Variance-Reduced Zeroth-Order Methods for Fine-Tuning Language Models

Tanmay Gautam[*1] **Youngsuk Park**[†2] **Hao Zhou**[3] **Parameswaran Raman**[2] **Wooseok Ha**[3]

Abstract

Fine-tuning language models (LMs) has demonstrated success in a wide array of downstream tasks. However, as LMs are scaled up, the memory requirements for backpropagation become prohibitively high. Zeroth-order (ZO) optimization methods can leverage memory-efficient forward passes to estimate gradients. Recently, MeZO, an adaptation of ZO-SGD, has been shown to consistently outperform zero-shot and in-context learning when combined with suitable task prompts. In this work, we couple ZO methods with variance reduction techniques to enhance stability and convergence for inference-based LM fine-tuning. We introduce Memory-Efficient Zeroth-Order Stochastic Variance-Reduced Gradient (MeZO-SVRG) and demonstrate its efficacy across multiple LM fine-tuning tasks, eliminating the reliance on task-specific prompts. Evaluated across a range of both masked and autoregressive LMs (up to 7B parameters) on benchmark downstream tasks, MeZO-SVRG outperforms MeZO with up to 20% increase in test accuracies in both full- and partial-parameter fine-tuning settings. MeZO-SVRG benefits from reduced computation time, often surpassing MeZO's peak test accuracy with a $2\times$ reduction in GPU-hours. MeZO-SVRG substantially decreases the memory requirement (by at least $2\times$ for autoregressive models), achieving greater memory savings as both the batch size and context lengths increase, in comparison to first-order methods.

1. Introduction

In recent years, language models (LMs) have exhibited exceptional performance in a vast array of domains within natural language processing (NLP) (Solaiman et al., 2019; OpenAI, 2023; Touvron et al., 2023).This development has generated immense excitement within the research community and has propelled the advancement of the aforementioned models to the forefront of deep learning research.

Fine-tuning LMs has been the dominant strategy for adapting pre-trained models to specialized downstream tasks (Gururangan et al., 2020). Fine-tuning often relies on first-order methods, such as stochastic gradient descent (SGD) (Robbins & Monro, 1951) or Adam (Kingma & Ba, 2015). However, as LMs are scaled up, backpropagation (Rumelhart et al., 1986) becomes prohibitive in terms of memory requirements. More concretely, Malladi et al. (2023) show that fine-tuning an OPT-13B model with full-parameter or parameter efficient fine-tuning (PEFT) using Adam requires $12\times$ and $6\times$ more memory than inference, respectively. This is due to the need to cache activations during the forward pass as well as gradients and optimizer states during the backward pass. This has given rise to memory-efficient inference-based adaptation methods, including in-context learning (ICL) and zeroth-order (ZO) optimization.

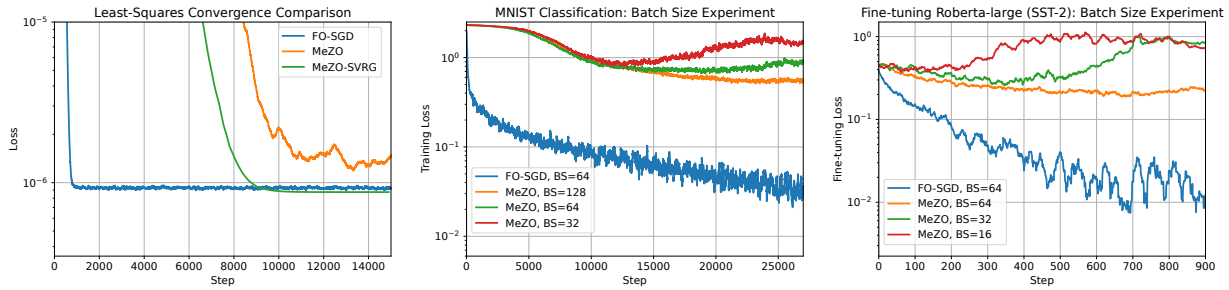
While ZO methods have been studied for decades (Spall, 1992; Ghadimi & Lan, 2013), it is only recently that these have been applied to fine-tune LMs (Malladi et al., 2023). In Malladi et al. (2023), authors propose the Memory-Efficient Zeroth-Order Optimizer (MeZO) and demonstrate its superior performance against ICL with a memory footprint equivalent to that of inference. By virtue of estimating gradients through loss computations, ZO methods are compatible with settings where gradients are non-accessible or infeasible to compute, e.g. when considering non-differentiable objectives or black-box access of LMs.

However, ZO methods still face challenges in large-scale settings. According to Malladi et al. (2023), MeZO requires a high number of iterations to achieve a good fine-tuning performance and works only in settings where the optimization trajectory is sufficiently well-behaved, i.e. when fine-tuning is coupled with appropriately crafted task prompts. As such, we revisit ZO optimization under the standard (non-prompted) fine-tuning setting. Through empirical stud-

[1]University of California, Berkeley, USA [2]Amazon AI Research & Education, Santa Clara, USA [3]Amazon AI Labs, Santa Clara, USA. Correspondence to: Youngsuk Park[†] <pyoungsu@amazon.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[*]Work conducted during an internship at Amazon.



(a) (b) (c)

Figure 1. (a) Shows that MeZO (Malladi et al., 2023) is unable to attain the optimal value when solving least-squares (LS) problems unlike our proposed MeZO-SVRG. In (b) and (c), MeZO is used for MNIST (LeCun et al., 1998) classification and fine-tuning RoBERTa-large on SST-2 (Socher et al., 2013), respectively, with varying batch sizes. These illustrate MeZO's instability w.r.t. smaller batch sizes.

ies, we probed further and identified that the method also contends with i) instability for smaller batch sizes, and ii) a notable convergence gap to first-order (FO) fine-tuning methods in non-prompted settings (see Figures 1a, 1b, 1c).

In this work, we demonstrate that variance-reduction enhances the stability and convergence properties of ZO methods in the large-scale LM fine-tuning setting. Based on our observation that ZO methods benefit from improved stability with larger batch sizes, we propose the Memory Efficient Zeroth-Order Stochastic Variance-Reduced Gradient (MeZO-SVRG) method: a ZO algorithm that combines fullbatch and minibatch information to yield asymptotically unbiased, low-variance gradient estimators. Our specific contributions are enumerated below.

1. We perform empirical studies across a range of problem scales to investigate the potential limitations of MeZO. We identified its susceptibility to unstable behavior for smaller batch sizes and convergence issues in spurious optimization landscapes as improvement avenues.

2. We propose MeZO-SVRG: an efficient variant of the ZO-SVRG method that leverages gradient estimators computed with single perturbation vectors to exploit data parallelism for speed and uses in-place operations to achieve a minimal memory footprint.

3. We fine-tune masked and autoregressive LMs (model scales up to 7B) on GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) tasks. MeZO-SVRG achieves consistent performance improvements with up to 20% increase in test accuracies over MeZO across all models and tasks. MeZO-SVRG achieves superior performance to MeZO in both full- and partial-parameter fine-tuning, in both full (FP32) and half (BF16) precision and under standard non-prompt settings.

4. MeZO-SVRG stands out by consistently surpassing MeZO's test accuracy in only half as many GPU-hours.

5. We show that MeZO-SVRG significantly reduces the required memory footprint compared to first-order methods, i.e. by at least $2\times$ for considered autoregressive models. Furthermore, our experiments highlight that MeZO-SVRG's memory savings progressively improve compared to SGD with larger batch sizes.

6. We establish convergence guarantees for MeZO-SVRG when equipped with gradient estimators that are computed using single perturbation vectors.

2. Background

2.1. Zeroth-Order Gradient Estimators

Consider solving the unconstrained optimization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{\theta}), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a non-convex objective. Note that (1) is akin to the standard empirical risk minimization framework, where each f_i is the objective evaluated for one of n training samples. For an iterative ZO algorithm, we need to find a means to approximate the gradient. We can define the following stochastic perturbation simultaneous approximation (SPSA) gradient estimator (Spall, 1992):

$$\hat{\nabla} f_i(\boldsymbol{\theta}) := \frac{f_i(\boldsymbol{\theta}+\mu\mathbf{z}_i) - f_i(\boldsymbol{\theta}-\mu\mathbf{z}_i)}{2\mu}\mathbf{z}_i \text{ for } i \in [n], \tag{2}$$

where $\hat{\nabla}$ denotes a gradient estimator, $\mathbf{z}_i \in \mathbb{R}^d$ is a random vector sampled from a standard normal distribution, and $\mu > 0$ is a perturbation scalar. The extension p-SPSA computes the average of p distinct SPSA estimates. Throughout this work, we consider the default setting of $p = 1$ as we didn't observe empirical benefits of setting $p > 1$. The SPSA gradient estimate is an asymptotically unbiased estimator of the true gradient as $\mu \to 0$ when each component in $\mathbf{z}_i$ is mutually independent and zero-mean (Spall, 1992).

Now suppose we have a minibatch $\mathcal{I} \subset [n]$ of size b. This allows us to define the following:

$$\hat{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) := \frac{1}{b} \sum_{i \in \mathcal{I}} \hat{\nabla} f_i(\boldsymbol{\theta}), \tag{3}$$

and by extension,

$$\hat{\nabla} f(\boldsymbol{\theta}) := \hat{\nabla} f_{[n]}(\boldsymbol{\theta}). \tag{4}$$

Observe that the gradient estimator in (3) requires $2b$ function queries and sampling b random vectors. In practice, there are two strategies to compute estimators (3) and (4): accumulate the minibatch estimator in-place by sequentially computing each samplewise estimator, or parallelize the operation by computing the samplewise estimators simultaneously. The trade-off between the two strategies is that the former has a minimal memory footprint (scales with dimension of problem) but takes longer, while the latter effectively parallelizes the operation but has to store b vectors.

Thus, we define another set of ZO gradient estimators that accommodate data parallelism: we perturb each samplewise SPSA estimator in the same direction $\mathbf{z} \in \mathbb{R}^d$. For minibatch $\mathcal{I} \subset [n]$ of size b we can construct

$$\bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) := \frac{\frac{1}{b} \sum_{i \in \mathcal{I}} [f_i(\boldsymbol{\theta} + \mu \mathbf{z}) - f_i(\boldsymbol{\theta} - \mu \mathbf{z})]}{2\mu} \mathbf{z}, \tag{5}$$

and

$$\bar{\nabla} f(\boldsymbol{\theta}) := \bar{\nabla} f_{[n]}(\boldsymbol{\theta}). \tag{6}$$

From an implementation standpoint, estimators (5) and (6) can exploit data parallelism across the batch $\mathcal{I}$ and benefit from a minimal required memory footprint.

2.2. Memory-efficient ZO-SGD (MeZO)

In Malladi et al. (2023), the authors propose a memory-efficient ZO-SGD optimizer (MeZO) to fine-tune LMs. MeZO is a ZO-SGD algorithm that estimates gradients based on the two-point SPSA estimator introduced in (5).

Definition 2.1. *(ZO-SGD)* Consider solving optimization (1). ZO-SGD is an iterative ZO optimizer characterized with update rule

$$\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \eta \bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}^{(t)}),$$

for learning rate $\eta > 0$, and SPSA estimator $\bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}^{(t)})$ over minibatch $\mathcal{I} \in [n]$.

Implementing a vanilla ZO-SGD algorithm requires twice the memory footprint of inference due to the need to store the perturbation vector $\mathbf{z} \in \mathbb{R}^d$. In Malladi et al. (2023), an in-place implementation of the algorithm is proposed, where the requirement of storing a full set of perturbation scalars is mitigated by merely storing a single random seed and regenerating the perturbation vector when required. This brings the memory cost of MeZO down to that of inference (see Appendix E.1 for more details on the implementation).

2.3. ZO-SVRG

The Zeroth-Order Stochastic Variance Reduced Gradient (ZO-SVRG) (Liu et al., 2018) method periodically combines a fullbatch gradient estimator with the minibatch estimator to mitigate the stochasticity of the latter. This variance reduction helps achieve a faster convergence rate compared to ZO-SGD (Liu et al., 2018). While the full algorithm is presented in Appendix C, the update rule is:

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta[\hat{\nabla} f_{\mathcal{I}_t}(\boldsymbol{\theta}^{(t)}) - \hat{\nabla} f_{\mathcal{I}_t}(\bar{\boldsymbol{\theta}}) + \hat{\nabla} f(\bar{\boldsymbol{\theta}})] \tag{7}$$

where $\eta > 0$ is the learning rate, $\mathcal{I}_t$ is a minibatch sampled at iteration t, $\boldsymbol{\theta}^{(t)}$ is the parameter state at iteration t, and $\bar{\boldsymbol{\theta}}$ is the last parameter state at which the fullbatch gradient estimator was computed. Throughout this work, we let $q \in \mathbb{N}$ denote the regularity of fullbatch SPSA computations, i.e. every q steps the fullbatch SPSA estimator is computed.

3. Our proposed method: MeZO-SVRG

In this section, we describe the proposed MeZO-SVRG method. We first motivate our method by discussing the observed limitations of MeZO and outline practical implementation concerns when using ZO-SVRG (Liu et al., 2018) to mitigate these. We then introduce MeZO-SVRG as a variant of ZO-SVRG that minimizes memory usage with in-place operations and accommodates data parallelism in its gradient estimators.

3.1. MeZO Limitations

In Malladi et al. (2023), authors mention that MeZO requires a suitable task prompt to perform well; under this setting the optimization trajectory is more well-behaved. This suggests that the applicability of MeZO is restricted to settings where the optimization landscape is sufficiently well-behaved and cannot be extended to more complex tasks such as pre-training. Moreover, the careful design of prompts for real-world fine-tuning tasks also demands additional effort and may not always be practical. This motivates developing a method that delivers robust performance independently of any reliance on input prompts.

While MeZO has demonstrated promise in fine-tuning settings, our empirical findings suggest that it still faces the following challenges: i) it is susceptible to instability when using smaller batch sizes, and ii) a considerable performance gap with respect to first-order (FO) fine-tuning exists in the non-prompted setting. We illustrate these issues in Figures

1a, 1b and 1c. The details of the experiments are provided in Appendix B. These observations motivate using variance-reduction techniques that leverage larger batch information to improve stability and convergence of ZO methods in the large-scale problem settings.

3.2. ZO-SVRG Implementation Concerns

Memory Footprint. Recalling $\boldsymbol{\theta} \in \mathbb{R}^d$, the ZO-SVRG method has a minimum memory requirement of storing d values. A naive implementation of ZO-SVRG presented in Algorithm 2 (see Appendix C) would require an additional $2d$ of memory space for storing the fullbatch gradient estimator and parameter state $\bar{\boldsymbol{\theta}}$ used in (7). Moreover, computing and storing $\hat{\nabla} f_{\mathcal{I}_t}(\boldsymbol{\theta}^{(t)})$ and $\hat{\nabla} f_{\mathcal{I}_t}(\bar{\boldsymbol{\theta}})$ also accrues an additional d values of memory each. Thus, a naive implementation of Algorithm 2 would require a minimum memory budget equivalent to $5\times$ the memory budget of inference, which is prohibitive for sufficiently large d.

Iteration Speed Concerns. The original ZO-SVRG method is proposed with the inefficient gradient estimators introduced in (3) and (4). In both, SPSA estimators are computed for individual samples and averaged over the batch. Consider computing (3) with batch size b. If we want to fully parallelize operations, we require computing and storing b many $\hat{\nabla} f_i(u)$ estimators. However, this increases the memory footprint. To save on memory usage, in-place operations can be used. However, this has the effect of drastically reducing the computation speed as we need to sequentially compute each of the b estimators in (3).

3.3. MeZO-SVRG

We propose MeZO-SVRG: a variant of ZO-SVRG that improves iteration speed by using estimators (5), (6) and reduces the memory footprint with in-place operations. The method is summarized in Algorithm 1.

Efficient Gradient Estimation. We utilize the efficient gradient estimators introduced in (5) and (6) that perturb the entire batch in a single direction. These estimators accommodate data parallelism offered by modern ML frameworks. Furthermore, we can utilize the "resampling trick" introduced in Malladi et al. (2023) to reduce the memory footprint when computing each of (5) and (6); each estimator requires a memory footprint equivalent to the problem dimension d (see Appendix E.1 for the memory-efficient SPSA computation procedure). Thus, using estimators (5) and (6), eliminates the memory/speed trade-off plaguing the ZO-SVRG implementation and get the best of both worlds.

In-place Operations for Memory Efficiency. MeZO-SVRG leverages in-place operations to minimize memory allocation for new variable definitions. Memory space is required for the current state of the d parameters, a copy of

Algorithm 1 Memory-Efficient ZO-SVRG (MeZO-SVRG)

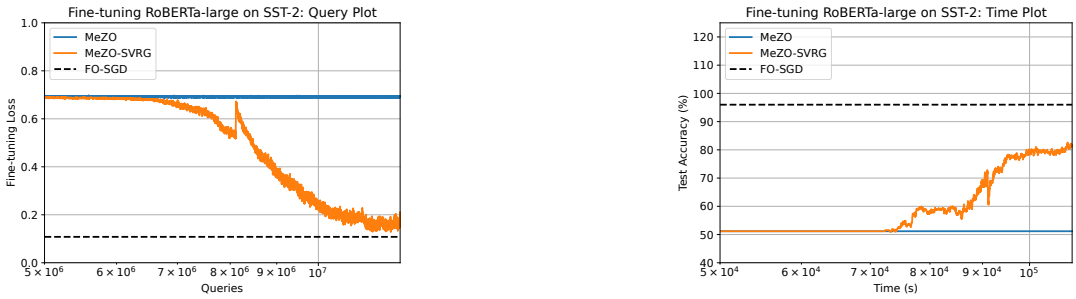
Input: Total iterations T, learning rates $\eta_1, \eta_2 > 0$, minibatch size b, parameters $\boldsymbol{\theta}_0$, iterations between full-batch gradient $q \in \mathbb{N}$
begin method
for $t = 0, \ldots, T$ **do**
  if $t \mod q = 0$ **then**
    1. $\mathbf{g} \leftarrow \hat{\bar{\nabla}} f(\boldsymbol{\theta}^{(t)})$
    2. $\bar{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^{(t)}$
    3. update: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_1 \mathbf{g}$ `#in-place`
  else
    4. Choose mini-batch $\mathcal{I}_t$ of size b
    5. $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_2 \hat{\bar{\nabla}} f_{\mathcal{I}_t}(\boldsymbol{\theta}^{(t)})$ `#in-place`
    6. $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t)} + \eta_2 \hat{\bar{\nabla}} f_{\mathcal{I}_t}(\bar{\boldsymbol{\theta}})$ `#in-place`
    7. update: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_2 \mathbf{g}$ `#in-place`
  end if
end for
end

the parameter state after each fullbatch SPSA computation as well as the fullbatch SPSA estimator itself. This requires a minimum memory requirement of storing $3d$ values. The minibatch updates can then be computed in-place in Lines 5, 6, and 7; thus, MeZO-SVRG achieves a reduced minimum memory footprint to $3\times$ that of inference. Note that this simplistic memory analysis does not account for any constant implementation overhead or intermediate activation storage during a forward pass of a network. Thus, in practice the memory usage ratio between MeZO and MeZO-SVRG is smaller (see Table 3).

Remark 3.1. As MeZO-SVRG queries the loss function with an inference pass through a network, it minimizes the storage of activations and intermediate variables. The memory footprint of MeZO-SVRG thus mainly stems from retaining copies of the fullbatch gradient estimator and parameters. Therefore, this method scales well with increasing batch sizes. Table 3 shows that for increasing batch sizes of up to 64, MeZO-SVRG yields more than 70% memory savings compared to first-order SGD (FO-SGD) on the RoBERTa-large (Liu et al., 2019) model. Similarly, MeZO-SVRG improves significantly on memory usage compared to FO-SGD for large context lengths and a fixed batch size (consistently $2\times$ smaller footprint, see Figure 3).

Remark 3.2. By storing $\bar{\boldsymbol{\theta}}$ in Line 2 (Algorithm 1), we can keep recomputing the fullbatch estimator on demand without storing $\mathbf{g}$. This would lower the memory footprint of MeZO-SVRG to $2\times$ that of inference. However, as computing the fullbatch estimator can slow down the iteration speed, throughout this work we store it.

Remark 3.3. In practice, fine-tuning datasets can be large enough that computing fullbatch SPSA estimators is infeasible (more than 10^5 training examples). MeZO-SVRG can be adapted so that the fullbatch estimator is approximated with a large batch estimator (e.g. 512 or 1024 samples).



(a) (b)

Figure 2. Performance of MeZO-SVRG, MeZO and FO-SGD when fine-tuning RoBERTa-large on the SST-2 (Socher et al., 2013) dataset. The dashed line serves as a reference to the training loss/test accuracy achieved by FO-SGD. (a) MeZO-SVRG is able to significantly reduce the convergence gap to FO-SGD compared to MeZO. (b) MeZO-SVRG attains a considerably better test accuracy than MeZO.

Additional Learning Rate. In Algorithm 1, we also include two independent learning rates η_1 and η_2 for the fullbatch and minibatch updates as shown in Lines 3 and Lines 5-7, respectively, of Algorithm 1. This design choice is based on our empirical observation that fullbatch updates are more accommodating of larger learning rates than minibatch steps. In our experiments we find that setting $\eta_1 > \eta_2$ improves convergence speed (see Appendices F.1, G.1, H.1).

Storage Efficiency of MeZO-SVRG. Parameter-efficient fine-tuning (PEFT) reduces the size of fine-tuned model checkpoints by optimizing only a small subset of parameters, e.g. LoRA (Hu et al., 2022) and prefix-tuning (Li & Liang, 2021). Both MeZO and MeZO-SVRG have the benefit of being able to recover an entire fine-tuning trajectory by storing a single seed and the difference of loss scalars in (5) at each step. The stored seed can regenerate step-wise seeds to recover the perturbation vectors $\mathbf{z}$ used for each SPSA computation. Together with the stored difference in loss values, we can recover the exact gradient estimators used in the fine-tuning process without needing to perform any forward passes. This allows recovering any model checkpoint along the fine-tuning trajectory. As we store only the initial random seed and a sequence of difference of loss scalars, we can achieve significant storage efficiency.

Compatibility with Non-differentiable Objectives and PEFT. As MeZO-SVRG uses only forward passes and a difference of loss values to estimate the gradient, it is applicable to settings where gradients are inaccessible or infeasible to compute, e.g. when considering non-differentiable objectives such as ranking in RLHF (Ouyang et al., 2022) or access to model gradients is restricted. Similar to MeZO, MeZO-SVRG also remains compatible with PEFT (e.g. LoRA (Hu et al., 2022), prefix-tuning (Li & Liang, 2021)).

4. Experiments

In this section, we evaluate MeZO-SVRG on a variety of fine-tuning tasks by comparing the performance against MeZO (Malladi et al., 2023) and memory usage against first-order stochastic gradient descent (FO-SGD) (Robbins & Monro, 1951) and first-order Adam (FO-Adam) (Kingma & Ba, 2015). We demonstrate empirically that MeZO-SVRG performs well in the absence of input prompts: it is able to significantly reduce the performance gap to FO methods and consistently surpasses MeZO's performance on a variety of fine-tuning tasks with significantly lower computation time. Furthermore, MeZO-SVRG necessitates a considerably smaller memory footprint compared to FO-SGD and FO-Adam. The code for the experiments is available at `https://github.com/amazon-science/mezo_svrg`.

Setup. We evaluate on both full (FP32) and half (BF16) precision. We detail the experiment results for the BF16 setting in Appendix J.We mainly consider a prompt-free fine-tuning setting (more challenging loss landscape) but include prompted results for RoBERTa-large (Liu et al., 2019) in Appendix G. All experiments are run on a single GPU; specifically, we consider Nvidia A100 40GB or H100 80GB GPUs. We evaluate the algorithms under two fine-tuning strategies: full- and partial-parameter fine-tuning. In the latter we fine-tune the last layers of the chosen models. We define a query as one forward pass for a single sample. For a fair comparison between MeZO and MeZO-SVRG, we ensured that the total number of queries used by both remains the same; thus, as MeZO-SVRG accrues more queries per step due to the fullbatch gradient estimates, MeZO was run for more steps. Further details of the experiment setup and implementation are provided in Appendices D and E.

Dataset. We fine-tune on tasks from the NLP GLUE and SuperGLUE benchmarks: Multi-Genre Natural Language Inference Corpus (MNLI), Stanford Question Answering Dataset (QNLI), Stanford Sentiment Treebank (SST-2),

Method	**DistilBert** Full-Precision (FP32)				**RoBERTa-large** Full-Precision (FP32)			
	MNLI	QNLI	SST-2	CoLA	MNLI	QNLI	SST-2	CoLA
MeZO (Full FT)	36 (1.09)	50 (0.69)	52 (0.68)	63 (0.64)	43 (0.94)	59 (0.58)	56 (0.69)	68 (0.51)
MeZO-SVRG (Full FT)	**46 (0.08)**	**68 (0.23)**	**72 (0.02)**	**68 (0.28)**	**49 (0.81)**	**80 (0.28)**	**84 (0.13)**	**79 (0.06)**
FO-SGD (Full FT)	59 (0.01)	78 (0.04)	88 (0.01)	70 (0.02)	85 (0.03)	89 (0.01)	96 (0.11)	85 (0.01)
MeZO (Partial FT)	35 (1.09)	52 (0.69)	51 (0.70)	60 (0.64)	42 (1.07)	50 (0.69)	54 (0.68)	65 (0.59)
MeZO-SVRG (Partial FT)	**47 (0.28)**	**65 (0.29)**	**74 (0.10)**	**67 (0.36)**	**43 (0.82)**	**67 (0.46)**	**72 (0.59)**	**79 (0.35)**
FO-SGD (Partial FT)	48 (0.26)	59 (0.42)	85 (0.05)	66 (0.45)	52 (0.99)	72 (0.60)	89 (0.58)	84 (0.41)

Table 1. Experiments on DistilBert and RoBERTa-large. We show the test accuracies and fine-tuning losses (in parentheses) of MeZO-SVRG and MeZO for both full/partial-parameter FT. We also provide results for FO-SGD as an upper-bound benchmark on performance. MeZO-SVRG consistently outperforms MeZO and significantly closes the gap to FO-SGD.

Method	**GPT2** Full-Precision (FP32)			**OPT-2.7B** Full-Precision (FP32)			**OPT-6.7B** Half-Precision (BF16)	
	MNLI	SST-2	CoLA	MNLI	SST-2	CoLA	SST-2	BoolQ
MeZO	41 (0.65)	59 (0.32)	61 (0.35)	42 (1.09)	61 (0.65)	62 (0.58)	74 (0.53)	65 (0.63)
MeZO-SVRG	**53 (0.41)**	**65 (0.20)**	**69 (0.25)**	**52 (0.81)**	**65 (0.55)**	**67 (0.53)**	**77 (0.52)**	**69 (0.57)**
FO-SGD	69 (0.59)	72 (0.23)	78 (0.38)	78 (0.33)	98 (0.02)	94 (0.17)	91 (0.10)	84 (0.29)

Table 2. Experiments on AR models. We show the test accuracies and fine-tuning losses (in parentheses) of MeZO-SVRG and MeZO for full-parameter FT. For reference we also provide results for FO-SGD as an upper-bound benchmark on performance. MeZO-SVRG consistently outperforms MeZO and approaches FO-SGD performance.

Corpus of Linguistic Acceptability (CoLA), and BoolQ (Williams et al., 2018; Wang et al., 2018; Socher et al., 2013; Warstadt et al., 2018; Wang et al., 2019). Similar to Malladi et al. (2023), for each task, our experiments are conducted in a many-shot fine-tuning setting: 512 training examples, 256 validation examples and 256 test samples are randomly sampled from the dataset.

Language Models. We considered Distilbert (Sanh et al., 2020) and RoBERTa-large as our masked LMs. Details on the hyperparameter configuration used for these experiments are provided in Appendix F, G and J. We extend our evaluation to fine-tuning larger autoregressive (AR) models. We consider the GPT2 (Radford et al., 2019), OPT-2.7B, and OPT-6.7B (Zhang et al., 2022) models. The hyperparameter configurations used for these experiments are detailed in Appendix H and J.

4.1. LM Fine-tuning Performance

MeZO-SVRG significantly outperforms MeZO in both the fine-tuning loss convergence and test accuracy. On all models and tasks, MeZO-SVRG improves on the test accuracy over MeZO: we see an improvement of up to 20% in Tables 1, 2 and Figure 2b. MeZO-SVRG also consistently achieves an improved fine-tuning loss compared to MeZO. This is particularly evident in Figure 2a. Additional results are presented in Appendices F, G and H.

MeZO-SVRG works well on both full and partial fine-tuning. The improvement over MeZO is consistent across both fine-tuning modes. In partial fine-tuning, MeZO-SVRG often achieves comparable performance to FO-SGD (within 5%) on several tasks (see Table 1).

MeZO-SVRG closes the gap to FO-SGD in training convergence and matches the test accuracy. Tables 1 and 2 demonstrate how MeZO-SVRG closes the performance gap with FO-SGD compared to MeZO.

MeZO-SVRG's superior performance to MeZO extends to the low (half) precision (BF16) setting. We summarize the half-precision results in Appendix J.

4.2. Memory Usage Profiling

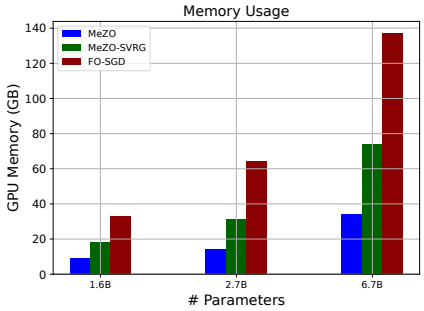


Figure 3. Shows the minimum memory usage on autoregressive models (batch size $= 1$, use max context length of model). MeZO-SVRG yields a $2\times$ smaller memory footprint over FO-SGD.

MeZO-SVRG can fit larger models on the same hard-

	Largest OPT/GPT that can fit		Memory Usage in GB for RoBERTa-large				
			Fixed context length (cl=128)			Fixed batch size (bs=64)	
Method	A100 (40GB)	H100 (80GB)	bs = 16	bs = 32	bs = 64	cl = 256	cl = 512
MeZO	6.7B	13B	2.07 (69%)	2.21 (79%)	2.51 (88%)	3.35	5.97
MeZO-SVRG	**2.7B**	**6.7B**	**4.36 (35%)**	**4.51 (58%)**	**4.72 (76%)**	5.13	8.02
FO-SGD	1.6B	2.7B	6.74	10.67	18.55	OOM	OOM
FO-Adam	350M	1.3B	10.44	14.33	22.41	OOM	OOM

Table 3. Shows the largest AR models that can fit on single 40, 80GB GPUs. We also measure the memory usage under different batch sizes (bs) and context lengths (cl) when fine-tuning RoBERTa-large. Percentages indicate the memory savings with respect to FO-SGD.

	GPT2				OPT-2.7B			
Method	MNLI	QNLI	SST-2	CoLA	MNLI	QNLI	SST-2	CoLA
MeZO	0.4	5.5	19.4	2.8	2.6	5.3	48	55
MeZO-SVRG	**0.3**	**1.9**	**5.6**	**2.2**	**1.1**	**2.7**	**25**	**1.4**

Table 4. Required GPU-hrs to achieve equivalent performance levels for MeZO-SVRG and MeZO.

ware than FO-SGD. We measure the minimum memory requirement to fine-tune (full-parameter) the considered autoregressive models using the different methods. We fine-tune GPT2, OPT-2.7B and OPT-6.7B on MNLI by setting the input sequence length to the maximum context length of the LM and report the peak GPU memory consumption for batch size $= 1$. Table 3 shows that *MeZO-SVRG consistently yields a significantly improved memory footprint compared to FO-SGD (approximately* $2\times$ *across considered autoregressive models)*. More details on how memory profiling was done is summarized in Appendix I.1.

MeZO-SVRG's memory savings progressively improve over FO-SGD and FO-Adam with increasing batch size and context lengths. For this experiment, we consider the masked model RoBERTa-large. Again we fine-tune on the MNLI dataset using a single Nvidia A100 40GB GPU and set the input sequence length to a constant size of 128. We measure the peak GPU memory consumption for the different methods for varying batch sizes $\{16, 32, 64\}$. Figure 3 shows that for a fixed model (RoBERTa-large) and context length (128), MeZO-SVRG exhibits memory savings of up to 76% w.r.t FO-SGD. We also vary the context lengths $\{256, 512\}$ of the input for a fixed batch size (64). Again we observe significant benefits for MeZO-SVRG over FO-SGD: the latter is subject to out-of-memory errors when running this setting with 40GB GPUs.

4.3. Computation Time

We compare the speed of MeZO-SVRG and MeZO by measuring the total GPU-hours required to achieve a certain performance threshold. For a fair comparison, we set the threshold to a level attained by both methods, namely, MeZO's peak test accuracy. Table 4 shows that for GPT2 and OPT-2.7B, MeZO-SVRG consistently achieves superior test accuracy with less than half the GPU-hours.

4.4. Understanding MeZO-SVRG

To better understand how the perturbation scale μ and regularity of full-batch update steps determined by q impact the MeZO-SVRG performance, we perform ablation studies in Appendix E with DistilBert on the MNLI dataset. For large fine-tuning datasets, estimating the full-batch gradient can be impractical. Therefore, we included an ablation study to examine the impact on MeZO-SVRG performance when substituting the full-batch gradient estimator with a large-batch estimator. Results in Table 10 suggest that large-batch estimators pose an effective alternative to full-batch estimators.

5. Convergence Theory

In this section, we provide a convergence analysis of MeZO-SVRG. We start by showing that our estimator is unbiased w.r.t. a minibatch set $\mathcal{I}$. We assume $\mathcal{I}$ is drawn either uniformly random with or without replacement.

Lemma 5.1.

$$\mathbb{E}_{\mathcal{I}} \bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) = \bar{\nabla} f(\boldsymbol{\theta}) \tag{8}$$

We denote $\mathbf{u}_{\mathcal{I}} = \bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) - \bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}') - \mathbb{E}_{\mathcal{I}}[\bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) - \bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}')]$ and $\mathbf{u}_{\mathcal{I}} = \mathbf{u}_i$ for $\mathcal{I} = \{i\}$. This $\mathbf{u}_i$ is a key component from the idea of control covariates (Tucker et al., 2017) in reducing variance.

Lemma 5.2. $\sum_{i=1}^{n} \mathbf{u}_i = 0$ *and* $\mathbb{E}_{\mathcal{I}}[\mathbf{u}_i \mathbf{u}_j] = 0$ *where* $i, j \in \mathcal{I}$ *and* $i \neq j$.

Assumptions. A1: Functions $\{f_i\}$ are L-smooth, i.e.,

$\|\nabla f_i(\boldsymbol{\theta}) - \nabla f_i(\boldsymbol{\theta}')\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2^2$. A2: The variance of stochastic gradients is bounded as $\frac{1}{n}\sum_{i=1}^n \|f_i(\boldsymbol{\theta}) - f(\boldsymbol{\theta}')\|_2^2 \leq \sigma^2$.

With the two Lemmas and assumptions, the following holds.

Theorem 5.3. *Assume A1 and A2 holds. Let learning rates $\eta = \eta_1 = \eta_2$. Then, MeZO-SVRG satisfies*

$$\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^{(T)})\|_2^2] \leq \frac{f(\boldsymbol{\theta}^{(0)}) - f^*}{T\bar{\gamma}} + \frac{L\mu^2}{T\bar{\gamma}} + \frac{c}{q\bar{\gamma}} \quad (9)$$

where $\bar{\gamma}$ and c are functions of learning rate η, dimension d, minibatch size b and L, σ. Moreover, by setting

$$\mu = \frac{1}{\sqrt{dT}}, \quad \eta = \frac{\rho}{L}, \quad q = \lceil \frac{d}{31\rho} \rceil,$$

where ρ is a universal constant, MeZO-SVRG satisfies

$$\mathbb{E}[\|\nabla f(\boldsymbol{\theta}^{(T)})\|_2^2] = O\left(\frac{d}{T} + \frac{\mathbf{1}(b < n)}{b}\right). \quad (10)$$

Theorem 5.3 demonstrates a linear convergence, inverse proportional to iteration T and q. The second term in Eq. (9) expresses the effect of μ, the magnitude of perturbation, which is small in practice. In Eq. (10), q is proportional to the problem dimension d, which can balance overall computational cost. It also reveals the effect of batch size b, indicating larger batch sizes are preferred in terms of iteration counts, which coincide with our empirical observation.

Remark 5.4. The derivation of Theorem 5.3 heavily relies on mathematical machinery and flows of original SVRG (Johnson & Zhang, 2013) and ZO-SVRG (Liu et al., 2018). However, note the gradient estimators in MeZO-SVRG and ZO-SVRG are different, e.g. different scaling, a single perturbation vector $\mathbf{z}$ vs multiple perturbation vectors $\{\mathbf{z}_i\}$ against `RandGradEst` (Liu et al., 2018). This requires careful examination often with different derivation like Lemma 5.1, 5.2, while making sure random vector $\mathbf{z}$ is conditioned consistently over sequence of derivations in (Liu et al., 2018). A proof sketch clarifying main distinct steps is provided in Appendix A.

6. Related work

Zeroth-Order Optimization. Zeroth-order (ZO) methods solve optimization problems without using gradient information. This class of methods typically estimates the gradient from function queries. Convergence theory has been developed for ZO stochastic gradient descent (ZO-SGD) in both convex (Jamieson et al., 2012; Raginsky & Rakhlin, 2011; Duchi et al., 2013) and non-convex settings (Liu et al., 2018; Ji et al., 2019; Park et al., 2020). However, these bounds generally depend on the number of parameters d. In Malladi et al. (2023), authors demonstrate via fine-tuning experiments that after pre-training and the inclusion of task prompts, the loss landscape is well-behaved enough and can be traversed by ZO-SGD. Zhang et al. (2024) benchmarks the performance of ZO methods in the context of LM fine-tuning. However, despite various advances on variance-reduced techniques (Johnson & Zhang, 2013; Defazio et al., 2014; Park & Ryu, 2020; Lu et al., 2021) within the FO class, to the best of our knowledge, we are the first to explore the direction of variance-reduced ZO optimization for fine-tuning LMs.

Memory-efficient Backpropagation Strategies. LLMs are typically fine-tuned by using FO methods such as SGD (Robbins & Monro, 1951) and Adam (Kingma & Ba, 2015). Several methods have been proposed to handle the memory overheads of backpropagation, for e.g. sparsifying gradients (Sun et al., 2017; Wei et al., 2017) and quantizing gradients to lower bit precisions (Dettmers et al., 2022b;a). Other techniques to save activation memory during forward and backward pass include Gradient checkpointing (Chen et al., 2016) and Flash Attention (Dao et al., 2022).

Gradient-free Adaptation of LLMs. The pre-training stage gives LLMs the ability to generalize to tasks for which it has not been explicitly trained. This form of adaptation requires instruction prompts and is referred to as *in-context learning* (ICL). While ICL enables quick adaptation of the model to specific tasks, drawbacks of this approach include that current models are constrained to limited context window and are sensitive to both the choice of input prompts and demonstrations (Malladi et al., 2023). Moreover, it has been empirically demonstrated that ICL on large models performs worse than full fine-tuning on medium-scale models (Brown et al., 2020). Another paradigm of adapting LLMs without using gradients is by using evolutionary algorithms Sun et al. (2022b;a), however the effectiveness of these methods has not been verified beyond smaller LMs.

7. Conclusion

This work introduces MeZO-SVRG: a variance-reduced ZO method that addresses the challenge of fine-tuning LMs under memory constraints. MeZO-SVRG is a variant of ZO-SVRG that exploits in-place operations for memory-frugality and efficient gradient estimators that accommodate data parallelism for significant improvement in the iteration speed. The method combines fullbatch and minibatch information to yield low variance gradient estimators. We demonstrate empirically that MeZO-SVRG outperforms MeZO consistently on a variety of LM fine-tuning tasks, even in a challenging non-prompted setting, and requires significantly less GPU-hours to achieve this performance. Furthermore, we show that across model types and fine-tuning tasks, MeZO-SVRG is able to considerably close

the performance gap to first-order methods while benefiting from a $2\times$ reduction in memory utilization, which progressively improves with larger batch sizes and context lengths.

We are excited to further explore the potential of MeZO-SVRG. In particular, we aim to examine MeZO-SVRG's performance when coupled with PEFT (LoRA, prefix-tuning) and settings where gradient-information is unavailable, e.g. prompt-tuning black-box models that are accessible only through an API. Finally, our work paves the way for exploring a broader spectrum of variance reduction techniques for ZO methods in the context of LM fine-tuning.

Impact Statement

One of the main challenges associated with adapting foundation models to specialized domains is the prohibitive computational burden during the fine-tuning phase. The high computational cost restricts the widespread use of larger models in resource-constrained settings. This deters a wider consumer-base from reaping the benefits of large models and, in turn, limits the democratization of the technology. Moreover, such constraints can be detrimental to the research community as the large-scale computational resources required for model adaptation are available to only a small fraction of researchers and users. This challenge can be overcome by investigating optimization methods that leverage the memory frugality of inference passes to develop effective fine-tuning strategies.

References

Brown, T., Mann, B., Ryder, N., and Subbiah, M. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost, 2016.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=dXiGWqBoxaD.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. 8-bit optimizers via block-wise quantization, 2022b.

Duchi, J., Jordan, M., Wainwright, M., and Wibisono, A. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61, 12 2013. doi: 10.1109/TIT.2015.2409256.

Ghadimi, S. and Lan, G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. doi: 10.1137/120880811. URL https://doi.org/10.1137/120880811.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jamieson, K. G., Nowak, R., and Recht, B. Query complexity of derivative-free optimization. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/e6d8545daa42d5ced125a4bf747b3688-Paper.pdf.

Ji, K., Wang, Z., Zhou, Y., and Liang, Y. Improved zeroth-order variance reduced algorithms and analysis for nonconvex optimization. *CoRR*, abs/1910.12166, 2019. URL http://arxiv.org/abs/1910.12166.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. 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. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., and Amini, L. Zeroth-order stochastic variance reduction for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 3731–3741, Red Hook, NY, USA, 2018. Curran Associates Inc.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Lu, Y., Park, Y., Chen, L., Wang, Y., De Sa, C., and Foster, D. Variance reduced training with stratified sampling for forecasting models. In *International Conference on Machine Learning*, pp. 7145–7155. PMLR, 2021.

Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., and Arora, S. Fine-tuning large language models with just forward passes. *https://arxiv.org/abs/2305.17333*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022.

Park, Y. and Ryu, E. K. Linear convergence of cyclic saga. *Optimization Letters*, 14(6):1583–1598, 2020.

Park, Y., Rossi, R., Wen, Z., Wu, G., and Zhao, H. Structured policy iteration for linear quadratic regulator. In *International Conference on Machine Learning*, pp. 7521–7531. PMLR, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Raginsky, M. and Rakhlin, A. Information-based complexity, feedback and dynamics in convex programming. *IEEE Transactions on Information Theory*, 57(10):7036–7056, 2011. doi: 10.1109/TIT.2011.2154375.

Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 1986.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S. (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., McCain, M., Newhouse, A., Blazakis, J., McGuffie, K., and Wang, J. Release strategies and the social impacts of language models, 2019.

Spall, J. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi: 10.1109/9.119632.

Sun, T., He, Z., Qian, H., Zhou, Y., Huang, X., and Qiu, X. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of EMNLP*, 2022a.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*, 2022b.

Sun, X., Ren, X., Ma, S., and Wang, H. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3299–3308. PMLR, 06–11 Aug

2017. URL https://proceedings.mlr.press/v70/sun17c.html.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Linzen, T., Chrupała, G., and Alishahi, A. (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. *SuperGLUE: a stickier benchmark for general-purpose language understanding systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Wei, B., Sun, X., Ren, X., and Xu, J. Minimal effort back propagation for convolutional neural networks. *ArXiv*, abs/1709.05804, 2017. URL https://api.semanticscholar.org/CorpusID:38548539.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, 2018.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. Opt: Open pre-trained transformer language models, 2022.

Zhang, Y., Li, P., Hong, J., Li, J., Zhang, Y., Zheng, W., Chen, P.-Y., Lee, J. D., Yin, W., Hong, M., Wang, Z., Liu, S., and Chen, T. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark, 2024.

A. Proof of Theorem

Throughout the proof, we drop bold notations $\boldsymbol{\theta}, \mathbf{z}, \mathbf{u} \to \theta, z, u$ for notational simplicity.

Lemma A.1.

$$\mathbb{E}_{\mathcal{I}} \bar{\nabla} f_{\mathcal{I}}(\theta) = \bar{\nabla} f(\theta) \tag{11}$$

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathcal{I}} \bar{\nabla} f_{\mathcal{I}}(\theta) &= \frac{1}{b} \mathbb{E}_{\mathcal{I}} \sum_{i \in \mathcal{I}} \frac{f_i(\theta + \mu z) - f_i(\theta - \mu z)}{2\mu} z \\
&= \frac{1}{b} \frac{b}{n} \sum_{i=1}^{n} \frac{f_i(\theta + \mu z) - f_i(\theta - \mu z)}{2\mu} z \\
&= \bar{\nabla} f(\theta)
\end{aligned}$$

The first and third equality comes from the definition of $\bar{\nabla} f_{\mathcal{I}}, \bar{\nabla} f$ and the second equality holds due to re-ordering under the assumption a minibatch set is sampled uniformly random or random with permutation. □

We denote $u_{\mathcal{I}} = \bar{\nabla} f_{\mathcal{I}}(\theta) - \bar{\nabla} f_{\mathcal{I}}(\theta') - \mathbb{E}_{\mathcal{I}}[\bar{\nabla} f_{\mathcal{I}}(\theta) - \bar{\nabla} f_{\mathcal{I}}(\theta')]$ and $u_{\mathcal{I}} = u_i$ for $\mathcal{I} = \{i\}$.

Lemma A.2. $\sum_{i=1}^{n} u_i = 0$ *and* $\mathbb{E}_{\mathcal{I}}[u_i u_j] = 0$ *where* $i, j \in \mathcal{I}$ *and* $i \neq j$.

Proof. By definition, $\sum_{i=1}^{n} \bar{\nabla} f_i(\theta) = n \bar{\nabla} f(\theta)$. It is immediate to see $\sum_{i=1}^{n} \mathbb{E}_{\mathcal{I}}[\bar{\nabla} f_{\mathcal{I}}(\theta)] = n \bar{\nabla} f(\theta)$, similar to Lemma A.1. Therefore $\sum_{i=1}^{n} u_i = 0$ holds. Conditioned on other randomness, e.g. perturbation z, $\mathbb{E}_{\mathcal{I}}[u_i u_j] = 0$ as i, j are independent. □

Assumptions. A1: Functions $\{f_i\}$ are L-smooth, i.e. $\|\nabla f_i(\theta) - \nabla f_i(\theta')\| \leq L \|\theta - \theta'\|_2^2$. A2: The variance of stochastic gradients is bounded as $\frac{1}{n} \sum_{i=1}^{n} \|f_i(\theta) - f(\theta')\|_2^2 \leq \sigma^2$.

Equipped with two Lemmas, the following holds

Theorem A.3. *Assume A1 and A2 holds. Let learning rate* $\eta = \eta_1 = \eta_2$. *Then, MeZO-SVRG satisfies*

$$\mathbb{E}[\|\nabla f(\theta^{(T)})\|_2^2] \leq \frac{f(\theta^{(0)}) - f^*}{T\bar{\gamma}} + \frac{L\mu^2}{T\bar{\gamma}} + \frac{c}{q\bar{\gamma}} \tag{12}$$

where $\bar{\gamma}$ *and* c *are the functions of stepsize* η, *dimension* d, *mini-batch size* b *and* L, σ. *Moreover, by setting*

$$\mu = \frac{1}{\sqrt{dT}}, \quad \eta = \frac{\rho}{L}, \quad q = \lceil \frac{d}{31\rho} \rceil$$

where ρ *is a universal constant, MeZO-SVRG satisfies*

$$\mathbb{E}[\|\nabla f(\theta^{(T)})\|_2^2] = O\left(\frac{d}{T} + \frac{\mathbf{1}(b < n)}{b}\right). \tag{13}$$

Proof. We rely on the proof provided by Liu et al. (2018). Note that we need to make sure that certain important steps and Lemmas still hold under MeZO-SVRG's gradient estimators. We start by using $d\bar{\nabla} f$ as our gradient estimate, through which Lemma 1 and 2 (in in (Liu et al., 2018)) hold by matching the scale of gradient to `RandGradEst` in (Liu et al., 2018). Lemma A.1 is used for Eq. (36) (Proposition 1 of Liu et al. (2018)). Lemma A.2 is used for Lemma 4, 5 in (Liu et al., 2018). Eq. (40) (Proposition 1 of (Liu et al., 2018)) holds because of a different conditional expectation, i.e., $E = E_z E_{\mathcal{I}|z} = E_z E_{\mathcal{I}}$, rather than $E = E_{\{z_i\}} E_{\mathcal{I}|\{z_i\}} = E_{\{z_i\}} E_{\mathcal{I}}$ where z and $\{z_i\}$ are random perturbations. The rest of proof follows through algebraic inequalities based on Lemmas 1,2, 4,5, and function assumptions, to derive convergence analysis. Finally we scale down learning rate η by d to adopt the gradient estimate of our definition. □

B. Exploring the Limits of MeZO Empirically

B.1. MNIST classification and RoBERTa-large fine-tuning

We ran experiments to better understand shortcomings in MeZO (Malladi et al., 2023). Two settings were considered: performing MNIST (LeCun et al., 1998) classification with a two-layer MLP (25K parameters) and fine-tuning RoBERTa-large (350M parameters) on the SST-2 (Socher et al., 2013) dataset. In the former, we used a two-layer feedforward network with 32 and 16 hidden units respectively. In the latter, we performed full-parameter fine-tuning. In Malladi et al. (2023), authors also remark that a simple instruction prompt is needed for the algorithm to succeed in fine-tuning tasks, i.e. it requires a sufficiently well-behaved optimization trajectory. While this, in itself, can be noted as a drawback, we adopted their proposed prompts in the experiment (Malladi et al., 2023). The training and fine-tuning runs are illustrated in Figures 1b and 1c. The hyperparameters selected for the runs are summarized in Tables 5 and 6. We paid particular attention to the effect of varying batch size on the algorithm performance. We also varied the perturbation scale μ used in the SPSA estimates (5). No improvement was found in reducing μ from the default setting used in MeZO ($\mu = 1e-3$) and thus we present results only for that configuration (Malladi et al., 2023). The largest learning rate values used in the grid search were selected for the MeZO runs. As an upper bound reference on performance, we also include the training curves for the FO-SGD algorithm. From both Figures 1b and 1c, it is clear the MeZO has to contend with instability incurred at smaller batch sizes.

Table 5. The hyperparameter grid optimized over in the initial the small-scale MNIST (LeCun et al., 1998) classification experiments.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{32, 64, 128\}\times$
	Learning rate	$\{1e-3, 1e-4\}\times$
	μ	$\{1e-3, 1e-4, 1e-5\}$
FO-SGD	Batch size	$\{64\}\times$
	Learning rate	$\{1e-3\}$

Table 6. The hyperparameter grid optimized over in the initial RoBERTa-large (Liu et al., 2019) fine-tuning experiments.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{16, 32, 64\}\times$
	Learning rate	$\{1e-5, 1e-6\}\times$
	μ	$\{1e-3, 1e-4, 1e-5\}$
FO-SGD	Batch size	$\{64\}\times$
	Learning rate	$\{1e-5\}$

B.2. Solving Least Squares

To make the aforementioned observations even more apparent, we examined the performance of MeZO on a simple linear least-squares (LS) problem. Specifically we solve

$$\min_{w\in\mathbb{R}^d} \|Xw - y\|_2^2, \tag{14}$$

where $X \in \mathbb{R}^{n\times d}$ is a randomly generated matrix, $w \in \mathbb{R}^d$ is fixed a priori, and $y \in \mathbb{R}^n = Xw +$ noise is the target labels. In our experiment, we focus on the 100-dimensional problem, i.e. with $d = 100$ and $n = 1000$. For comparison, we also report the performances of our proposed MeZO-SVRG and FO-SGD. The hyperparameter configurations used are presented in Table 7. Figure 1a makes it clear that MeZO is unable to attain the optimal value and yields a performance gap w.r.t. MeZO-SVRG and FO-SGD.

Table 7. The hyperparameters used for the Least Squares (LS) convergence experiment.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{32\}\times$
	Learning rate	$\{1e-3\}\times$
	μ	$\{1e-3\}$
MeZO-SVRG	Batch size	$\{32\}\times$
	Learning rate (η_1)	$\{1e-3\}\times$
	Learning rate (η_2)	$\{1e-4\}\times$
	μ	$\{1e-3\}\times$
	q	$\{2\}$
FO-SGD	Batch size	$\{32\}\times$
	Learning rate	$\{1e-3\}$

C. Zeroth-Order Stochastic Variance-Reduced Gradient

For the sake of completeness, we present the ZO-SVRG algorithm proposed in (Liu et al., 2018). This algorithm was proposed without a focus on memory efficiency, in contrast to our MeZO-SVRG, which offers significant memory-saving advantages, particularly in the context of fine-tuning large-scale LMs.

Algorithm 2 ZO-SVRG (Liu et al., 2018)

Input: Total iterations T, learning rate $\eta > 0$, minibatch size b, parameters $\boldsymbol{\theta}_0$, iterations between fullbatch estimators $q \in \mathbb{N}$
begin method
for $t = 0, \ldots, T$ **do**
  if $t \mod q = 0$ **then**
    1. $\mathbf{g} \leftarrow \hat{\nabla} f(\boldsymbol{\theta}^{(t)})$
    2. $\bar{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}^{(t)}$
  end if
  3. Choose mini-batch $\mathcal{I}_t$ of size b
  4. $\hat{\mathbf{g}} \leftarrow \hat{\nabla} f_{\mathcal{I}_t}(\boldsymbol{\theta}^{(t)})$
  5. $\bar{\mathbf{g}} \leftarrow \hat{\nabla} f_{\mathcal{I}_t}(\bar{\boldsymbol{\theta}})$
  6. Compute gradient blending: $\mathbf{v}_t \leftarrow \hat{\mathbf{g}} - \bar{\mathbf{g}} + \mathbf{g}$
  7. update: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \mathbf{v}^{(t)}$
end for
end

D. Experiment Setup

D.1. Datasets

For experiments on LMs, we considered fine-tuning on classification datasets. Specifically, we focused on the following datasets from the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmark: Multi-Genre Natural Language Inference (MNLI) (Williams et al., 2018), Question Natural Language Inference (QNLI) (Wang et al., 2018) for sentence pair classification, Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) for sentiment analysis, and Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2018). To incorporate a more challenging task, we also evaluated on the BoolQ dataset from the SuperGLUE (Wang et al., 2019) benchmark.

The datasets are imported from the Huggingface `datasets` library. We randomly sampled 512 examples for training, 256 for validation and 256 for testing.

D.2. Model

In our implementation, we used models from the Huggingface `transformers` package. As we considered classification datasets, we instantiated models from the `AutoModelsForSequenceClassification` and `OPTModelsForSequenceClassification` classes. These libraries add a classification head on top of the considered pre-trained model. For the prompted experiment setting, we instantiate from the `RobertaModelForPromptFinetuning` custom class implemented in the MeZO repository (Malladi et al., 2023).

Tables 8 and 9 summarize the models that where considered in our experiments. For the masked models both full- and partial parameter fine-tuning was performed.

Model	Total Trainable Parameters ($\times 10^6$)	Partial FT Layers	Partial FT Nr. of Parameters ($\times 10^6$)
DistilBert (`distilbert-base-cased`)	66	[`transformer.layer.5`, `classifier`]	8
RoBERTa-large (`roberta-large`)	355	[`roberta.encoder.layer.20`, `roberta.encoder.layer.21`, `roberta.encoder.layer.22`, `roberta.encoder.layer.23`, `classifier`]	38

Table 8. An overview of the masked LMs used in the experiments. Both full- and partial-parameter fine-tuning (FT) was considered for these LLMs.

Model	Total Trainable Parameters ($\times 10^6$)
GPT2 (`gpt2-xl`)	1557
OPT-2.7B (`facebook/opt-2.7B`)	2651
OPT-6.7B (`facebook/opt-6.7B`)	6658

Table 9. An overview of the autoregressive LMs used in the experiments.

E. MeZO-SVRG Implementation and Ablations

E.1. Memory-efficient SPSA

In our implementation we adopt the memory-efficient strategy of computing the SPSA estimator as proposed in Malladi et al. (2023). Rather than sampling and storing the entire perturbation vector $\mathbf{z} \in \mathbb{R}^d$, we sample a random seed and use it to regenerate the random vector when required. This allows in-place perturbations of the optimization parameters which minimizes the memory footprint. The memory-efficient perturbation routine is shown in 3. The parameters are perturbed in groups rather than individually, i.e. in Algorithm 3, each θ_i denotes a parameter group (e.g. an entire weight matrix). The scaling factor $s \in \{1, -2\}$ is used to perturb the parameters in a forward and backward direction as required in central difference approximations.

Algorithm 3 Memory-Efficient Parameter Perturbation

Design choices: Scaling factor $s \in \{1, -2\}$, perturbation size μ
Input: Parameters $\boldsymbol{\theta}$, random seed r
Return: Updated parameters $\boldsymbol{\theta}$

begin method
1. Set random seed r
for $\theta_i \in \boldsymbol{\theta}$ **do**
  2. $z_i \sim \mathcal{N}(0, 1)$
  3. $\theta_i \leftarrow \theta_i + s * z_i * \mu$
end for
end

In this work, experiments were conducted with single SPSA estimators which require exactly 2 forward passes. In p-SPSA, p estimators are computed and averaged. A total of $2p$ forward passes are required to compute a p-SPSA estimator. We used the default setting of $p = 1$ suggested in Malladi et al. (2023) for both MeZO and MeZO-SVRG implementations.

E.2. Role of the Perturbation Parameter

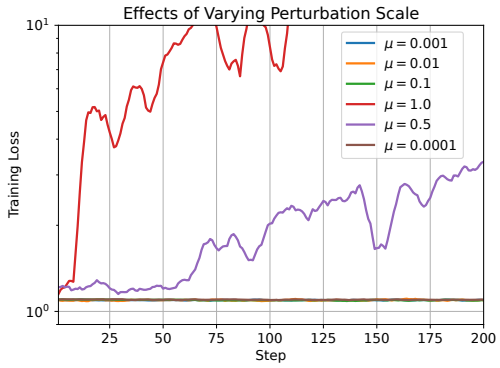
We investigated the role of the perturbation parameter μ in MeZO-SVRG. Recall that μ defines the forward and backward perturbation scale when computing SPSA estimators (5) and (6). We know from Spall (1992) that the SPSA estimator is asymptotically unbiased as $\mu \to 0$. We wanted to see the practical effects of different μ settings for MeZO-SVRG. Thus we carried out an ablation study where the perturbation parameter was varied. We fine-tune DistilBert (Sanh et al., 2020) on the MNLI (Williams et al., 2018) dataset. The experiment settings are summarized in Figure 4b.

Figure 4a shows how the different values of μ affected the fine-tuning process of the MeZO-SVRG algorithm. We observe that for a sufficiently small values of μ (i.e. smaller than $1e-1$) we see no noticeable difference in performance, while larger μ result in diverging behaviour. Similar findings were also empirically corroborated in Malladi et al. (2023). Thus, throughout our work we used the default value of $\mu = 1e-3$.

E.3. Role of q

The parameter q plays a significant role in the performance of MeZO-SVRG (Algorithm 1). Concretely, q determines the frequency of fullbatch update steps in the algorithm: smaller q increases the regularity of fullbatch updates. We perform an ablation to better understand the extent to which fullbatch updates help or hinder the MeZO-SVRG performance. We consider the task of fine-tuning the DistilBert (Sanh et al., 2020) model on the MNLI (Williams et al., 2018) dataset. The experiment setup is summarized in Figure 5b.

Figure 5a shows the training curves of MeZO-SVRG for different settings of q over 3500 steps. Increasing the frequency of fullbatch update steps enhances the convergence rate. However, our findings also indicate that a combination of fullbatch and minibatch updates (with $q \geq 2$) contributes to a more stable algorithm performance compared to exclusively using fullbatch updates (when $q = 1$).

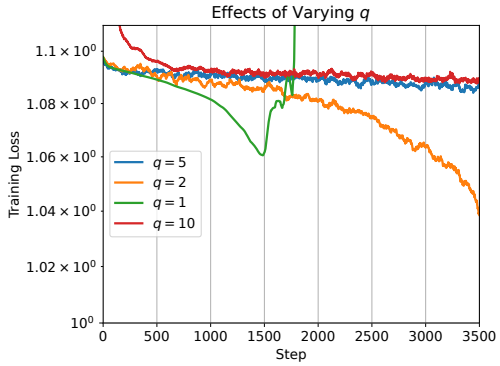


(a)

Algorithm	Hyperparameters	Values
MeZO-SVRG	Batch size	$\{64\}\times$
	Learning rate (η_1)	$\{1e-4\}\times$
	Learning rate (η_2)	$\{1e-6\}\times$
	μ	$\{1, 0.5, 1e-1, 1e-2, 1e-3, 1e-4\}\times$
	q	$\{2\}\times$
	Total Steps	$\{200\}$

(b)

Figure 4. a) Shows the effects of varying the perturbation scale on the performance of MeZO-SVRG. b) Shows the hyperparameter settings used in this experiment.



(a)

Algorithm	Hyperparameters	Values
MeZO-SVRG	Batch size	$\{64\}\times$
	Learning rate (η_1)	$\{1e-4\}\times$
	Learning rate (η_2)	$\{1e-6\}\times$
	μ	$\{1e-3\}\times$
	q	$\{1, 2, 5, 10\}\times$
	Total Steps	$\{3500\}$

(b)

Figure 5. a) Shows the effects of varying q on the convergence performance MeZO-SVRG. b) Shows the hyperparameter settings used in this experiment.

E.4. Improved Robustness to Batch Size

In Figures 1a, 1b and 1c we emphasize one of the practical drawbacks of MeZO with respect to instability with small batch sizes. We saw this behavior even in the more benign prompted setting. In Figure 6, we compare the behavior of MeZO-SVRG and MeZO when fine-tuning RoBERTa-large (Liu et al., 2019) on the SST-2 dataset in the prompt-free setting. The plot showcases MeZO-SVRG's advantage as a low-variance method with improved robustness to different batch sizes. In particular, MeZO's tendencies of diverging with smaller batch sizes are mitigated by MeZO-SVRG. Note that this improvement already becomes apparent over the first 100 iterations of fine-tuning.

E.5. Approximating Fullbatch Estimators with Large Batches

For sufficiently large training datasets, estimating the fullbatch gradient estimator is prohibitive and time-consuming. Thus we carry out an ablation study to see the effects on the MeZO-SVRG performance when approximating the fullbatch gradient estimator with a large-batch estimator. Specifically, we carry out partial-parameter fine-tuning of DistilBert on a training set of 512 samples for 8000 steps. We choose a mini-batch size of 64 which is consistent across experiment runs. This ablation study is carried out in the half-precision (BF16) setting. We approximate the fullbatch (512 samples) with large batch sizes of 256 and 128. The fine-tuning performances are summarized in Table 10. The obtained results are comparable, suggesting that the large batch-based gradient estimation offers a viable approximation of the fullbatch gradient estimator.

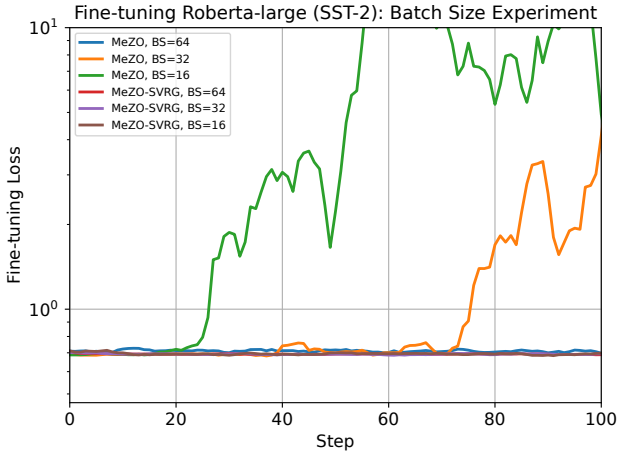


Figure 6. Shows improved robustness to smaller batch sizes for MeZO-SVRG compared to MeZO when fine-tuning RoBERTa-large on the SST-2 dataset.

Table 10. Performance of partial-parameter fine-tuning of DistilBert with half-precision when approximating the fullbatch with large batch sizes. Partial FT refers to partial-parameter fine-tuning (see Appendix D for details).

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
SST-2 (Full FT)	MeZO-SVRG (fullbatch$=512$)	0.4393	70	2560
	MeZO-SVRG (Large batch$=256$)	0.4946	71	1536
	MeZO-SVRG (Large batch$=128$)	0.5502	69	1024

E.6. Learning Rate Scheduling

In our implementation, we couple the MeZO-SVRG method with a basic learning rate annealing schedule. This schedule is shown in Algorithm 4. This scheduling scheme operates on feedback from training loss values. We compute the average loss values in consecutive epochs. If an increasing trend of average losses is observed, the learning rates are annealed with a factor of α. Specifically, if the ratio of leading and trailing average losses is above threshold κ, we anneal the learning rates. In our experiments we set $\kappa = 1.05$ and annealing factor $\alpha = 5$.

Algorithm 4 Learning Rate Scheduling for MeZO-SVRG

Input: Learning rates η_1, η_2, annealing factor α, losses L, annealing threshold κ, total number of batches in an epoch w
begin method
1. $m_1 \leftarrow \texttt{mean}(L[-w, :])$
2. $m_2 \leftarrow \texttt{mean}(L[-2w, -w])$
if $\frac{m_1}{m_2} > \kappa$ **then**
  3. $\eta_1 \leftarrow \frac{\eta_1}{\alpha}, \eta_2 \leftarrow \frac{\eta_2}{\alpha}$
end if
end

F. Fine-tuning DistilBert

F.1. Hyperparameter Selection

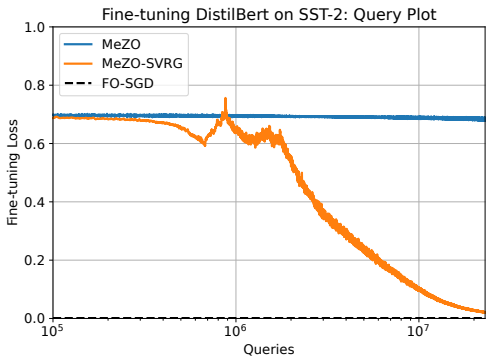
Table 11 shows the hyperparameter grid optimized over in the DistilBert (Sanh et al., 2020) experiment. The hyperparameter search was done by running the different algorithms for 1K steps on the MNLI (Williams et al., 2018) dataset and selecting the best configuration. The chosen configuration was then used for a longer fine-tuning runs for all considered tasks, i.e. 200K steps for MeZO and 50K steps for MeZO-SVRG.

Table 11. The hyperparameter grid optimized over for the DistilBert (Sanh et al., 2020) experiments. In the case of MeZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results.

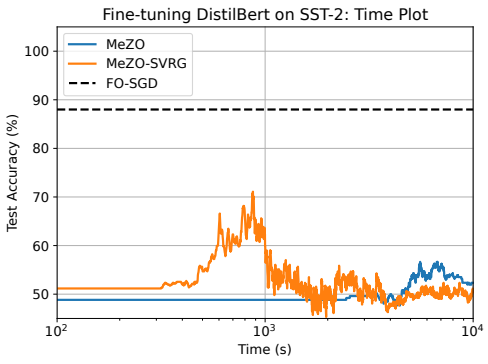
Algorithm	Hyperparameters	Values
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}4, 1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{200K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{\mathbf{1e{-}3}, 1e{-}4\}\times$
	Learning rate (η_2)	$\{1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	q	$\{\mathbf{2}, 5, 10\}\times$
	Total Steps	$\{\mathbf{50K}\}$
FO-SGD	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}2, \mathbf{1e{-}3}, 1e{-}4\}\times$
	Total Steps	$\{\mathbf{1K}\}$

F.2. Convergence Performance

We fine-tune Distilbert (Sanh et al., 2020) on the SST-2 (Socher et al., 2013) dataset. In Figure 7a, we show the improved convergence performance of MeZO-SVRG over MeZO. MeZO-SVRG is able to significantly reduce the convergence gap compared to the FO-SGD baseline. Figure 7b shows the evolution of the test accuracy over time. Observe that MeZO-SVRG achieves a significant improvement over MeZO in test performance. Moreover, MeZO-SVRG surpasses the peak test accuracy achieved by MeZO in over an order of magnitude less time.



(a)



(b)

Figure 7. Performance of MeZO-SVRG and MeZO when fine-tuning Distilbert (Sanh et al., 2020) on the SST-2 (Socher et al., 2013) dataset. The dashed line serves as a reference to the training loss/test accuracy achieved by FO-SGD. (a) MeZO-SVRG is able to significantly reduce the convergence gap to FO-SGD compared to MeZO. (b) MeZO-SVRG surpasses the peak test performance of MeZO in an order of magnitude less time.

F.3. Additional Results

Table 12. Experiments on DistilBERT (with 512 fine-tuning examples). FO refers to first-order methods. Full FT refers to full-parameter fine-tuning and Partial FT refers to partial-parameter fine-tuning (see Appendix D for details).

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI (Full FT)	MeZO	1.0908	36	25600
	MeZO-SVRG	**0.0757**	**46**	25600
	FO-SGD	0.0101	59	64
MNLI (Partial FT)	MeZO	1.0925	35	25600
	MeZO-SVRG	**0.2775**	**47**	25600
	FO-SGD	0.2617	48	64
QNLI (Full FT)	MeZO	0.6914	50	25600
	MeZO-SVRG	**0.2335**	**68**	25600
	FO-SGD	0.0372	78	64
QNLI (Partial FT)	MeZO	0.6929	52	25600
	MeZO-SVRG	**0.2925**	**65**	25600
	FO-SGD	0.4176	59	64
SST-2 (Full FT)	MeZO	0.6822	52	25600
	MeZO-SVRG	**0.0203**	**72**	25600
	FO-SGD	0.0121	88	64
SST-2 (Partial FT)	MeZO	0.6990	51	25600
	MeZO-SVRG	**0.1034**	**74**	25600
	FO-SGD	0.0507	85	64
CoLA (Full FT)	MeZO	0.6408	62	25600
	MeZO-SVRG	**0.2807**	**68**	25600
	FO-SGD	0.0159	70	64
CoLA (Partial FT)	MeZO	0.6422	60	25600
	MeZO-SVRG	**0.3617**	**67**	25600
	FO-SGD	0.44719	66	64

G. Fine-tuning RoBERTa-large

G.1. Hyperparameter selection

Table 13 presents the hyperparameters searched over in our RoBERTa-large (Liu et al., 2019) experiment. The hyperparameter search was done by fine-tuning the model on the MNLI (Williams et al., 2018) dataset for 1K steps and selecting the best configuration. This selected configuration was subsequently applied to extended fine-tuning sessions across all considered tasks. For our final results, MeZO-SVRG was run for 24K steps and MeZO was run for 96K steps.

Table 13. The hyperparameter grid optimized over for the RoBERTa-large (Liu et al., 2019) experiments. In the case of ZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}4, 1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{96K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{1e{-}4, 5e{-}5, \mathbf{1e{-}5}\}\times$
	Learning rate (η_2)	$\{1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	q	$\{\mathbf{2}, 5, 10\}\times$
	Total Steps	$\{\mathbf{24K}\}$
FO-SGD	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}3, \mathbf{1e{-}4}, 1e{-}5\}\times$
	Total Steps	$\{\mathbf{1K}\}$

G.2. Additional Results

Table 14. Experiments on RoBERTa-large (with 512 fine-tuning examples). Here partial refers to fine-tuning the last layers of the model (see Appendix D for details). FO refers to first-order methods. Full FT refers to full-parameter fine-tuning and Partial FT refers to partial-parameter fine-tuning.

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI (Full FT)	MeZO	0.9447	43	12288
	MeZO-SVRG	**0.8125**	**49**	12288
	FO-SGD	0.0292	85	64
MNLI (Partial FT)	MeZO	1.0729	42	12288
	MeZO-SVRG	**0.8176**	**43**	12288
	FO-SGD	0.9859	52	64
QNLI (Full FT)	MeZO	0.5845	59	12288
	MeZO-SVRG	**0.2750**	**80**	12288
	FO-SGD	0.01426	89	64
QNLI (Partial FT)	MeZO	0.6885	50	12288
	MeZO-SVRG	**0.4557**	**67**	12288
	FO-SGD	0.5974	72	64
SST-2 (Full FT)	MeZO	0.69155	56	12288
	MeZO-SVRG	**0.1336**	**84**	12288
	FO-SGD	0.1086	96	64
SST-2 (Partial FT)	MeZO	0.6837	54	12288
	MeZO-SVRG	**0.5896**	**72**	12288
	FO-SGD	0.5786	89	64
CoLA (Full FT)	MeZO	0.5062	68	12288
	MeZO-SVRG	**0.0644**	**79**	12288
	FO-SGD	0.0099	85	64
CoLA (Partial FT)	MeZO	0.5868	65	12288
	MeZO-SVRG	**0.3538**	**79**	12288
	FO-SGD	0.4075	84	64

Table 15. Experiments on RoBERTa-large (with 512 fine-tuning examples) in the prompted setting. Here partial refers to fine-tuning the last layers of the model (see Appendix D for details). FO refers to first-order methods. Full FT refers to full-parameter fine-tuning and Partial FT refers to partial-parameter fine-tuning.

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI with Prompt (Full FT)	MeZO	0.0076	73	12288
	MeZO-SVRG	**0.0058**	**75**	12288
	FO-SGD	0.0036	96	64
MNLI with Prompt (Partial FT)	MeZO	0.4614	65	12288
	MeZO-SVRG	**0.3177**	65	12288
	FO-SGD	0.3676	81	64
SST-2 With Prompt (Full FT)	MeZO	**0.2959**	**93**	12288
	MeZO-SVRG	0.3063	92	12288
	FO-SGD	0.1578	93	64
SST-2 with Prompt (Partial FT)	MeZO	**0.3280**	89	12288
	MeZO-SVRG	0.3393	89	12288
	FO-SGD	0.2981	90	64

H. Additional Results for fine-tuning Autoregressive Models

H.1. Hyperparameter Selection

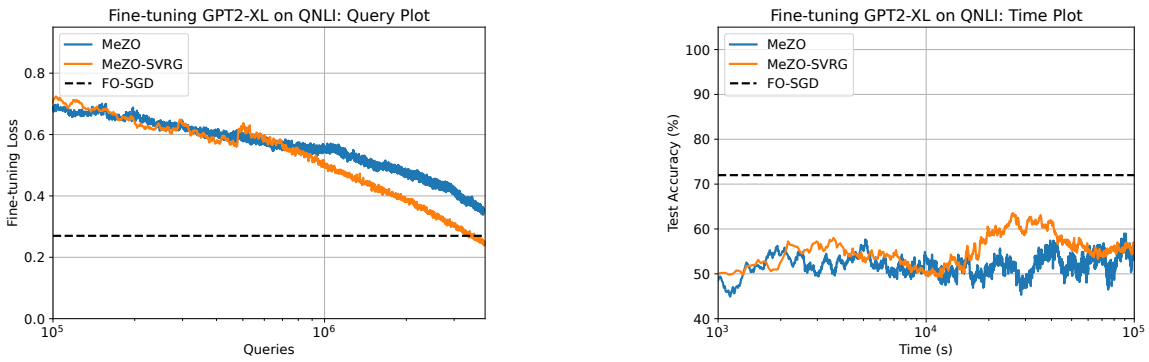
Table 16 presents the hyperparameter grid searched over for the experiments on autoregressive models. The hyperparameter search was conducted by fine-tuning the models on the MNLI (Williams et al., 2018) dataset for 100 steps and selecting the best configuration. This selected configuration was used in extended fine-tuning sessions across all considered tasks. For our final results, MeZO-SVRG was run for 8K steps and MeZO was run for 32K steps.

Table 16. The hyperparameter grid optimized over for the GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022) experiments. In the case of MeZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results for both models.

Algorithm	Hyperparameters	Values
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}6, \mathbf{5e{-}6}, 1e{-}7\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{32K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{1e{-}4, \mathbf{5e{-}5}, 1e{-}5\}\times$
	Learning rate (η_2)	$\{\mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	q	$\{\mathbf{2}, 5, 10\}\times$
	Total Steps	$\{\mathbf{8K}\}$
FO-SGD	Batch size	$\{8, \mathbf{16}\}\times$
	Learning rate	$\{\mathbf{1e{-}4}, 1e{-}5\}\times$
	Total Steps	$\{\mathbf{500}\}$

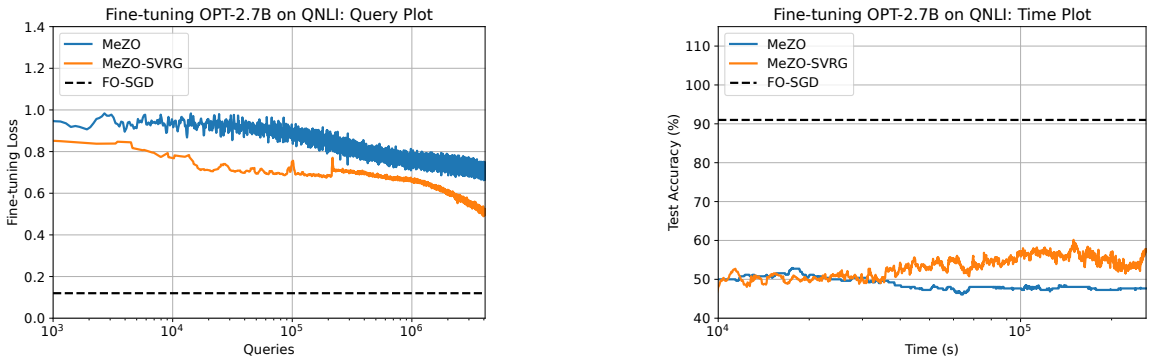
H.2. Convergence Performance

We fine-tune GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022) on the QNLI (Wang et al., 2018) dataset. In Figures 8a and 9a, we show the improved convergence performance of MeZO-SVRG over MeZO. For both models, MeZO-SVRG is able to significantly reduce the convergence gap compared to the FO-SGD baseline. Figures 8b and 9b show the evolution of the test accuracy over time. As with the experiments on masked models, MeZO-SVRG achieves a significant improvement over MeZO in test performance.



(a) (b)

Figure 8. Convergence performance of MeZO-SVRG, MeZO and FO-SGD when fine-tuning GPT2 (Radford et al., 2019) on the QNLI (Wang et al., 2018) dataset. The dashed line serves as a reference to the training loss achieved by FO-SGD. MeZO-SVRG is able to surpass the fine-tuning loss obtained by FO-SGD. It also improves on the test accuracy attained by MeZO.



(a) (b)

Figure 9. Performance of MeZO-SVRG, MeZO and FO-SGD when fine-tuning OPT-2.7B (Zhang et al., 2022) on the QNLI (Wang et al., 2018) dataset. The dashed line serves as a reference to the training loss/test accuracy achieved by FO-SGD. MeZO-SVRG is able to reduce the convergence gap to FO-SGD compared to MeZO and improve on the test accuracy.

H.3. Additional Results

Tables 17 and 18 present extended results on the fine-tuning tasks for GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022).

Table 17. Experiments on GPT2 (with 512 fine-tuning examples). FO refers to first-order methods. This table summarizes results for full-parameter fine-tuning.

Task	**Method**	**Fine-tuning Loss** ↓	**Test Accuracy** (%)↑	**Queries** ($\times 10^3$) ↓
MNLI (Full FT)	MeZO	0.6526	41	4096
	MeZO-SVRG	**0.4116**	**53**	4096
	FO-SGD	0.5924	69	8
QNLI (Full FT)	MeZO	0.3351	58	4096
	MeZO-SVRG	**0.2372**	**63**	4096
	FO-SGD	0.2799	72	8
SST-2 (Full FT)	MeZO	0.3240	59	4096
	MeZO-SVRG	**0.2024**	**65**	4096
	FO-SGD	0.2343	72	8
CoLA (Full FT)	MeZO	0.3544	68	4096
	MeZO-SVRG	**0.2455**	**69**	4096
	FO-SGD	0.3855	78	8

Table 18. Experiments on OPT-2.7B (with 512 fine-tuning examples). FO refers to first-order methods. This table summarizes results for full-parameter fine-tuning.

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI (Full FT)	MeZO	1.0875	42	4096
	MeZO-SVRG	**0.8159**	**52**	4096
	FO-SGD	0.3305	78	8
QNLI (Full FT)	MeZO	0.7026	53	4096
	MeZO-SVRG	**0.4634**	**60**	4096
	FO-SGD	0.1222	91	8
SST-2 (Full FT)	MeZO	0.6530	61	4096
	MeZO-SVRG	**0.5501**	**65**	4096
	FO-SGD	0.0167	98	8
CoLA (Full FT)	MeZO	0.5823	62	4096
	MeZO-SVRG	**0.5335**	**67**	4096
	FO-SGD	0.1724	94	8

I. Memory Usage and Computation Time

I.1. Memory Profiling

We performed memory profiling experiments without any advanced memory-saving options such as lowering bit precision (Dettmers et al., 2022b) or gradient check-pointing (Chen et al., 2016). We used full (f32) floating-point precision.

In the first experiment, we measured the memory requirement needed to run the different methods on full-parameter fine-tuning tasks. The MNLI (Williams et al., 2018) dataset was used to fine-tune autoregressive models GPT2 (Radford et al., 2019), OPT-2.7B, OPT-6.7B (Zhang et al., 2022). We set the input sequence length to the maximum context length for each model, i.e. 1024 for GPT2 and 2048 for the OPT models. The batch size was set to 1. Figure 3 shows the peak memory consumption in GB as reported by the `nvidia-smi` command. The peak memory consumption was obtained after executing the methods for at least 100 steps. Table 3 presents the largest GPT/OPT model that can be fit for each method under the aforementioned settings on single Nvidia A100 40GB and H100 80GB GPUs.

In the second experiment, we measured how the memory usage for the different methods scales with increasing batch size. We fine-tuned RoBERTa-large (Liu et al., 2019) on the MNLI (Williams et al., 2018) dataset. The input sequence length was set to a constant 128 and we varied the batch size $\{16, 32, 64\}$. The memory consumption was again measured using the `nvidia-smi` command and measurements were taken after running the methods for at least 100 steps. Table 3 summarizes the results.

We finally also measured how the memory usage varies for the considered algorithms when using a fixed batch size (64) and changing the context length of the input. We used a similar setting to the second experiment: fine-tuning RoBERTa-large (Liu et al., 2019) on the MNLI (Williams et al., 2018) dataset. The input context length was varied $\{128, 256, 512\}$ and the memory consumption was measured using the `nvidia-smi` command. Table 3 reports the results.

We replicated all experiments in the half-precision (BF16) setting; the results are given in Table 25.

I.2. Computation Time

We compared the speed of MeZO-SVRG and MeZO (Malladi et al., 2023) by measuring the time taken by each method to achieve the test performance attained by MeZO. These measurements are based on fine-tuning GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022) on all considered datasets. Table 4 summarizes the results.

J. Half-Precision Experiments

In the section, we run preliminary experiments to evaluate the considered fine-tuning algorithms on the half-precision (BF16) setting.

J.1. Half-Precision Experiments on DistilBert

The hyperparameter grid that was optimized over for the DistilBert experiments in the half-precision setting is presented in Table 19. As each iteration under the half-precision setting is faster than under the full-precision setting, we run experiments for longer. Specifically, we run MeZO-SVRG for 80K steps, MeZO for 400K steps and FO-SGD for 2K steps. The results are summarized in Table 20.

Table 19. The hyperparameter grid optimized over for the half-precision DistilBert (Sanh et al., 2020) experiments. In the case of MeZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}4, \mathbf{1e{-}5}, 1e{-}6\}\times$
	μ	$\{\mathbf{1e{-}2}\}\times$
	Total Steps	$\{\mathbf{400K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{\mathbf{1e{-}3}, 1e{-}4\}\times$
	Learning rate (η_2)	$\{\mathbf{1e{-}5}, 1e{-}6\}\times$
	μ	$\{\mathbf{1e{-}2}\}\times$
	q	$\{\mathbf{2}, 5\}\times$
	Total Steps	$\{\mathbf{80K}\}$
FO-SGD	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{\mathbf{1e{-}2}, 1e{-}3, 1e{-}4\}\times$
	Total Steps	$\{\mathbf{2K}\}$

Table 20. Half-precision experiments on DistilBERT (with 512 fine-tuning examples). FO refers to first-order methods. Partial FT refers to partial-parameter fine-tuning (see Appendix D for details).

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI (Partial FT)	MeZO	1.0892	43	51200
	MeZO-SVRG	**0.8746**	**45**	51200
	FO-SGD	0.3508	51	128
QNLI (Partial FT)	MeZO	0.6904	60	51200
	MeZO-SVRG	**0.5416**	**64**	51200
	FO-SGD	0.2998	66	128
SST-2 (Partial FT)	MeZO	0.6889	61	51200
	MeZO-SVRG	**0.3887**	**79**	51200
	FO-SGD	0.0555	82	128
CoLA (Partial FT)	MeZO	0.6420	66	51200
	MeZO-SVRG	**0.6170**	**71**	51200
	FO-SGD	0.4218	70	128

J.2. Half-Precision Experiments on RoBERTa-large

The hyperparameter grid that was optimized over for the DistilBert experiments in the half-precision setting is presented in Table 21. As each iteration under the half-precision setting is faster than under the full-precision setting, we run experiments for longer. Specifically, we run MeZO-SVRG for 40K steps, MeZO for 200K steps and FO-SGD for 1K steps. The results are summarized in Table 22.

Table 21. The hyperparameter grid optimized over for the half-precision RoBERTa-large (Liu et al., 2019) experiments. In the case of MeZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{\mathbf{64}\}\times$
	Learning rate	$\{1e{-}4, \mathbf{1e{-}5}, 1e{-}6\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{200K}\}$
MeZO-SVRG	Batch size	$\{\mathbf{64}\}\times$
	Learning rate (η_1)	$\{\mathbf{1e{-}4}, 1e{-}5\}\times$
	Learning rate (η_2)	$\{\mathbf{1e{-}5}, 1e{-}6\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	q	$\{\mathbf{2}, 5\}\times$
	Total Steps	$\{\mathbf{40K}\}$
FO-SGD	Batch size	$\{\mathbf{64}\}\times$
	Learning rate	$\{\mathbf{1e{-}2}, 1e{-}3, 1e{-}4\}\times$
	Total Steps	$\{\mathbf{1K}\}$

Table 22. Half-precision experiments on RoBERTa-large (with 512 fine-tuning examples). FO refers to first-order methods. Partial FT refers to partial-parameter fine-tuning (see Appendix D for details).

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$) $\downarrow$
MNLI (Partial FT)	MeZO	1.0898	42	25600
	MeZO-SVRG	**1.0695**	**43**	25600
	FO-SGD	0.1820	55	64
QNLI (Partial FT)	MeZO	0.6835	62	25600
	MeZO-SVRG	**0.6070**	**68**	25600
	FO-SGD	0.3112	67	64
SST-2 (Partial FT)	MeZO	0.6630	66	25600
	MeZO-SVRG	**0.5278**	**77**	25600
	FO-SGD	0.1356	93	64
CoLA (Partial FT)	MeZO	0.6308	66	25600
	MeZO-SVRG	**0.5781**	**69**	25600
	FO-SGD	0.1537	88	64

J.3. Half-Precision Experiments on OPT-6.7B

The hyperparameter grid optimized for the OPT-6.7B experiments in the half-precision setting is detailed in Table 23. We conducted the MeZO-SVRG experiments for 8k steps, MeZO for 24k steps, and FO-SGD for 1k steps. The outcomes of these experiments are summarized in Table 24. We include the BoolQ dataset from the SuperGLUE (Wang et al., 2019) benchmark to evaluate a more challenging fine-tuning task.

Table 23. The hyperparameter grid optimized over for the half-precision OPT-6.7B (Zhang et al., 2022) experiments. In the case of MeZO-SVRG we use the learning rate schedule proposed in Algorithm 4. The bold values indicate the configuration used to generate the final results.

Algorithm	**Hyperparameters**	**Values**
MeZO	Batch size	$\{\mathbf{128}\}\times$
	Learning rate	$\{1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{24K}\}$
MeZO-SVRG	Batch size	$\{\mathbf{128}\}\times$
	Learning rate (η_1)	$\{\mathbf{1e{-}4}, 1e{-}5\}\times$
	Learning rate (η_2)	$\{1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	q	$\{\mathbf{2}, 5\}\times$
	Total Steps	$\{\mathbf{8K}\}$
FO-SGD	Batch size	$\{\mathbf{64}\}\times$
	Learning rate	$\{1e{-}3, \mathbf{1e{-}4}\}\times$
	Total Steps	$\{\mathbf{1K}\}$

Table 24. Half-precision experiments on OPT-6.7B (with 512 fine-tuning examples). FO refers to first-order methods. Full FT refers to full-parameter fine-tuning (see Appendix D for details).

Task	**Method**	**Fine-tuning Loss** $\downarrow$	**Test Accuracy** (%)$\uparrow$	**Queries** ($\times 10^3$)$\downarrow$
SST-2 (Full FT)	MeZO	0.5318	74	6144
	MeZO-SVRG	**0.5278**	**77**	6144
	FO-SGD	0.103	91	128
BoolQ (Full FT)	MeZO	0.6259	65	6144
	MeZO-SVRG	**0.5703**	**69**	6144
	FO-SGD	0.2872	84	128

J.4. Memory Profiling with Half-Precision

		Memory Usage in GB for RoBERTa-large				
	Largest OPT/GPT that can fit	Fixed context length (cl=128)			Fixed batch size (bs=64)	
Method	A100 (40GB)	bs = 16	bs = 32	bs = 64	cl = 256	cl = 512
MeZO	13B	1.03	1.13	1.25	1.39	2.66
MeZO-SVRG	**6.7B**	**2.10 (39%)**	**2.11 (66%)**	**2.12 (79%)**	**2.27 (90%)**	**3.66**
FO-SGD	2.7B	3.42	5.81	9.83	21.87	OOM
FO-Adam	1.3B	5.85	8.07	12.16	24.29	OOM

Table 25. Memory profiling with half-precision. Shows the largest AR models that can fit on single 40 GPUs. We also measure the memory usage under different batch sizes (bs) and context lengths (cl) when fine-tuning RoBERTa-large. Percentages indicate the memory savings with respect to FO-SGD.