TERMEZO: TERNARY SPARSE ZEROTH-ORDER OPTIMIZATION FOR FINE-TUNING BITNET MODELS AT THE EDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning large language models (LLMs) with first-order optimizers requires a memory several times larger than that required for inference. Memory-efficient zeroth-order optimization (MeZO) sidesteps this cost by estimating gradients from forward passes only. However, for BitNet architectures, a family of LLMs with ternary $\{-1, 0, 1\}$ weights and 8-bit activations, fine-tuning requires updating full-precision latent weights, and thus the memory footprint of MeZO no longer matches that of inference. A promising solution is to finetune only a subset of the latent weights, but existing sparse zeroth-order (ZO) methods either ignore the ternary structure or require first-order gradient information to build a sparse mask, which is at odds with the purpose of ZO fine-tuning. We propose TerMeZO, a sparse MeZO scheme that exploits the geometry of the ternary quantizer itself to identify the latent weights that are more likely to change values during fine-tuning, at no additional data or memory cost. Our convergence analysis shows that TerMeZO can converge faster than full-parameter MeZO, owing to its optimized reduction of the fine-tuning effective dimension. We run extensive experiments on BitNet models ranging from 1B to 3B parameters, spanning classification, instruction-following, and mathematical reasoning tasks. TerMeZO matches or exceeds the performance of full-parameter MeZO while substantially reducing the fine-tuning memory footprint.

1 INTRODUCTION

Deploying large language models (LLMs) on resource-constrained and edge hardware has become a practical necessity for a growing range of applications, yet it remains difficult, since the memory, compute, and energy footprint of modern LLMs far exceeds what such platforms can afford (Rajbhandari et al., 2020; Dettmers et al., 2022). Extreme quantization has emerged as the most promising route around this barrier. The BitNet family of models (Wang et al., 2023; Ma et al., 2024; 2025) trains LLMs from scratch with ternary $\{-1, 0, 1\}$ weights and 8-bit activations, replacing most matrix multiplications with integer additions and shrinking the memory footprint enough to make LLM inference viable on commodity and edge devices.

Adapting such a model to a new task or user requires fine-tuning, and here the memory advantages of quantization are lost. In fact, fine-tuning with first-order (FO) optimizers must store gradients, optimizer states, and activations, incurring a footprint several times larger than inference (Rajbhandari et al., 2020; Dettmers et al., 2022). Our work is part of the broader effort to relieve this bottleneck and to bring the cost of fine-tuning quantized LLMs back down toward the cost of inference.

Zeroth-order (ZO) optimization (Malladi et al., 2023) offers a promising direction for this goal, as it estimates gradients from forward passes alone through simultaneous perturbation stochastic approximation (SPSA) (Spall, 1992). Being backpropagation (BP)-free, ZO fine-tuning avoids storing gradients and activations, and for standard LLMs, the memory-efficient ZO optimizer (MeZO) (Malladi et al., 2023) fine-tunes at roughly the memory cost of inference. However, this parity does not carry over to BitNet models, which are fine-tuned through full-precision *latent weights* that are quantized on the fly at each forward pass, accumulating updates in full precision. As a result, even MeZO must

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

perturb and update every latent weight at each step, so that its memory footprint no longer matches that of inference for the quantized model.

As an alternative route to memory savings, parameter-efficient fine-tuning (PEFT) (Hu et al., 2022; Houlsby et al., 2019; Lester et al., 2021), reduces the number of trainable parameters, while still relying on BP. Accordingly, PEFT retains much of the activation and gradient overhead that ZO is designed to avoid. Closer to our setting, several sparse ZO optimization methods build on MeZO to lighten fine-tuning further, yet none are designed for heavily quantized models like BitNet. FIM-S-MeZO (Guo et al., 2024) restricts fine-tuning to a subset of parameters selected by FO gradient information, a process that itself requires BP and thus partially negates the memory benefits of ZO. S-MeZO (Liu et al., 2024b) selects a sparse working set by weight magnitude, a criterion that ignores the ternary decision boundaries. QZO (Shang et al., 2026) avoids FO overhead by freezing the integer weights entirely and tuning only the full-precision quantization scales. QZO is effective when the required knowledge is already present in the pre-trained model, but too restrictive on harder tasks that demand acquiring new knowledge.

In this work, we propose TerMeZO, a sparse MeZO scheme tailored to BitNet models that does not require FO gradients. Our key observation is that the ternary quantizer itself provides a natural criterion for identifying the latent weights most likely to benefit from fine-tuning. Weights closest to the quantization boundaries are the most sensitive to perturbations, and therefore the most likely to flip their integer value during ZO fine-tuning. In contrast, weights far from the quantization boundaries are effectively frozen in ternary space. TerMeZO restricts perturbations and updates to this sparse, boundary-adjacent subset of latent weights, while also updating the remaining full-precision parameters in BitNet models (Wang et al., 2023; Ma et al., 2024; 2025), namely the embeddings, normalization layers, and language-modeling (LM) head. Unlike the selection rule in (Guo et al., 2024), which requires full-parameter FO gradients, TerMeZO's selection rule requires only a linear scan of the latent weights. This scan can be performed offline, before the model is uploaded to the edge device, or on-device, provided the device has enough memory to hold the model together with its latent weights.

We first show empirically that this boundary-proximity criterion selects the latent weights carrying the largest gradient signal, confirming that TerMeZO concentrates fine-tuning where it matters most. Then, we provide a convergence analysis proving that TerMeZO can converge faster than full-parameter MeZO by virtue of its reduced effective dimension (Section 4). The design and theory are validated through extensive experiments on BitNet models ranging from 1B to 3B parameters, including Microsoft's BitNet-b1.58-2B-4T (Ma et al., 2025) and the Falcon-Edge family (Falcon-LLM Team, 2025). TerMeZO is seen to match full-parameter MeZO, exceeding it in several cases, while consistently outperforming existing sparse MeZO baselines that do not rely on FO gradient information (Section 6).

The main contributions are as follows:

- We propose TerMeZO, a sparse MeZO scheme tailored to BitNet models that enables ZO fine-tuning without sacrificing the memory benefits of quantization, making on-device adaptation at the edge more practical.
- We provide a convergence analysis establishing that TerMeZO can converge faster than full-parameter MeZO owing to its reduced effective dimension.
- We demonstrate across BitNet models from 1B to 3B parameters and a range of downstream tasks that TerMeZO matches or exceeds full-parameter MeZO at a reduced memory footprint.

2 Background and Related Work

Our work sits at the intersection of three lines of research, namely PEFT, ZO optimization for memory-efficient fine-tuning, and extreme quantization. We review these areas in the following, with additional details provided in Appendix A.

Parameter-Efficient Fine-Tuning. For modern LLMs, full fine-tuning via FO optimizers is increasingly impractical, especially for resource-constrained scenarios, requiring a memory footprint several times larger than that of inference to store optimizer states and activations (Rajbhandari et al., 2020; Dettmers et al., 2022). PEFT methods address this bottleneck by exploiting the observation that fine-tuning updates lie in a low-dimensional subspace of the full-parameter space (Aghajanyan

et al., 2021; Lialin et al., 2023; Han et al., 2024). They freeze the pre-trained weights and introduce a small set of trainable parameters, achieving performance close to full-parameter fine-tuning. PEFT methods can be broadly grouped into four families, namely adapter-based methods (Houlsby et al., 2019; Pfeiffer et al., 2021; He et al., 2022; Mahabadi et al., 2021; Zhang et al., 2022), soft-prompt and prefix methods (Lester et al., 2021; Liu et al., 2022; Li & Liang, 2021; Lester et al., 2021; He et al., 2022), low-rank reparameterization (Hu et al., 2022; Dettmers et al., 2023; Xu et al., 2024b; Zhang et al., 2023; Liu et al., 2024a; Kopiczko et al., 2024), and sparse fine-tuning (Guo et al., 2021; Zaken et al., 2022; Sung et al., 2021; Ansell et al., 2022; Xu et al., 2024a).

Zeroth-Order Optimization and MeZO. ZO optimization estimates gradients from function evaluations alone, without requiring a backward pass. A widely adopted ZO estimator is SPSA (Spall, 1992), which approximates the gradient $\nabla F(\mathbf{w})$ of the objective function $F(\mathbf{w})$ along a random direction $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ as

$$\widehat{\nabla F}(\mathbf{w}) \;=\; \frac{F(\mathbf{w}+\epsilon\mathbf{z}) - F(\mathbf{w}-\epsilon\mathbf{z})}{2\epsilon}\,\mathbf{z}, \tag{1}$$

where ϵ is a small constant, requiring only two function evaluations per step. Other choices of the random direction $\mathbf{z}$ include uniform (Duchi et al., 2015) and Rademacher (Dang et al., 2026). ZO optimization has historically been applied to non-differentiable or black-box objectives, arising in settings such as adversarial attacks (Chen et al., 2017; Ilyas et al., 2018) and reinforcement learning (Salimans et al., 2017), where gradients are inherently unavailable. More recently, *memory-efficient ZO* (MeZO) (Malladi et al., 2023) demonstrated that SPSA can also be used to fine-tune LLMs using only the memory footprint of inference. MeZO's success in fine-tuning hinges on the locally well-conditioned loss landscape of pre-trained LLMs near initialization, which bypasses the dimension dependence of standard ZO convergence bounds (Malladi et al., 2023). A growing body of follow-up work has sought to improve MeZO along several directions. In particular, reduced-variance methods (Zhao et al., 2025; Cai et al., 2026; Dang et al., 2026; Dbouk et al., 2026) has been investigated in to enhance convergence. Other schemes optimize the perturbation using curvature-aware (Seung et al., 2025) or activation-guided (Lin et al., 2026) criteria, and gradient-informed perturbations (Mi et al., 2025). These methods are compatible with TerMeZO and they could be combined with it, although we do not explore this direction in this work. Information about curvature can be also leveraged to dynamically select which weights to update (Wang et al., 2026). However, such approach incur a memory requirement higher than full-parameter MeZO, and we therefore consider them poorly suited to the near-inference-memory of quantized models regime we target in this work.

Fine-Tuning Quantized Models. Quantization replaces full-precision weights and activations with low-bit representations to reduce memory and compute. Post-training quantization (PTQ) methods such as GPTQ (Frantar et al., 2022), AWQ (Lin et al., 2024), and SmoothQuant (Xiao et al., 2023) achieve 4- and 8-bit inference at a minor accuracy gap, while quantization-aware training (QAT) recovers the gap by simulating quantization during training (Nagel et al., 2021). *Extreme* quantization, 1-bit and ternary weights, has been studied for energy-constrained deployment. BinaryConnect (Courbariaux et al., 2015), XNOR-Net (Rastegari et al., 2016), and ternary weight networks (Li et al., 2016) demonstrated that aggressive weight quantization is feasible for vision models. More recently, the BitNet framework (Wang et al., 2023; Ma et al., 2024) has introduced training LLMs from scratch with $\{-1, 0, 1\}$ weights and 8-bit activations, enabling matrix-multiplication-free inference kernels (Ma et al., 2024; 2025). To further increase the availability of BitNet models, recent works have also explored pipelines for distilling existing pretrained models into BitNet versions (Wu et al., 2025). BitNet distillation is orthogonal to our work, as it is typically applied prior to fine-tuning.

The straight-through estimator (STE) (Bengio et al., 2013) is the standard BP surrogate in this regime, treating the non-differentiable quantizer as the identity in the backward pass. However, STE is known to be a biased gradient surrogate (Yin et al., 2019) and to interact poorly with optimizer states stored at low precision (Li et al., 2021). This further motivates the use of ZO for fine-tuning extremely quantized models.

FIM-S-MeZO (Guo et al., 2024) combines ZO fine-tuning with sparsity, fine-tuning a tiny fraction of parameters identified at pre-training, while keeping the remaining weights quantized to 4 bits. The work (Yang et al., 2024) reduces the effective ZO dimension by inserting adapters and adapting the per-step query budget to stabilize convergence. QuZO (Zhou et al., 2025) adopts a stochastic rounding scheme that lowers gradient bias for quantized weights, allowing fine-tuning with low

precision forward passes. However, we found that this approach fails for ternary quantization (See Appendix H). Most directly related to our work, QZO (Shang et al., 2026) fine-tunes quantized models with MeZO by keeping the quantized weights fixed and fine-tuning only the quantization scales as full-precision parameters. We instead target complex tasks that may require fine-tuning of the ternary weights.

Table 1 summarizes the related work for memory-constrained, BitNet-style deployment along five axes. FO PEFT methods (Hu et al., 2022; Dettmers et al., 2023) require storing full-precision adapters, gradients, and activations throughout training; evaluate matrix multiplication in full-precision; and do not benefit from sparse updates. Among ZO methods, sparse MeZO variants (Liu et al., 2024b; Guo et al., 2024) and TerMeZO save memory by restricting the perturbation and update to a sparse full-precision working set. TerMeZO stands out by obtaining this set directly from the ternary quantization, thus avoiding both the FO optimization steps required to construct a mask in (Guo et al., 2024) and the scale-only update constraints of QZO (Shang et al., 2026).

Table 1: Comparison of fine-tuning methods for BitNet-style models (Matmul = matrix multiplication).

Method	Near inference memory of BitNet	Matmul-free (BitNet like)	No FO overhead	Sparsity	Quantization-Aware Mask
PEFT — FO fine-tuning					
LoRA (Hu et al., 2022)	✗	✗	✗	✗	✗
QLoRA (Dettmers et al., 2023)	✗	✗	✗	✗	✗
ZO fine-tuning					
MeZO (Malladi et al., 2023)	✗	✓	✓	✗	✗
MeZO (LoRA) (Malladi et al., 2023)	✗	✗	✓	✗	✗
S-MeZO (Liu et al., 2024b)	✓	✓	✓	✓	✗
FIM-S-MeZO (Guo et al., 2024)	✗	✓	✗	✓	✗
LoZO (Chen et al., 2025)	✗	✗	✓	✗	✗
ZO fine-tuning — quantized models					
QZO (Shang et al., 2026)	✓	✓	✓	✗	✗
QuZO (Zhou et al., 2025)	✓	✓	✓	✗	✗
TerMeZO (ours)	✓	✓	✓	✓	✓

3 TERMEZO

In this section, we propose a sparse version of MeZO tailored to BitNet models, a family of LLMs with ternary weights (Ma et al., 2024; 2025). BitNet models apply ternary quantization to most linear projections of the transformer blocks, while keeping the token embeddings, normalization modules, and the language modeling (LM) head in full precision (Ma et al., 2025; Wang et al., 2023). Accordingly, ternarization is applied to every linear projection in the transformer blocks, i.e., the query, key, value, and output projections of the multi-head attention modules, as well as to all linear layers in the multi-layer perceptron (MLP) blocks. In the sequel, we refer to these layers as *ternary linear projections*. The proposed TerMeZO preserves the ternary arithmetic of BitNet in the ternary linear projections, perturbing a subset of the latent weights, which are quantized on-the-fly during the forward pass.

Ternary Quantization. Let $\mathbf{w} \in \mathbb{R}^d$ denote the vector collecting all latent weights of the ternary linear projections, and $\boldsymbol{\theta} \in \mathbb{R}^p$ the vector collecting all full-precision parameters of the remaining blocks. The ternary weights dominate the parameter count, i.e., we have $p \ll d$. For example, for the Bitnet model Falcon-E-3B-Instruct (Falcon-LLM Team, 2025), we have the ratio $p/d = 0.044$. During full-parameter fine-tuning, each ternary linear projection maintains full-precision *latent weights* that are quantized on-the-fly in the forward pass. The latent weights $\mathbf{w}$ act as an accumulator for gradient updates. To elaborate, denote as $\mathbf{w}^\ell \in \mathbb{R}^{n^\ell}$ the vectorized latent weights of a ternary linear projection indexed by integer ℓ, where n^ℓ is the number of parameters in that

projection. The ternary weight $\widetilde{\mathbf{w}}^\ell$ is obtained element-wise as

$$[\widetilde{\mathbf{w}}^\ell]_i = Q([\mathbf{w}^\ell]_i) = \begin{cases} 1 & \text{if } [\mathbf{w}^\ell]_i > \tau^\ell \\ -1 & \text{if } [\mathbf{w}^\ell]_i < -\tau^\ell \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

with threshold $\tau^\ell = 0.5\,(\gamma^\ell + \delta)$, where $\gamma^\ell = \sum_{i=1}^{n^\ell} |w_i^\ell|/n^\ell$ is the average absolute value of the weights, and $\delta > 0$ is a small constant for numerical stability (e.g., $\delta = 10^{-5}$). Accordingly, weights far from either threshold are effectively frozen in ternary space when subject to small updates, while weights in the immediate vicinity of the thresholds are the most sensitive to adaptation. Based on this observation, TerMeZO restricts fine-tuning to the subset of latent weights closest to the decision boundaries.

TerMeZO. In TerMeZO, a subset of latent weights $\mathbf{w}^\ell$ and the corresponding quantization threshold τ^ℓ are updated at each fine-tuning iteration t. We denote by $\mathbf{w}_t^\ell$ the current weight vector of ternary linear projection ℓ at iteration t and by τ_t^ℓ the corresponding threshold. For a given latent-weight vector $\mathbf{w}_t^\ell$, the distance of the i-th entry $[\mathbf{w}_t^\ell]_i$ to the quantization boundary is measured by

$$d_{i,t}^\ell(\mathbf{w}_t^\ell) = \min\big(|[\mathbf{w}_t^\ell]_i - \tau_t^\ell|,\ |[\mathbf{w}_t^\ell]_i + \tau_t^\ell|\big), \tag{3}$$

where $[\cdot]_i$ denotes the i-th element of the argument vector. TerMeZO takes as input a target fraction $\rho_0 \in (0, 1]$ of latent weights $[\mathbf{w}^\ell]_i$ to be fine-tuned. To this end, based on the pretrained model, it selects the subsets

$$\mathcal{S}_0^\ell = \Big\{ i : d_{i,0}^\ell\big(\mathbf{w}_0^\ell\big) \le \xi_0 \Big\}, \tag{4}$$

where the threshold ξ_0 is selected so that a fraction ρ_0 of the total number of latent weights, across all layers, is retained. Concretely, this is done by computing the distances of all latent weights (across all projection layers) to their decision boundaries, setting ξ_0 to the ρ_0-quantile of all distances. A validation of this criterion based on a comparison with BP can be found in Appendix B.

The sets $\{\mathcal{S}_0^\ell\}_{\ell=1}^L$ are computed once, offline, and the complexity of this step is linear in the number of latent weights. For the following iterations $t \ge 1$, we allow the active set only to shrink by retaining only previously-active indices whose latent weights still lie within distance ξ_0 of the decision boundaries as

$$\mathcal{S}_t^\ell = \Big\{ i \in \mathcal{S}_{t-1}^\ell : d_{i,t}^\ell\big(\mathbf{w}_t^\ell\big) \le \xi_0 \Big\}. \tag{5}$$

We denote the sparsification mask by $\mathbf{m}_t^\ell \in \{0,1\}^{n^\ell}$, with $[\mathbf{m}_t^\ell]_i = 1 \iff i \in \mathcal{S}_t^\ell$. We further let $\mathbf{m}_t = \{\mathbf{m}_t^\ell\}_{\ell=1}^L$ denote the vector collecting the masks across all layers, and we write the masked latent weights as $\overline{\mathbf{w}}_t = \mathbf{m}_t \odot \mathbf{w}_t$. By construction of set (5), we have the inclusion $\mathcal{S}_t^\ell \subseteq \mathcal{S}_{t-1}^\ell$ via (5), so that the fraction of active latent weights is non-increasing in the iteration index t and upper-bounded by the prescribed budget ρ_0. Once a weight drifts outside the band ξ_0 around its threshold, it is permanently frozen for the remainder of fine-tuning.

The pseudo-code of the TerMeZO algorithm can be found in Appendix C. At each iteration t, TerMeZO updates the sparse sets $\mathcal{S}_t^\ell$ via (4), and performs a standard ZO update on the full-precision parameters $\boldsymbol{\theta}_t$ and the selected latent weights $\overline{\mathbf{w}}_t$. To elaborate, we focus here on supervised fine-tuning (SFT), and refer to Appendix D for the application of TerMeZO for reinforcement learning (RL)-based fine-tuning as a form of evolution strategies (ES) (Qiu et al., 2025).

Let $f(\tilde{\mathbf{w}}, \boldsymbol{\theta}; \mathbf{x}, y)$ represent the sample-wise supervised loss for input-output data point $(\mathbf{x}, y)$, which is parameterized by ternary weights $\tilde{\mathbf{w}}$ and full-precision parameters $\boldsymbol{\theta}$. Accordingly, the population loss for BitNet models can be expressed as a function of the full-precision weights $\mathbf{w}$ and $\boldsymbol{\theta}$ as

$$\tilde{F}(\mathbf{w}, \boldsymbol{\theta}) = F(Q(\mathbf{w}), \boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)}\big[f(Q(\mathbf{w}), \boldsymbol{\theta}; (\mathbf{x}, y))\big], \tag{6}$$

with $Q(\cdot)$ being the quantization function and $\mathbb{E}_{(\mathbf{x},y)}[\cdot]$ representing the expectation over the data distribution. A batch loss estimate using a mini-batch $\mathcal{B}$ of size $|\mathcal{B}|$ is denoted as

$$\widehat{\tilde{F}}(\mathbf{w}, \boldsymbol{\theta}; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{x},y)\in\mathcal{B}} \big[f(Q(\mathbf{w}), \boldsymbol{\theta}; (\mathbf{x}, y))\big]. \tag{7}$$

At each iteration t, K Gaussian perturbations are applied to both the masked latent weights, i.e., the elements i of vector $\overline{\mathbf{w}}_t = \mathbf{m}_t \odot \mathbf{w}_t$ with $[\mathbf{m}_t]_i = 1$, and to the full-precision parameters $\boldsymbol{\theta}_t$.

A two-sided SPSA estimator is used, where the loss is evaluated along each perturbation direction with scale $\epsilon > 0$ on a shared mini-batch $\mathcal{B}_t$, and the resulting finite differences are averaged over the K perturbations to update the model parameters.

4 Convergence Analysis

In this section, we study the convergence of TerMeZO. To facilitate the analysis, we assume that a single perturbation is applied at each iteration, i.e., $K = 1$, but the extension to general K follows by standard averaging arguments (Malladi et al., 2023)). The objective function $\tilde{F}(\mathbf{w}, \boldsymbol{\theta})$ in (6) is non-differentiable with respect to latent weights $\mathbf{w}$, making it difficult to analyze the convergence to a local optimizer. However, SPSA with Gaussian perturbations $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ provides an unbiased estimator of the gradient of the Gaussian smoothed-objective (Nesterov & Spokoiny, 2015)

$$\tilde{F}_\epsilon(\mathbf{w}, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[F(\mathbf{w} + \epsilon\mathbf{z}, \boldsymbol{\theta})], \tag{8}$$

i.e.,

$$\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{\tilde{F}(\mathbf{w} + \epsilon\mathbf{z}, \boldsymbol{\theta}) - \tilde{F}(\mathbf{w} - \epsilon\mathbf{z}, \boldsymbol{\theta})}{2\epsilon}\,\mathbf{z}\right] \;=\; \nabla_{\mathbf{w}}\tilde{F}_\epsilon(\mathbf{w}, \boldsymbol{\theta}). \tag{9}$$

For completeness, a proof of this result can be found in Appendix E. As in (Ghadimi & Lan, 2013; Liu et al., 2018; Malladi et al., 2023), we study convergence to a stationary point with respect to the smoothed objective (8) through the time-averaged expected squared gradient norm

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla\tilde{F}_\epsilon(\mathbf{w}_t, \boldsymbol{\theta}_t)\|^2 = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla_{\mathbf{w}}\tilde{F}_\epsilon(\mathbf{w}_t, \boldsymbol{\theta}_t)\|^2 + \|\nabla_{\boldsymbol{\theta}}\tilde{F}_\epsilon(\mathbf{w}_t, \boldsymbol{\theta}_t)\|^2\big]. \tag{10}$$

To this end, we make the following four assumptions, which are formally stated in Appendix F.1: *Assumption 1)* The objective function $\tilde{F}(\mathbf{w}, \boldsymbol{\theta})$ is uniformly bounded; *Assumption 2)* The loss function $F(\mathbf{w}, \boldsymbol{\theta})$ is L-Lipschitz in $\mathbf{w}$ and has L-Lipschitz gradients in $\boldsymbol{\theta}$; *Assumption 3)* The masked gradient estimator captures a c-fraction of the gradient energy of the smoothed objective $\tilde{F}_\epsilon(\mathbf{w}, \boldsymbol{\theta})$ along the latent weights $\mathbf{w}$; and *Assumption 4)* the minibatch gradients have variance at most σ_B^2. Furthermore, we define as k_t the number of active coordinates during iteration t, and introduce the *average boundary-crossing fraction* q_t to measure the average fraction of active weights that change value after a perturbation:

$$k_t = \sum_{\ell=1}^{L}\sum_{i=1}^{n^\ell}[\mathbf{m}_t^\ell]_i, \qquad q_t = \frac{1}{k_t}\sum_{\ell=1}^{L}\sum_{i:\,[\mathbf{m}_t^\ell]_i=1} P\big[Q([\mathbf{w}_t^\ell + \epsilon\mathbf{z}^\ell]_i) \neq Q([\mathbf{w}_t^\ell]_i)\big]. \tag{11}$$

We donote by $N_t = \sum_{\ell=1}^{L}\sum_{i=1}^{n^\ell}\mathbb{E}\big[\big(Q([\mathbf{w}_t^\ell + \epsilon\mathbf{z}^\ell]_i) - Q([\mathbf{w}_t^\ell]_i)\big)^2\big]$ the expected number of latent weights that change value after a perturbation.

Theorem 1. *Let Assumptions 1–4 hold. Define* $L_{\mathbf{w}} = \sqrt{2k_0}\,L/\epsilon^2$ *and* $L_{\max} = \max\big\{L_{\mathbf{w}},\ 8c^2L(p+2)\big\}$. *Consider TerMeZO with constant step sizes* $\eta = c/(2L_{\mathbf{w}})$ *for the latent weights* $\mathbf{w}$ *and* $\mu = c^2/L_{\max}$ *for the full-precision parameters* $\boldsymbol{\theta}$. *Then, after* T *steps of TerMeZO algorithm, the time-averaged gradient norm* (10) *satisfies the inequality*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla\tilde{F}_\epsilon(\mathbf{w}_t, \boldsymbol{\theta}_t)\|^2 \;\leq \frac{2L_{\max}\Delta_0}{c^2T} + \mathcal{O}\Big(\epsilon^2p^2 + \overline{N} + \sigma_B^2 + L_{\max}\,\overline{\sigma_{\mathbf{w}}^2}\Big), \tag{12}$$

where $\overline{N} = \frac{1}{T}\sum_{t=0}^{T-1}N_t$; $\overline{\sigma_{\mathbf{w}}^2} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\sigma_{\mathbf{w},t}^2]$, *with* $\sigma_{\mathbf{w},t}^2 = k_t^2q_t + \sqrt{3}\,k_t\sqrt{q_t}$; $\Delta_0 = \tilde{F}_\epsilon(\mathbf{w}_0, \boldsymbol{\theta}_0) - \tilde{F}_\epsilon^*$ *is the initial optimality gap;* $c \in (0, 1]$ *is the mask-overlap constant of Assumption 3; and* σ_B^2 *is the minibatch variance of Assumption 4.*

We make the following remarks about connections of Theorem 1 with prior art:

- When no quantization is applied, only the terms corresponding to the full-precision parameters $\boldsymbol{\theta}$ remain in the bound (12), and thus standard MeZO convergence rate $O(p/T)$ for p trainable parameters is recovered (Malladi et al., 2023; Liu et al., 2018).

- When the initial mask size k_0 dominates the number of trainable parameters, i.e., when $k_0 \gg p$, we have $L_{\max} = L_{\mathbf{w}} = \sqrt{2k_0}\, L/\epsilon^2$, and the bound (12) gives an overall convergence rate $O\big(\sqrt{k_0}/(\epsilon^2 T)\big)$ to a neighborhood of a stationary point, which is governed by the number k_0 of active latent weights.
- A convergence rate of $O(k/T)$ was proved for sparse MeZO (Guo et al., 2024), where k is the mask size. Setting $\epsilon^2 = 1/\sqrt{k_0}$ in (12) recovers this rate. Note that this scale gives the range $\epsilon \in [3 \cdot 10^{-3}, 10^{-2}]$ for active-parameter counts $k_0 \in [10^8,\, 10^{10}]$, matching the typical order of magnitude of the perturbation scale ϵ used in practice (Malladi et al., 2023).

5 Memory-Footprint Analysis

In this section, we elaborate on the memory savings afforded by TerMeZO over BP and full-parameter MeZO. We start by relating the memory footprints of full-parameter MeZO and TerMeZO for the same Bitnet model, and we then provide an experimental evaluation to support the analysis. The key difference between TerMeZO and full-parameter MeZO is that the former keeps a full-precision copy only of the active latent weights, while also additionally storing the sparsification mask. Let B be the number of bytes per full-precision weight, e.g., $B = 2$ for bfloat16, and let B_Q be the effective number of bytes per packed ternary weight, e.g., $B_Q = 2/8$ for BitNet models (Wang et al., 2023; Ma et al., 2025). Downgrading a frozen latent weight from full precision to ternary saves $B - B_Q$ bytes. A fraction $1 - \rho_0$ of the weights are frozen in TerMeZO, while each of the $\rho_0 d$ active weights additionally retains its B_Q-byte quantized copy. Hence, the peak TerMeZO memory footprint M_{TerMeZO} can be expressed as a function of the memory footpring of MeZO M_{MeZO} as

$$M_{\text{TerMeZO}} \;=\; M_{\text{MeZO}} \;-\; \lceil d\,(1-\rho_0)\,(B - B_Q)\rceil \;+\; \lceil \rho_0 d\, B_Q\rceil \;+\; M_{\text{mask}}. \tag{13}$$

The mask size in (13) is given by $M_{\text{mask}} = \min\big(\lceil d/8\rceil,\ \lceil \rho_0 d \lceil \log_2 d\rceil/8\rceil\big)$ bytes, where the first term corresponds to the option of a dense bit-mask storing one bit per latent weight, and the second to an active-index list that records the $\lceil \log_2 d\rceil$-bit index of each active weight.

Table 2 reports the measured memory footprint on GPU of BP, full-parameter MeZO, and TerMeZO, along with that of pure inference as a lower bound, for fine-tuning of Falcon-E-1B and Falcon-E-3B on an H100 GPU. Measuring the footprint of TerMeZO and of inference each requires a custom kernel. A BitNet inference kernel for GPUs is already available (Ma et al., 2025), and we implement an analogous kernel for TerMeZO. The table also shows the estimate (13) for comparison. The results in Table 2 confirm the significant memory savings of MeZO relative to BP (Katti et al., 2025). TerMeZO improves further on MeZO by a wide margin. In particular, at sparsity level $\rho_0 = 0.05$, it reduces the footprint by roughly $3\times$ relative to MeZO, and by around $4\times$ at level $\rho_0 = 0.01$, bringing the memory cost of fine-tuning close to that of inference.

Table 2: Memory footprint (GB) during fine-tuning of Falcon-E-1B and Falcon-E-3B on an H100 GPU. For TerMeZO, we report both measured and estimated values using (13). The TerMeZO and inference numbers assume 2-bit storage of the ternary weights. Results here are measured usi SFT with the context length fixed to 512.

	BP	**MeZO**	**TerMeZO** ($\rho_0 = 0.05$)	**TerMeZO** ($\rho_0 = 0.01$)	**Inference**
Falcon-E-1B	16.08	3.61	**1.33** (est. 1.31)	**0.94** (est. 0.93)	0.89
Falcon-E-3B	28.50	6.28	**2.08** (est. 2.04)	**1.36** (est. 1.34)	1.24

6 Empirical Results

We conduct experiments on a set of publicly available BitNet models across a range of tasks. Specifically, we evaluate on Microsoft's BitNet-b1.58-2B-4T (Ma et al., 2025), a 2B-parameter model pre-trained on 4T tokens, and the Falcon-Edge family (Falcon-LLM Team, 2025), a set of BitNet models ranging from 1B to 3B parameters designed for on-device deployment. Through these experiments, we aim to answer the following questions: 1) Does TerMeZO match the performance of

full-parameter MeZO on BitNet models? 2) How does TerMeZO compare to existing sparse and PEFT variants of MeZO on BitNet models?

GLUE/SuperGLUE Classification Tasks. Here, we benchmark TerMeZO against the following baselines: (i) standard BP fine-tuning with Adam optimizer; (ii) in-context learning (ICL) (Brown et al., 2020); (iii) full-parameter MeZO (Malladi et al., 2023), which applies ZO updates to all latent weights and full-precision parameters; (iv) MeZO with LoRA (LoRA-MeZO); (v) FIM-S-MeZO (Guo et al., 2024); and (vi) QZO (Shang et al., 2026), a MeZO variant tailored to quantized models that perturbs the quantization scales rather than the latent weights themselves. Following the conventional MeZO evaluation protocol (Malladi et al., 2023), fine-tuning is cast as language modeling over a task-specific prompt template. We report results on six standard classification benchmarks from the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2021) suites; namely SST2 (Socher et al., 2013), RTE (Dagan et al., 2005), CoPA (Roemmele et al., 2011), CB (CommitmentBank) (De Marneffe et al., 2019), BoolQ (Clark et al., 2019), and MultiRC (Khashabi et al., 2018). For all tasks, we report the accuracy on the validation set in Table 3 after fine-tuning using the BitNet-b1.58-2B-4T model. All ZO methods are run for the same number of optimization steps to ensure a fair comparison. Hyperparameters and experiment details can be found in Appendix G.1.

We observe that TerMeZO is as effective as full-parameter MeZO fine-tuning on most of these tasks, and even exceeds it on some tasks. QZO, which fine-tunes only the scales of the ternary projections, is also observed to perform well. This is expected as these tasks are relatively simple, implying that the model already possesses the required knowledge and tuning only a subset of the full-precision parameters is sufficient. In contrast, LoRA-MeZO lags behind the other baselines. We attribute this to the known limitation of ZO optimization as a method for training from scratch (Malladi et al., 2023). In fact, unlike full fine-tuning, which starts from a well-conditioned pretrained initialization, LoRA introduces randomly initialized low-rank matrices. This effect is compounded in our setting, since, in order to remain compatible with the BitNet architecture, we constrain the LoRA projections to be ternary.

Table 3: Accuracy (%) of TerMeZO and baselines on classification tasks using the Microsoft/bitnet-b1.58-2B-4T BitNet model (Ma et al., 2025). The fraction of trainable latent weights is fixed to $\rho_0 = 0.05$ for TerMeZO.

	SST2	RTE	CoPA	CB	BoolQ	MultiRC
BP	94.8	84.1	83	88.0	86.1	83.4
ICL	85.4	75.4	80	78.5	76.8	73.1
MeZO	93.2	81.9	78	87.9	82.3	82.2
LoRA-MeZO	55.0	71.6	73	55.5	54.5	66.6
FIM-S-MeZO	92.3	82.4	75	83.8	79.5	81.6
QZO	91.7	79.7	77	82.8	82.9	81.8
TerMeZO (Ours)	93.1	83.4	80	84.8	82.7	83.4

Instruction Following and Reasoning Tasks. We now consider more challenging fine-tuning tasks that potentially require the acquisition of new knowledge, and for which optimizing the ternary weights is required. Specifically, we adopt an instruction-following dataset, MagiCoder (Wei et al., 2023), which measures the ability of the fine-tuned model to follow natural language instructions and generate correct code, as well as a mathematical reasoning task, GSM8K (Cobbe et al., 2021). We include here the S-MeZO baseline with two selection rules, namely maximum weight magnitude and minimum weight magnitude (Liu et al., 2024b).

Table 4 reports results on GSM8K and MagiCoder for two BitNet models, Falcon-E-1B-Base and Falcon-E-3B-Base. For GSM8K, we report the test accuracy, whereas for the models fine-tuned on MagiCoder, we report pass@1 on HumanEval (Chen et al., 2021), the percentage of problems for which the single sampled code completion passes all unit tests. TerMeZO substantially improves over the base model and closely tracks full-parameter MeZO. QZO, by contrast, fails to match TerMeZO, indicating that for these tasks, fine-tuning the quantization scales alone is insufficient. We also observe a consistent performance gap between BP and TerMeZO for a given model size. However, TerMeZO's lower memory footprint makes it possible to fine-tune larger models than BP would allow under the same memory budget. For example, the 3B model fine-tuned with TerMeZO outperforms the 1B model fine-tuned with BP, despite the latter requiring considerably more memory on a smaller model size as shown in Table 2. S-MeZO, whose mask selection relies on latent

Table 4: Performance of TerMeZO with $\rho_0 = 0.05$ and benchmarks on GSM8K and MagiCoder using two BitNet models, Falcon-E-1B-Base and Falcon-E-3B-Base. We report accuracy (%) on GSM8K and pass@1 (%) on MagiCoder.

	Falcon-E-1B-Base		**Falcon-E-3B-Base**	
	GSM8K	MagiCoder	GSM8K	MagiCoder
Base Model	14.0	17.1	20.0	37.8
BP	37.7	32.9	57.4	49.4
MeZO	32.6	25.6	45.3	42.7
FIM-S-MeZO	32.5	25.0	44.5	43.3
S-MeZO (min)	27.8	15.8	26.8	41.5
S-MeZO (max)	22.1	14.6	27.4	42.1
QZO	18.6	16.4	23.3	42.0
TerMeZO (Ours)	31.8	24.4	44.4	43.3

weight magnitude and ignores the quantizer geometry, falls behind by a significant margin. In contrast, while FIM-S-MeZO matches the performance of TerMeZO, it requires initial SGD steps on a separate dataset prior to fine-tuning, resulting in a memory footprint comparable to BP. This performance parity with FIM-S-MeZO further validates the effectiveness of TerMeZO's selection criteria.

Although FIM-S-MeZO performs similarly to TerMeZO in these experimnts, it does so under stronger requirements (e.g., access to FO gradient information). We also observe that TerMeZO is more robust to the choice of learning rate. Specifically, for the same fraction of latent weights, some learning rates (e.g., 5×10^{-4}) cause FIM-S-MeZO to diverge, whereas TerMeZO remains stable (see Appendix H.1). We attribute this to the fact that a large proportion of the weights selected by FIM-S-MeZO are not reachable by perturbations, since they lie far from the quantization boundaries, which yields pure noise in the updates. Finally, we note that when an FO warmup before fine-tuning is affordable (e.g., offline in the cloud), combining the selection criteria of TerMeZO and FIM-S-MeZO could capture the benefits of both. We leave this investigation to future work.

7 Conclusion

Summary. This paper proposed TerMeZO, a sparse MeZO scheme for LLM fine-tuning tailored to BitNet models. The sparse mask selection exploits the ternary weight geometry and is inexpensive to evaluate for this class of models, requiring only a pass through all latent weights before fine-tuning. A convergence analysis of the proposed scheme is provided. Extensive experiments on a variety of datasets confirm the effectiveness of TerMeZO, showing that it recovers, and in some cases exceeds, the performance of full-parameter MeZO, while enabling a substantial reduction in the memory required for fine-tuning. By reducing the fraction of latent weights needed during fine-tuning, fine-tuning BitNet models can be carried out on edge devices with a near-inference memory footprint.

Limitations. An efficient implementation of TerMeZO requires custom, hardware-dependent kernels to realize the memory savings in practice. Future work could target such kernels with a focus on both memory and latency. Additionally, due to the limited availability of larger open-source BitNet models, we have only evaluated our scheme on models ranging from 1B to 3B parameters. Validating the approach at larger scales is left for future work.

Perspectives. A natural extension of this work is to evaluate TerMeZO on BitNet-equipped linear-attention or hybrid-attention models, which would allow us to jointly address the memory footprint of the KV cache alongside that of the latent weights. This is particularly relevant for RL-based fine-tuning where long generation rollouts can make KV cache storage a significant bottleneck. Another important direction is to validate the memory savings reported in this paper on real edge hardware with custom kernels, beyond the GPU-based setup used in our experiments.

AI USE STATEMENT

Generative AI tools were used solely to provide feedback on research methodology and experimental design, refine text for readability, and assist with code editing. All core research activities including theoretical modeling, proof writing, hypothesis formulation, and result interpretation were conducted entirely by the authors without AI assistance.

Text-level AI usage was strictly limited to polishing author-written drafts, and all suggested edits were evaluated before adoption. All AI-assisted code was reviewed. The authors take full responsibility for the final manuscript, including all text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

We provide a detailed description of our experimental setup, including the training and evaluation protocols and the complete set of hyperparameters, in Appendix G. The code is available at `https://anonymous.4open.science/r/termezo-EA51/`.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328, 2021.

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *ACL*, 2022.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zhijie Cai, Haolong Chen, and Guangxu Zhu. Adamezo: Adam-style zeroth-order optimizer for llm fine-tuning without maintaining the moments. *arXiv preprint arXiv:2605.00650*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. *ICML*, 2025.

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

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

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, pp. 177–190, 2005.

Sizhe Dang, Yangyang Guo, Yanjun Zhao, Xiaodong Zheng, Guang Dai, Ivor Tsang, and Haishan Ye. FZOO: Fast zeroth-order optimizer for fine-tuning large language models towards adam-scale speed. In *The Fourteenth International Conference on Learning Representations*, 2026.

Hassan Dbouk, Nidham Gazagnadou, Matthias Reisser, and Christos Louizos. On adaptivity in zeroth-order optimization. *arXiv preprint arXiv:2605.03869*, 2026.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, pp. 107–124, 2019.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *International Conference on Learning Representations (ICLR)*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Falcon-LLM Team. Falcon-e, a series of powerful, universal and fine-tunable 1.58bit language models., May 2025. URL https://falcon-lm.github.io/blog/falcon-edge.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *ACL*, 2021.

Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R. Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. Zeroth-order fine-tuning of LLMs with extreme sparsity. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024. URL https://openreview.net/forum?id=pW4MmsnVRq.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=lIsCS8b6zj.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, 2022.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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 (ICLR)*, 2022.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International conference on machine learning*, pp. 2137–2146. PMLR, 2018.

Prabodh Katti, Houssem Sifaou, Sangwoo Park, Bipin Rajendran, and Osvaldo Simeone. On-device fine-tuning via backprop-free zeroth-order optimization. *arXiv preprint arXiv:2511.11362*, 2025.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *ICLR*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3045–3059, 2021.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 4582–4597, 2021. URL https://api.semanticscholar.org/CorpusID:230433941.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *ICLR*, 2021.

Vladislav Lialin, Vijeta Deshpande, Xiaowei Yao, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *MLSys*, 2024.

Wei Lin, Yining Jiang, Qingyu Song, Qiao Xiang, and Hong Xu. AGZO: Activation-guided zeroth-order optimization for LLM fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *ICML*, 2024a.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. In *ACL*, 2022.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning. *arXiv preprint arXiv:2402.15751*, 2024b.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Shuming Ma, Hongyu Wang, Shaohan Huang, Xingxing Zhang, Ying Hu, Ting Song, Yan Xia, and Furu Wei. BitNet b1.58 2B4T technical report. *arXiv preprint arXiv:2504.12285*, 2025.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*, 2021.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Zhendong Mi, Qitao Tan, Grace Li Zhang, Zhaozhuo Xu, Geng Yuan, and Shaoyi Huang. Towards fast LLM fine-tuning through zeroth-order optimization with projected gradient-aligned perturbations. *arXiv preprint arXiv:2510.18228*, 2025.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Yurii Nesterov and Vladimir G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527 – 566, 2015. URL https://api.semanticscholar.org/CorpusID:2147817.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*, 2021.

Xin Qiu, Yulu Gan, Conor F Hayes, Qiyao Liang, Yinggan Xu, Roberto Dailey, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: LLM fine-tuning beyond reinforcement learning. *arXiv preprint arXiv:2509.24372*, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2020.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pp. 90–95, 2011.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Hyunseok Seung, Jaewoo Lee, and Hyunsuk Ko. Low-rank curvature for zeroth-order optimization in llm fine-tuning. In *AAAI Conference on Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:282922251.

Sifeng Shang, Jiayi Zhou, Chenyu Lin, Minxian Li, and Kaiyang Zhou. Fine-tuning quantized neural networks with zeroth-order optimization. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013.

James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In *NeurIPS*, 2021.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial GLUE: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*, 2021.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

Shuo Wang, Ziyu Chen, and Ming Tang. CurvZO: Adaptive curvature-guided sparse zeroth-order optimization for efficient LLM fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-Instruct. *arXiv preprint arXiv:2312.02120*, 2023.

Xun Wu, Shaohan Huang, Wenhui Wang, Ting Song, Li Dong, Yan Xia, and Furu Wei. Bitnet distillation. *arXiv preprint arXiv:2510.13998*, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023.

Mingyu Xu, Zheng Xu, Subhabrata Choudhury, et al. Random masking finds winning tickets for parameter efficient fine-tuning. *ICML*, 2024a.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. QA-LoRA: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 977–995, 2024.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *ICLR*, 2019.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1–9, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.

Yaqing Zhang, Xiaodong Wang, Junjie Liu, et al. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *EMNLP*, 2022.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for LLMs: A hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*, volume 2025, pp. 43496–43520, 2025.

Jiajun Zhou, Yifan Yang, Kai Zhen, Ziyue Liu, Yequan Zhao, Ershad Banijamali, Athanasios Mouchtaris, Ngai Wong, and Zheng Zhang. QuZO: Quantized zeroth-order fine-tuning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5341–5359, 2025.