VARIANCE-REDUCED ZEROTH-ORDER METHODS FOR FINE-TUNING LANGUAGE MODELS

Tanmay Gautam[1,*] **Youngsuk Park**[2,†] **Hao Zhou**[2] **Parameswaran Raman**[2] **Wooseok Ha**[2]
[1]University of California, Berkeley [2]Amazon AI, Santa Clara, USA

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. More 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 on benchmark GLUE 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 as it often surpasses MeZO's peak test accuracy with a $2\times$ reduction in GPU-hours. MeZO-SVRG significantly reduces the required memory footprint compared to first-order SGD, i.e. by $2\times$ for autoregressive models. Our experiments highlight that MeZO-SVRG's memory savings progressively improve compared to SGD with larger batch sizes.

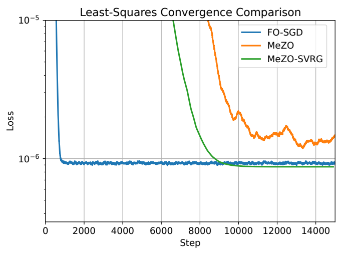
1 INTRODUCTION

Fine-tuning Language Models (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. 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.

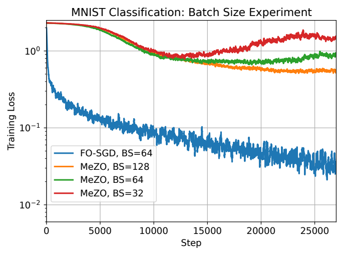
While ZO methods have been studied for decades Spall (1992); Ghadimi & Lan (2013), it is only recently that these have been applied to fine-tune LMs (Malladi et al., 2023). In Malladi et al. (2023), authors propose the Memory-Efficient Zeroth-Order Optimizer (MeZO) and demonstrate its superior performance against ICL with a memory footprint equivalent to that of inference. 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. We revisit ZO optimization under the standard (non-prompted) fine-tuning setting. Through empirical studies, 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

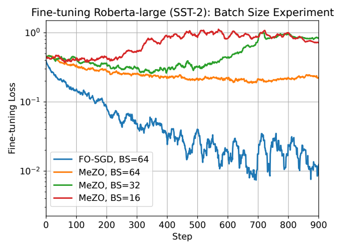
[*]Work conducted during an internship at Amazon. [†]Correspondence to: Youngsuk Park `<pyoungsu@amazon.com>`



(a)



(b)



(c)

Figure 1: (a) 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.

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

2. We fine-tune masked and autoregressive LMs on GLUE (Wang et al., 2018) tasks. MeZO-SVRG achieves consistent performance improvements with up to 20% increase in test accuracies over MeZO across all models and tasks. MeZO-SVRG stands out by consistently surpassing MeZO's test accuracy in only half as many GPU-hours.

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

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 equation 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 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.

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

$$\hat{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) := \frac{1}{b}\sum_{i\in\mathcal{I}} \hat{\nabla} f_i(\boldsymbol{\theta}) \quad \text{and} \quad \hat{\nabla} f(\boldsymbol{\theta}) := \hat{\nabla} f_{[n]}(\boldsymbol{\theta}). \tag{3}$$

The gradient estimator in equation 3 requires $2b$ function queries and sampling b random vectors. In practice, there are two strategies to compute estimators equation 3: 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 accommodates data parallelism: we perturb each samplewise SPSA estimator in the same direction $\mathbf{z} \in \mathbb{R}^d$. We can construct

$$\bar{\nabla} f_{\mathcal{I}}(\boldsymbol{\theta}) := \frac{\frac{1}{b}\sum_{i\in\mathcal{I}}[f_i(\boldsymbol{\theta}+\mu\mathbf{z}) - f_i(\boldsymbol{\theta}-\mu\mathbf{z})]}{2\mu}\mathbf{z} \quad \text{and} \quad \bar{\nabla} f(\boldsymbol{\theta}) := \bar{\nabla} f_{[n]}(\boldsymbol{\theta}). \tag{4}$$

From an implementation standpoint, estimators 4 can exploit data parallelism across the batch $\mathcal{I}$ and benefit from a minimal required memory footprint.

2.2 Memory-efficient ZO-SGD (MeZO)

Malladi et al. (2023) proposes the Memory-Efficient ZO Optimizer (MeZO): an adaptation of ZO-SGD with the same memory footprint as inference to fine-tune LMs.

Definition 2.1. *(ZO-SGD)* Consider solving optimization equation 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.

3 Our proposed method: MeZO-SVRG

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. 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.

3.2 MeZO-SVRG

We propose MeZO-SVRG: a variance-reduced ZO algorithm that enhances the stability and convergence of inference-based LM fine-tuning. MeZO-SVRG is a variant of ZO-SVRG (Liu et al., 2018) that improves iteration speed by using estimators 4 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 equation 4 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 equation 4; each estimator requires a memory footprint equivalent to the problem dimension d.

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

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. $\mathbf{g} \leftarrow \bar{\nabla} f(\boldsymbol{\theta}^{(t)})$
        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. $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_2 \bar{\nabla} f_{\mathcal{I}_t}(\boldsymbol{\theta}^{(t)})$   #in-place
        6. $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t)} + \eta_2 \bar{\nabla} f_{\mathcal{I}_t}(\bar{\boldsymbol{\theta}})$   #in-place
        7. update: $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta_2 \mathbf{g}$   #in-place
    end if
end for
end
```

state of the d parameters, a copy of 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 minimum memory footprint equal to $3\times$ that of inference.

4 EXPERIMENTS

In this section, we evaluate MeZO-SVRG on a variety of fine-tuning tasks by comparing against MeZO (Malladi et al., 2023) and stochastic gradient descent (FO-SGD) (Robbins & Monro, 1951).

Setup. We opt for SGD as our FO comparison benchmark, as it provides a more competitive bound on the memory utilization than Adam (Kingma & Ba, 2015). We mainly consider a prompt-free fine-tuning setting (more challenging loss landscape). All experiments are run on a single GPU (Nvidia A100 40GB or H100 80GB). 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. Further details of the experiment setup and implementation are provided in Appendices C and D.

Dataset. We fine-tune on 4 tasks/datasets from the NLP GLUE benchmark (Wang et al., 2018). 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.

Language Models. We considered Distilbert (Sanh et al., 2020), RoBERTa-large as our masked LMs, and GPT2 (Radford et al., 2019), OPT-2.7B, OPT-6.7B (Zhang et al., 2022) as our autoregressive LMs. Hyperparameter configurations for these experiments are given in Appendices E.1, F.1, G.1.

4.1 LM FINE-TUNING PERFORMANCE

	DistilBert				**RoBERTa-large**			
Method	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.

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. MeZO-SVRG also consistently achieves an improved fine-tuning loss compared to MeZO. Additional results are presented in Appendices E, F and G.

	GPT2				OPT-2.7B			
Method	MNLI	QNLI	SST-2	CoLA	MNLI	QNLI	SST-2	CoLA
MeZO	41 (0.65)	57 (0.36)	59 (0.32)	61 (0.35)	42 (1.09)	53 (0.70)	61 (0.65)	62 (0.58)
MeZO-SVRG	**53 (0.41)**	**63 (0.24)**	**65 (0.20)**	**69 (0.25)**	**52 (0.81)**	**60 (0.46)**	**65 (0.55)**	**67 (0.53)**
FO-SGD	69 (0.59)	72 (0.28)	72 (0.23)	78 (0.38)	78 (0.33)	91 (0.12)	98 (0.02)	94 (0.17)

Table 2: Experiments on GPT2 and OPT-2.7B. We show the test accuracies and fine-tuning losses (in parentheses) of MeZO-SVRG and MeZO for full-parameter FT.

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).

4.2 Memory Usage Profiling and Computation Time

	Minimum Memory Usage in GB (bs=1)			Memory Usage in GB for RoBERTa-large		
Method	GPT2 (cl=1024)	OPT-2.7B (cl=2048)	OPT-6.7B (cl=2048)	bs = 16	bs = 32	bs = 64
MeZO	9	14	34	2.07 (69%)	2.21 (79%)	2.51 (88%)
MeZO-SVRG	**18**	**31**	**74**	**4.36 (35%)**	**4.51 (58%)**	**4.72 (76%)**
FO-SGD	34	64	137	6.74	10.67	18.55

Table 3: We measure the minimum memory usage on the autoregressive models (batch size (bs) = 1, use max context length (cl) of model). We also measure the memory usage under different batch sizes (bs) when fine-tuning RoBERTa-large (Liu et al., 2019) with the methods. The percentages in the parentheses indicate the memory savings with respect to FO-SGD.

MeZO-SVRG can fit larger models on the same hardware 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 given in Appendix H.1.

MeZO-SVRG's memory savings progressively improve over FO-SGD with increasing batch size. 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 $\{16, 32, 64\}$. Table 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 compare the speed of MeZO-SVRG and MeZO by measuring the total GPU-hours required to achieve MeZO's peak test accuracy. Table 17 shows that for GPT2 and OPT-2.7B, **MeZO-SVRG consistently achieves superior test accuracy with less than half the GPU-hours.**

5 Conclusion

This work introduces MeZO-SVRG: a variance-reduced ZO method that addresses the challenge of fine-tuning LLMs under memory constraints. MeZO-SVRG is a variant of ZO-SVRG that exploits in-place operations for memory-frugality and gradient estimators that accommodate data parallelism for 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 SGD while benefiting from a $2\times$ reduction in memory utilization.

REFERENCES

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization, 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. doi: 10.1137/120880811. URL `https://doi.org/10.1137/120880811`.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 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`.

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

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

Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning large language models with just forward passes. *https://arxiv.org/abs/2305.17333*, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Herbert Robbins and Sutton Monro. 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`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (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`.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (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`.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. *SuperGLUE: a stickier benchmark for general-purpose language understanding systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 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.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

A Exploring the Limits of MeZO Empirically

A.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 4 and 5. 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 equation 4. 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 4: 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 5: 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	$\{1e-5\}$

A.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

$$\min_{w \in \mathbb{R}^d} \|Xw - y\|_2^2, \tag{5}$$

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 6. 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 6: 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	$\{1e-3\}$

B Zeroth-Order Stochastic Variance-Reduced Gradient

B.1 Algorithm Overview

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). The full algorithm is presented in Algorithm 2.

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
```

B.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 B) would require an additional $2d$ of memory space for storing the fullbatch gradient estimator and parameter state $\bar{\boldsymbol{\theta}}$. 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 equation 3. In both, SPSA estimators are computed for individual samples and averaged over the batch. Consider computing equation 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 equation 3.

C Experiment Setup

C.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).

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

C.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 7 and 8 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 Fine-tuning Layers
DistilBert (`distilbert-base-cased`)	66	[`transformer.layer.5`, `classifier`]
RoBERTa-large (`roberta-large`)	355	[`roberta.encoder.layer.20`, `roberta.encoder.layer.21`, `roberta.encoder.layer.22`, `roberta.encoder.layer.23`, `classifier`]

Table 7: An overview of the masked LMs used in the experiments. Both full- and partial-parameter fine-tuning 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 8: An overview of the autoregressive LMs used in the experiments.

D MeZO-SVRG Implementation and Ablations

D.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 $\mu$
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.

D.2 Role of the Perturbation Parameter

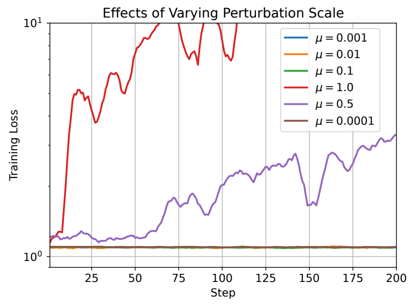
We investigated the role of the perturbation parameter μ in MeZO-SVRG. Recall that μ defines the forward and backward perturbation scale when computing SPSA estimators equation 4. We know from Spall (1992) that the SPSA estimator is asymptotically unbiased as $\mu \to 0$. We wanted to see the practical 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 2b.

Figure 2a 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 $\mu = 1e - 3$.

D.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 full-batch 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 3b.

Figure 3a shows the training curves of MeZO-SVRG for different settings of q over 3500 steps. Increasing the frequency of full-batch 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 $q = 1$).

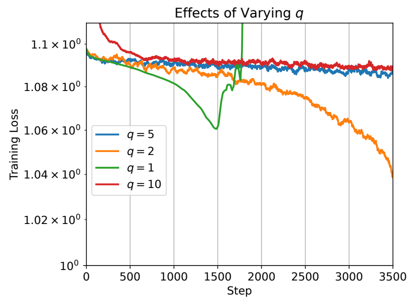


(a)

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 2: 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 3: a) Shows the effects of varying q on the convergence performance MeZO-SVRG. b) Shows the hyperparameter settings used in this experiment.

D.4 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 $\eta_1$, $\eta_2$, annealing factor $\alpha$, losses $L$, annealing threshold $\kappa$, 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
```

E Fine-tuning DistilBert

E.1 Hyperparameter Selection

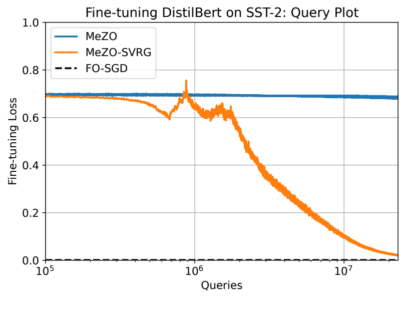
Table 9 shows the hyperparameter grid optimized over in the DistilBert (Sanh et al., 2020) experiment. The hyperparameter search was done by running the different algorithms for 1K steps on the MNLI (Williams et al., 2018) dataset and selecting the best configuration. The chosen configuration was then used for a longer fine-tuning runs for all considered tasks, i.e. 200K steps for MeZO and 50K steps for MeZO-SVRG.

Table 9: 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.

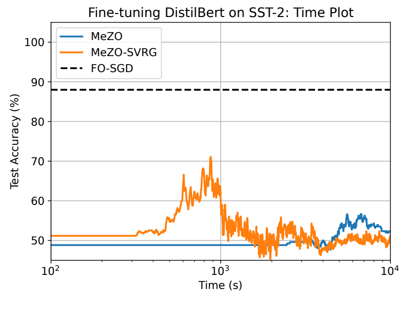
Algorithm	Hyperparameters	Values
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}4, 1e{-}5, \mathbf{1e{-}6}\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{200K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{\mathbf{1e{-}3}, 1e{-}4\}\times$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

E.2 Convergence Performance

We fine-tune Distilbert (Sanh et al., 2020) on the SST-2 (Socher et al., 2013) dataset. In Figure 4a, 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 4b 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 4: 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.

E.3 Additional Results

Table 10: 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 C for details).

Task	Method	Fine-tuning Loss ↓	Test Accuracy (%)↑	Queries ($\times 10^3$) ↓
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

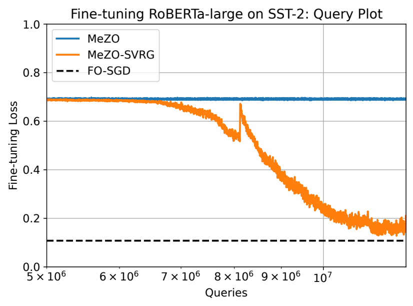
F Fine-tuning RoBERTa-large

F.1 Hyperparameter selection

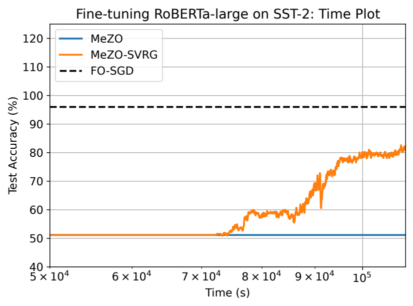
Table 11 presents the hyperparameters searched over in our RoBERTa-large (Liu et al., 2019) experiment. The hyperparameter search was done by fine-tuning the model on the MNLI (Williams et al., 2018) dataset for 1K steps and selecting the best configuration. This selected configuration was subsequently 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 11: 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	$\{\mathbf{1K}\}$



(a)



(b)

Figure 5: 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.

F.2 Convergence Performance

F.3 Additional Results

Table 12: Experiments on RoBERTa-large (with 512 fine-tuning examples). Here partial refers to fine-tuning the last layers of the model (see Appendix C 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 13: 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 C 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 ↓	Test Accuracy (%)↑	Queries ($\times 10^3$) ↓
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

G Additional Results for fine-tuning Autoregressive Models

G.1 Hyperparameter Selection

Table 14 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 14: 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.

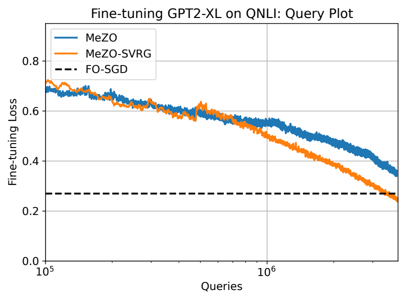
Algorithm	Hyperparameters	Values
MeZO	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate	$\{1e{-}6, \mathbf{5e{-}6}, 1e{-}7\}\times$
	μ	$\{\mathbf{1e{-}3}\}\times$
	Total Steps	$\{\mathbf{32K}\}$
MeZO-SVRG	Batch size	$\{32, \mathbf{64}\}\times$
	Learning rate (η_1)	$\{1e{-}4, \mathbf{5e{-}5}, 1e{-}5\}\times$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

G.2 Convergence Performance

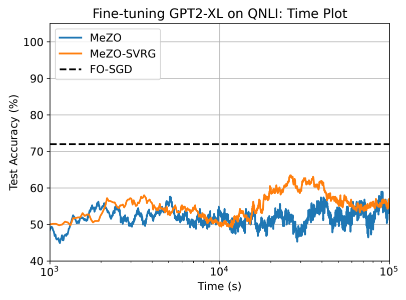
We fine-tune GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022) on the QNLI (Wang et al., 2018) dataset. In Figures 6a and 7a, we show the improved convergence performance of MeZO-SVRG over MeZO. For both models, MeZO-SVRG is able to significantly reduce the convergence gap compared to the FO-SGD baseline. Figures 6b and 7b 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.

G.3 Additional Results

Tables 15 and 16 present extended results on the fine-tuning tasks for GPT2 (Radford et al., 2019) and OPT-2.7B (Zhang et al., 2022).

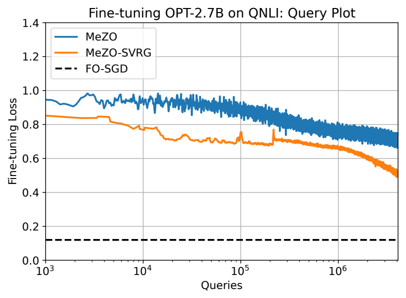


(a)

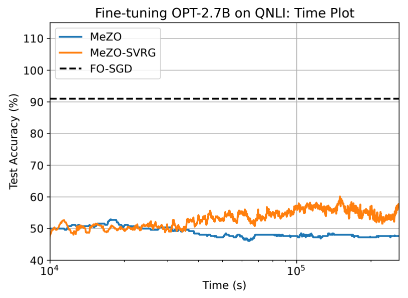


(b)

Figure 6: 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 7: 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 Memory Usage and Computation Time

H.1 Memory Profiling

We performed memory profiling experiments without any advanced memory-saving options such as lowering bit precision (Dettmers et al., 2022) 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. Table 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

Table 15: 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 16: 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 ↓	Test Accuracy (%)↑	Queries ($\times 10^3$) ↓
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

measurements were taken after running the methods for at least 100 steps. Table 3 summarizes the results.

We replicated all experiments in the half-precision (BF16) setting; the results are given in Table 24.

H.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 17 summarizes the results.

	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 17: Required GPU-hrs to achieve equivalent performance levels for MeZO-SVRG and MeZO.

I Half-Precision Experiments

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

I.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 18. 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 19.

Table 18: 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 19: 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 C 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

I.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 20. 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 21.

Table 20: 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 21: 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 C 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

I.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 22. 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 23. We include the BoolQ dataset from the SuperGLUE (Wang et al., 2019) benchmark to evaluate a more challenging fine-tuning task.

I.4 Memory Profiling with Half-Precision

We repeated memory profiling experiments under the half-precision (BF16) settings. The results are presented in Table 24.

Table 22: 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 23: 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 C 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

	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)	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 24: 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.