\mathcal{L}(t)\}_{t=1}^T$ is non-increasing after warm up*

2. *All Converged matrices satisfy* $\|\nabla_{\mathbf{W}}\mathcal{L}\|_{1,1} < \tau$ *at convergence*

3. *The algorithm terminates in finite time with* $\min_{t\in[T]} \|\nabla\mathcal{L}(t)\| \leq \tau$

Proof. **Part 1: Monotonic Loss Decrease.** After the warmup period ($t > 0.05T$), the cosine schedule ensures η_t is decreasing. For any active parameter $\mathbf{W} \notin \mathcal{F}$ at step t, the update rule yields:

$$\mathcal{L}(\mathbf{W}_{t+1}) \leq \mathcal{L}(\mathbf{W}_t) + \langle \nabla\mathcal{L}(\mathbf{W}_t), \mathbf{W}_{t+1} - \mathbf{W}_t\rangle + \frac{L}{2}\|\mathbf{W}_{t+1} - \mathbf{W}_t\|^2 \tag{6}$$

$$= \mathcal{L}(\mathbf{W}_t) - \eta_t\|\nabla\mathcal{L}(\mathbf{W}_t)\|^2 + \frac{L\eta_t^2}{2}\|\nabla\mathcal{L}(\mathbf{W}_t)\|^2 \tag{7}$$

For Converged matrices $\mathbf{W} \in \mathcal{F}$, we have $\mathbf{W}_{t+1} = \mathbf{W}_t$, thus $\mathcal{L}(\mathbf{W}_{t+1}) = \mathcal{L}(\mathbf{W}_t)$. The cosine schedule with maximum value $\eta_0 < \frac{2}{L}$ ensures:

$$\mathcal{L}(t+1) \leq \mathcal{L}(t) - \sum_{\mathbf{W}\notin\mathcal{F}} \eta_t \left(1 - \frac{L\eta_t}{2}\right) \|\nabla_{\mathbf{W}}\mathcal{L}(t)\|^2 \tag{8}$$

Since $\eta_t \leq \eta_0 < \frac{2}{L}$ throughout training, the loss sequence is non-increasing.

Part 2: Converged matrices at Stationary Points. A parameter matrix $\mathbf{W}$ is frozen at step t_f when $\|\nabla_{\mathbf{W}}\mathcal{L}(t_f)\|_{1,1} < \tau$. Since Converged matrices receive no further updates:

$$\mathbf{W}_t = \mathbf{W}_{t_f} \quad \forall t > t_f \tag{9}$$

As shown in Part 1, the gradient magnitudes are non-increasing after warmup. Combined with the continuity of gradients:

$$\|\nabla_{\mathbf{W}}\mathcal{L}(t)\|_{1,1} \leq \|\nabla_{\mathbf{W}}\mathcal{L}(t_f)\|_{1,1} < \tau \quad \forall t > t_f > 0.05T \tag{10}$$

This ensures that once a parameter is frozen, its gradient remains below the threshold.

Part 3: Finite-Time Termination. Define the active parameter set at time t as $\mathcal{A}_t = \{\mathbf{W} : \mathbf{W} \notin \mathcal{F}_t\}$. The cardinality $|\mathcal{A}_t|$ is non-increasing since parameters can only transition from active to frozen.

Under the cosine schedule, as $t \to T$, we have $\eta_t \to 0$. In experiment, gradient magnitudes decrease monotonically after warmup. Therefore, there exists a finite $T^* < T$ such that either:

- All parameters satisfy $\|\nabla_{\mathbf{W}}\mathcal{L}\|_{1,1} < \tau$ and are frozen, or
- The cosine schedule drives $\eta_t\|\nabla\mathcal{L}(t)\| < \epsilon$ for arbitrarily small ϵ

In both cases, the algorithm effectively converges with $\min_{t\in[T]} \|\nabla\mathcal{L}(t)\| \leq \tau$. □

Corollary 3. *GradES achieves an* ϵ*-stationary point while potentially reducing computational cost by a factor proportional to* $|\mathcal{F}|/d$*, where* d *is the total number of parameters. The cosine schedule ensures smooth convergence without oscillations in gradient magnitudes.*

This analysis establishes that *GradES* maintains convergence guarantees under the practical cosine learning rate schedule used in our experiments. The threshold τ controls the trade-off between convergence accuracy and computational savings, while the 5% warmup period ensures stable gradient behavior before monitoring begins.

A.3 *GradES* for Low-Rank Adaptation

LoRA is one of the most common approaches used for fine-tuning. Since LoRA constrains parameters to a low-dimensional subspace, gradient dynamics in this reduced space exhibit fundamentally different convergence properties than full fine-tuning. When applying *GradES* to LoRA Hu et al. (2022) fine-tuning, we monitor gradient magnitudes in the low-rank space rather than the full parameter space. Let $l \in \{1, \ldots, L\}$ denote the layer index where L is the total number of layers. Within each transformer layer l, we apply LoRA decomposition to individual weight matrices

$\mathbf{W}^{(l)} \in \{\mathbf{W}_q^{(l)}, \mathbf{W}_k^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{W}_o^{(l)}, \mathbf{W}_{\text{gate}}^{(l)}, \mathbf{W}_{\text{up}}^{(l)}, \mathbf{W}_{\text{down}}^{(l)}\}$, where the first four correspond to attention projections and the latter three to MLP components.

For each weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, where $d_{\text{out}}, d_{\text{in}}$ are the output and input dimensions, in layer l, the LoRA weight is:

$$\mathbf{W}_{\text{adapted}}^{(l)} = \mathbf{W}_{\text{frozen}}^{(l)} + \mathbf{B}_{\mathbf{W}}^{(l)} \mathbf{A}_{\mathbf{W}}^{(l)} \tag{11}$$

where $\mathbf{B}_{\mathbf{W}}^{(l)} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $\mathbf{A}_{\mathbf{W}}^{(l)} \in \mathbb{R}^{r \times d_{\text{in}}}$ are the trainable low-rank matrices with rank $r \ll \min(d_{\text{out}}, d_{\text{in}})$.

For each individual LoRA matrix $\mathbf{W}$ in layer l, we track convergence by monitoring the combined gradient magnitude:

$$G_{\mathbf{W}}^{(l)}(t) = \|\nabla \mathbf{A}_{\mathbf{W}}^{(l)}\|_1 + \|\nabla \mathbf{B}_{\mathbf{W}}^{(l)}\|_1 \tag{12}$$

where $\nabla \mathbf{A}_{\mathbf{W}}^{(l)}$ and $\nabla \mathbf{B}_{\mathbf{W}}^{(l)}$ denote the gradients of the low-rank matrices at training step t, and $\|\cdot\|_1$ denotes the element-wise L_1 norm.

The early stopping operates at the matrix level, enabling precise control. After the grace period period $t > t_{\text{grace period}}$, we independently stop each LoRA matrix when:

$$G_{\mathbf{W}}^{(l)}(t) < \tau_r \quad \text{for } \mathbf{W} \in \{\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o, \mathbf{W}_{\text{gate}}, \mathbf{W}_{\text{up}}, \mathbf{W}_{\text{down}}\} \tag{13}$$

where τ_r is the convergence threshold adjusted for the reduced parameter count. Once a specific matrix reaches convergence, we stop updating its corresponding $\mathbf{A}_{\mathbf{W}}^{(l)}$ and $\mathbf{B}_{\mathbf{W}}^{(l)}$ while continuing to compute gradients through them for backpropagation. Training terminates when all LoRA matrices across all layers are frozen.

A.4 Ablation Studies

We first investigate the necessity of the grace period α. When we remove the grace period entirely ($\alpha = 0$), training terminates immediately after initialization. As illustrated in Figure 1, gradient norms across all components remain small during the warm-up phase, falling below the convergence threshold τ. This observation confirms that the grace period is crucial for our method.

Then we evaluate the impact of the grace period $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ and convergence threshold $\tau \in \{1.5, 3.0, 4.5, 6.0, 7.5, 9.0\}$ on both model performance and training efficiency. All experiments are conducted on the Qwen-14B model across eight benchmark tasks: BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, OpenBookQA, ARC-Challenge, and ARC-Easy. Table 1 reports the average evaluation accuracy across these benchmarks, while Table 2 presents the corresponding training times.

Our results reveal that optimal performance (92.81% average accuracy) is achieved with $\tau = 1.5$ and $\alpha = 0.5$, suggesting that a moderate grace period combined with a convergence threshold yields the best generalization. Conversely, the most significant speedup occurs at $\tau = 9.0$ and $\alpha = 0.1$.

Table 1: Accuracy values are shown as percentages for threshold (τ) and grace period (α) parameters, with the best result highlighted in bold.

τ / α	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**
1.5	63.98	58.55	76.67	67.58	**92.81**	82.96
4.5	57.54	70.43	51.47	61.07	53.80	82.96
7.5	49.81	68.63	61.87	61.43	73.56	80.11
9.0	46.45	67.89	66.52	58.48	68.38	77.78

A.5 Experimental Setup

A.5.1 Models

We evaluate *GradES* on the 5 most popular language models and 2 vision language model on huggingface HuggingFace (2020) (at the time of conducting this experiment) to represent different

Table 2: Fine-tuning time for threshold (τ) and grace period (α) parameters, with the fastest time highlighted in bold.

τ / α	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**
1.5	10726.74	10592.54	10629.99	10840.85	11187.48	11774.73
4.5	8925.71	9269.45	9317.26	9789.08	10273.89	11318.94
7.5	8038.52	8337.48	8599.51	9262.74	9962.62	11087.04
9.0	**7393.14**	7746.00	8036.02	9130.12	9477.76	10916.65

parameter scales, quantization strategies, and architectural designs. Our language model selection spans three orders of magnitude in parameter count (0.6B to 14B) to investigate the scalability of our approach. We conducted all experiments using 4-bit quantized models, due to hardware limitations of the experimental platform. Specifically, we employ: **Qwen3-14B** Qwen et al. (2025); **Microsoft Phi4-14B** Abdin et al. (2024); **Llama-3.1-8B-Instruct** Dubey et al. (2024); **Mistral-7B-Instruct-v0.3** Jiang et al. (2023); and the compact **Qwen3-0.6B** Qwen et al. (2025). To assess performance on multimodal tasks, we also evaluate two vision-language models: **Qwen2.5-VL-7B** Bai et al. (2025) and the smaller **nanoVLM** Wiedmann et al. (2025).

A.5.2 Benchmark and Evaluation Metrics

We evaluate *GradES* on eight widely used benchmarks covering different aspects of language understanding. For reasoning tasks, we use BoolQ Clark et al. (2019) for boolean question answering requiring complex reasoning, PIQA Bisk et al. (2020) for physical commonsense reasoning about everyday situations, SIQA Sap et al. (2019) for social commonsense reasoning about human interactions, and HellaSwag Zellers et al. (2019) for commonsense inference about plausible continuations. For knowledge-intensive task completion, we employ OpenBookQA Mihaylov et al. (2018) for science questions requiring multi-hop reasoning, ARC-Easy and ARC-Challenge Clark et al. (2018) for grade school science questions at two difficulty levels, and WinoGrande Sakaguchi et al. (2019) for pronoun resolution requiring commonsense knowledge. We measure both task accuracy and computational efficiency to provide a rounded view of *GradES*'s benefits. Average task accuracy is measured by accuracy on each benchmark's test set. For computational efficiency metrics, we track training time on identical hardware and floating point operations (FLOPs) computed using PyTorch profiler Paszke et al. (2019).

We evaluate vision transformers using the LMMs-Eval harness Zhang et al. (2024); Li et al. (2024) to assess nanoVLM training accuracy. We conduct fine-tuning experiments on Qwen2.5-VL-7B and evaluate performance across three established benchmarks: GQA Hudson & Manning (2019) for compositional visual reasoning, VQAv2 Goyal et al. (2017) for robust visual question answering, and COCO Captions Chen et al. (2015) for image captioning capabilities.

A.5.3 PEFT and Early Stopping Methods

We evaluate *GradES* against established fine-tuning paradigms to demonstrate its benefits. Our baseline methods comprise Full Parameter Fine-tuning (FP), which updates all model parameters without constraints; and LoRA Hu et al. (2022) to assess the composability of our method. In particular, we apply *GradES* to both full fine-tuning (FP+GradES) and LoRA (LoRA+GradES), yielding six distinct configurations for comprehensive evaluation. For validation-based early stopping baselines (FP+ES and LoRA+ES), we perform validation checks at 5% intervals throughout training. Terminating training when validation loss fails to improve for consecutive checkpoints. We presents the overhead of early stopping compared to *GradES* in 6. More detailed experiment settings and hyperparameter selection are provided in Appendix A.8.

A.6 Detailed Experimental Results

This section presents comprehensive experimental results supporting the findings discussed in the main paper. We provide detailed performance metrics, training times, and computational cost analyses across all evaluated models and benchmarks. Tables 3 and 4 present accuracy results for vision-language models, demonstrating the effectiveness of GradES across different multimodal tasks.

Table 5 contains the complete benchmark scores for five language models (Qwen3-14B, Phi4-14B, Qwen3-0.6B, Llama-3.1-8B, and Mistral-7B) across eight evaluation metrics. Tables 6 and 7 provide detailed computational efficiency measurements, including training times and FLOPs comparisons, highlighting the significant speedup and computational savings achieved by GradES across both full parameter and LoRA fine-tuning settings.

Table 3: Performance comparison of *GradES* against standard fine-tuning methods on vision-language benchmarks. Results show accuracy (%) for Qwen2.5-VL-7B across visual reasoning (GQA), question answering (VQAv2), and image captioning (COCO Cap) tasks.

Model	**Method**	**GQA**	**VQAv2**	**COCO Cap**	**Avg.**
Qwen2.5-VL-7B	Full Parameter	75.69	81.0	41.38	66.08
	FP+GradES	76.08	80.81	41.61	66.20
	LoRA	76.49	81.01	53.22	70.24
	LoRA+GradES	76.13	81.24	54.44	70.6

Table 4: Performance of *GradES* on nanoVLM training across multimodal reasoning benchmarks. Results show accuracy (%) for full-parameter fine-tuning with and without *GradES* across perception, reasoning, and knowledge-based tasks.

Benchmark	**Training**	**Training+GradES**
Coarse Perception	38.87	42.42
Fine-grained Perception	22.40	29.82
Instance Reasoning	36.07	36.42
Logical Reasoning	28.86	37.21
Math	27.60	28.60
Science & Technology	31.10	33.74
Avg.	**30.82**	**34.70**

A.7 Hyperparameter Configuration

We present comprehensive hyperparameter configurations to ensure experimental reproducibility. Tables 8–9 detail the training configurations for five language models—Qwen3-14B, Phi4-14B, Qwen3-0.6B, Llama-3.1-8B, and Mistral-7B—under both full-parameter (FP) fine-tuning and Low-Rank Adaptation (LoRA). For all experiments, we employ early stopping with a validation loss threshold of $\delta = 0.0005$, patience of 3 epochs, and validation performed at 5% intervals throughout training. Tables 10–11 additionally present configurations and efficiency metrics for vision-language model experiments.

A.8 Code Availability

We are committed to ensuring the reproducibility of our research. To facilitate this, we provide comprehensive resources:

Implementation. Our complete implementation, including training scripts, evaluation pipelines, and gradient monitoring utilities, is publicly available at `https://github.com/Anonymous/GradES`. The repository includes detailed documentation, environment setup instructions, and scripts to reproduce all experimental results presented in this paper.

Licensing. All code is released under the MIT License, promoting open scientific collaboration and industrial adoption. Model weights follow the original Qwen license terms.

Table 5: Comparison of the accuracy for different fine-tuning methods on five different language models. Values are reported in percentages, and the best one in each category is highlighted in bold.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	Winogrande	OpenBookQA	ARC-C	ARC-E	Avg.
Qwen3 14B	Full Parameter	91.07	91.29	81.93	95.11	83.03	91.60	94.31	98.07	90.80
	FP+ES	91.07	91.08	81.99	95.05	83.03	91.40	93.98	98.07	90.71
	FP+GradES	**91.22**	91.19	82.04	94.97	83.03	**91.80**	**94.31**	97.89	**90.81**
	LoRA	90.86	**91.24**	81.68	95.21	83.35	91.40	94.31	97.19	90.65
	LoRA+ES	90.64	91.08	81.99	**95.40**	81.53	91.40	93.65	97.54	90.40
	LoRA+GradES	90.67	91.13	**82.29**	95.22	**84.77**	91.20	93.31	97.02	90.70
Phi4 14B	Full Parameter	90.49	91.95	83.27	95.49	88.71	93.00	94.31	98.25	91.93
	FP+ES	90.34	92.11	82.55	95.50	88.56	93.00	94.31	98.07	91.80
	FP+GradES	90.31	**92.60**	**83.06**	95.43	**88.95**	92.60	94.31	98.25	**91.94**
	LoRA	90.31	92.00	82.45	95.36	87.53	91.80	**94.65**	98.07	91.52
	LoRA+ES	90.61	92.22	82.60	**95.44**	87.37	92.00	94.31	**98.25**	91.60
	LoRA+GradES	**90.49**	92.60	82.40	95.38	87.37	91.60	94.65	98.07	91.57
Qwen3 0.6B	Full Parameter	77.28	69.31	66.99	65.09	50.28	61.20	61.54	80.53	66.53
	FP+ES	77.06	68.99	67.09	65.16	49.57	61.00	63.21	81.23	66.66
	FP+GradES	77.03	**71.38**	66.84	64.26	**51.62**	62.40	61.20	79.65	66.80
	LoRA	**79.14**	69.15	**69.14**	67.94	49.09	**66.20**	60.54	77.19	67.30
	LoRA+ES	79.11	69.10	68.83	**68.18**	49.64	65.00	61.20	77.89	**67.37**
	LoRA+GradES	78.56	69.42	68.17	68.20	49.41	65.40	**61.54**	77.72	67.30
Llama-3.1-8B	Full Parameter	89.27	88.08	81.01	94.40	81.93	85.00	83.61	92.46	86.97
	FP+ES	89.24	87.81	80.86	94.42	82.56	85.80	82.61	92.11	86.93
	FP+GradES	88.87	**88.25**	80.45	94.23	82.79	84.80	**83.95**	**92.81**	**87.02**
	LoRA	87.98	87.65	79.73	94.31	80.35	**87.20**	83.28	91.40	86.49
	LoRA+ES	88.62	88.19	79.32	94.20	80.35	87.20	83.28	91.75	86.62
	LoRA+GradES	**88.78**	88.03	**79.68**	**94.43**	**81.69**	87.00	83.95	90.53	86.76
Mistral-7B	Full Parameter	85.26	80.25	77.33	83.71	66.30	75.60	62.54	75.44	75.80
	FP (ES)	85.32	80.09	76.20	83.95	57.54	74.60	64.21	74.39	74.54
	FP+GradES	85.38	78.56	75.79	84.14	66.47	76.40	61.20	75.10	75.38
	LoRA	**89.33**	87.43	79.79	**94.95**	79.72	84.20	76.25	88.42	85.01
	LoRA+ES	88.93	**88.30**	**81.01**	94.69	**82.24**	**85.00**	79.60	90.35	**86.27**
	LoRA+GradES	89.36	88.03	80.71	94.69	80.90	84.40	**81.61**	**89.65**	86.17

Table 6: Training time and computational cost comparison of different fine-tuning methods on 5 different language models. Training time in seconds, FLOPs in floating point operations. Speedup and FLOPs ratios are computed relative to Full Parameter(Base) for all methods. The best one in each category is highlighted in bold.

Model	**Method**	**Training Time (s)**	**Speedup**	**FLOPs**	**FLOPs Ratio**
Qwen3-14B	Full Parameter(Base)	16,202	1.00×	1.17×10^{18}	1.00×
	FP+ES	22,466	0.72×	8.76×10^{17}	0.75×
	FP+GradES	10,721	1.51×	6.43×10^{17}	**0.55**×
	LoRA	6,387	2.54×	2.74×10^{18}	2.34×
	LoRA+ES	23,932	0.68×	2.74×10^{18}	2.34×
	LoRA+GradES	5,643	**2.87**×	2.43×10^{18}	2.08×
Phi4-14B	Full Parameter(Base)	14,627	1.00×	1.12×10^{18}	1.00×
	FP+ES	23,040	0.63×	9.49×10^{17}	0.85×
	FP+GradES	9,218	1.59×	6.15×10^{17}	**0.55**×
	LoRA	6,030	2.43×	2.69×10^{18}	2.40×
	LoRA+ES	24,394	0.60×	2.69×10^{18}	2.40×
	LoRA+GradES	5,506	**2.66**×	2.38×10^{18}	2.13×
Qwen3-0.6B	Full Parameter(Base)	6,550	1.00×	3.68×10^{16}	1.00×
	FP+ES	8,569	0.76×	2.76×10^{16}	0.75×
	FP+GradES	4,018	1.63×	2.03×10^{16}	**0.55**×
	LoRA	892	**7.34**×	8.94×10^{16}	2.43×
	LoRA+ES	6,155	1.06×	8.94×10^{16}	2.43×
	LoRA+GradES	907	7.22×	8.43×10^{16}	2.29×
Llama-3.1-8B	Full Parameter(Base)	9,541	1.00×	7.45×10^{17}	1.00×
	FP+ES	16,129	0.59×	6.70×10^{17}	0.90×
	FP+GradES	5,832	1.64×	4.10×10^{17}	**0.55**×
	LoRA	3,737	2.55×	1.58×10^{18}	2.12×
	LoRA+ES	15,499	0.62×	1.58×10^{18}	2.12×
	LoRA+GradES	3,370	**2.83**×	1.40×10^{18}	1.88×
Mistral-7B	Full Parameter(Base)	9,256	1.00×	6.38×10^{17}	1.00×
	FP+ES	14,521	0.64×	6.38×10^{17}	1.00×
	FP+GradES	6,996	1.32×	4.51×10^{17}	**0.71**×
	LoRA	3,752	2.47×	1.51×10^{18}	2.37×
	LoRA+ES	11,159	0.83×	1.51×10^{18}	2.37×
	LoRA+GradES	3,259	**2.84**×	1.32×10^{18}	2.07×

Table 7: Time and FLOPs comparison of *GradES* on vision-language model for Qwen2.5-VL-7B across different fine-tuning methods.

Model	**Method**	**Training Time (s)**	**Speedup**	**FLOPs**	**FLOPs Ratio**
Qwen2.5-VL-7B	Full Parameter	157,239	1.00×	2.27×10^{19}	1.00×
	FP+GradES	134,686	1.17×	1.99×10^{19}	**0.88**×
	LoRA	103,466	1.52×	2.80×10^{19}	1.23×
	LoRA+GradES	87,572	**1.80**×	2.52×10^{19}	1.11×

Table 8: Hyperparameter configuration for language model fine-tuning experiments. FP denotes full-parameter fine-tuning.

Model	Method	Learning Rate	Batch Size	Max Seq Len	LoRA Rank
Qwen3 14B	FP	2e-5	1	4096	-
	LoRA	2e-4	16	4096	32
Phi4 14B	FP	2e-5	1	4096	-
	LoRA	2e-4	16	4096	32
Qwen3 0.6B	FP	2e-5	1	4096	-
	LoRA	2e-4	16	4096	32
Llama-3.1-8B	FP	2e-5	1	4096	-
	LoRA	2e-4	16	4096	32
Mistral-7B	FP	2e-5	1	4096	-
	LoRA	2e-4	16	4096	32

Table 9: *GradES* convergence thresholds for language models. Grace period ratio (α) determines the fraction of training steps before gradient monitoring begins. Threshold τ defines the gradient magnitude below which components are converged. Full-parameter fine-tuning requires higher thresholds due to larger gradient magnitudes.

Model	Method	Grace Period Ratio(α)	Threshold Tau (τ)
Qwen3 14B	FP	0.55	6.387926
	LoRA	0.55	0.025181
Phi4 14B	FP	0.55	3.512882
	LoRA	0.55	0.025181
Qwen3 0.6B	FP	0.55	1.804456
	LoRA	0.55	0.001183
Llama-3.1-8B	FP	0.55	2.404167
	LoRA	0.55	0.021637
Mistral-7B	FP	0.55	2.726866
	LoRA	0.55	0.029591

Table 10: Hyperparameter configuration for vision-language model experiments. Component-specific thresholds enable targeted convergence monitoring for vision and language transformers separately.

Model	Method	Vision τ	Language τ	Grace Period (α)
Qwen2.5-VL-7B	LoRA	3.3	33.0	0.30
	FP (4-bit)	0.13	0.09	0.30
NanoVLM	Training	0.30	6.00	0.28
	Training (alt)	6.30	3.90	0.28

Table 11: Computational efficiency of *GradES* on vision-language models. FLOPs and training time measurements demonstrate consistent improvements across different fine-tuning methods and precision settings.

Model	**Method**	**FLOPs**	**Time (s)**	**Speedup**
Qwen2.5-VL-7B	LoRA (bf16)	2.80E+19	103,466.02	-
	LoRA+GradES (bf16)	2.52E+19	87,572.36	1.18×
	FP (4-bit)	2.27E+19	157,239.03	-
	FP+GradES (4-bit)	1.99E+19	134,686.05	1.17×
NanoVLM	Training	-	62,009.00	-
	Training+GradES	-	34,669.00	1.79×