OPT2025: 17th Annual Workshop on Optimization for Machine Learning

Towards Robust Unroll Generalization in Learned Optimizers

Xiaolong Huang[1,3] HIROX827@GMAIL.COM
Benjamin Thérien[2,3] BENJTHERIEN@GMAIL.COM
Eugene Belilovsky[1,3] BELILOVSKY.EUGENE@GMAIL.COM
[1] *Concordia University*
[2] *Université de Montréal*
[3] *Mila - Quebec AI Institute*

Abstract

Recent works have demonstrated that learned optimizers (LOs) can be competitive and at times outperform hand-designed counterparts, paving a path towards improved optimizers by scaling up LOs. However, learned optimizers still require substantial meta-learning compute, which limits their scalability, requiring new methods that allow them to generalize to a wider array of problems from a smaller meta-learning problems. One aspect of this is the training horizon mismatch between meta-learning and real world training. We consider the problem of efficiently meta-learning LOs that can generalize to long training time horizons. We propose LoLO, which employs a replay buffer to efficiently extend unroll length during meta-training without increasing meta-learning cost. Furthermore, it incorporates on-policy imitation learning to ensure faithful trajectories and stabilize meta-training. We evaluate LoLO on a variety of vision and language tasks, demonstrating its success in achieving long unroll generalization in practical scenarios.

1. Introduction

The remarkable achievements of deep neural networks have largely been driven by scaling up training. Large-scale model training invariably relies on hand-designed gradient-based optimizers such as SGD[17], Adam[2, 6], or their variants[8]. Given the massive cost of these training runs, it is natural to search for more performance gradient-based optimizers. One approach is to learned the optimization algorithms themselves. Learned Optimizers (LOs)[1] offer the potential to automatically discover better update rules from data and, thereby, can accelerate training and achieve improved convergence. Despite being a promising paradigm, LOs are still in their early stages, with many challenges yet to be addressed. One of the most pressing problems for LO is meta-generalization as meta-learning new optimizers is expensive[13, 20]. A particularly pressing issue of meta-generalization is how to ensure that LOs maintain stable convergence when applied over very long unrolls, which is essential for effective downstream model training in practice.

In this work, we propose Long-horizon Resilient Learned Optimizer (LoLO) to address this challenge. Our method integrates a replay buffer[18], enabling LOs to experience longer unrolls during meta-training without incurring additional computational cost. This, in turn, equips LOs with stronger generalization capabilities on downstream tasks that require long unrolls. However, naively applying replay buffer mechanisms makes meta-training unstable in the early stages. To address this, we incorporate behavior cloning[16, 21] by having the LO imitate a hand-designed optimizer (in our case, Adam), ensuring constantly stable meta-training while still enabling the LO to improve.

Our main contributions can be summarized as follows:

- We introduce a replay buffer mechanism that enables LOs to observe sufficiently long unrolls during meta-training without adding extra training cost.
- We propose a behavior cloning strategy to stabilize meta-training and to transfer potential generalization benefits from an *expert* optimizer.
- We empirically demonstrate, across vision and language tasks, that LoLO consistently outperforms strong hand-designed and learned baselines.

2. Related Work

Learned Optimizers (LOs). LOs employ trainable models (e.g., MLPs) to replace hand-crafted optimization algorithms [1]. Previous literature has proposed a variety of approaches to improve LOs. [10, 12] explored new architectures and large-scale training regimes. [20] introduced techniques such as maximal update parameterization to leverage LOs in the training of large-scale models. Some researchers have also proposed to apply off-policy imitation learning to LOs to establish stronger baselines [3, 19]. In this paper, while we focus on enhancing the generalization ability of LOs under long unrolls, we also integrate on-policy imitation learning, but in a distinct way: we introduce a set of novel hybrid integration strategies that tightly couple imitation learning with replay buffer mechanisms. This joint design not only saves training resources but also strengthens the unroll generalization of LOs.

Replay Buffers. Replay buffer [18] originates from reinforcement learning (RL)[7, 24], where experience replay buffers have become a fundamental mechanism. Such buffers store trajectories or transitions collected during training, and allow the learner to sample from past experiences rather than relying solely on the most recent data. This approach not only stabilizes training but also improves sample efficiency and long-term credit assignment [23].

3. Method

3.1. Unroll Initialization from Replay Buffer for LOs

In existing meta-learning setups for learned optimizers, every unroll is randomly initialized[11, 20]. However, this approach is inefficient because, over time, the learned optimizer becomes adept at handling the initial training stages. We thus propose reusing checkpoints from previous unrolled trajectories to initialize new unrolls, rather than always starting from scratch. This can be achieved efficiently via replay buffers from RL, which enable meta-learning to access longer unrolls without substantially increasing the computational cost.

Specifically, we design our buffer $\mathcal{B}$ following the principles of a queue. At the initialization of the outer states, we first define $\mathcal{B}$'s capacity, i.e., the maximum number of inner state entries it can hold. In this work, we set the default size of $\mathcal{B}$ to be 4. Then we define $P_{th} \in [0, 1]$, a fixed threshold determines whether $\mathcal{B}$ should be used at a random outer step t. According to the Binomial distribution, the expected buffer usage is $\mathbb{E}[X] = T(1 - P_{th})$. Thus, a smaller P_{th} leads to more frequent buffer reuse, increasing the probability of encountering longer consecutive unrolls and yielding a larger expected effective unroll length. Conversely, a larger P_{th} reduces buffer usage, resulting in shorter average unrolls.

At the beginning of each unroll, we first randomly generate a probability $P_{\mathcal{B}} \sim \mathrm{Uniform}(0, 1)$. If $P_{\mathcal{B}} > P_{th}$ and $t > 0$, then the $\mathcal{B}$'s head element is used for inner initialization; otherwise, inner initialization falls back to a random state. Then we randomly sample a inner step index (between the start step and the end step of the unroll), marked as N_{push}. N_{push} determines at which step in the unroll the current state will be inserted into $\mathcal{B}$. When inner training reaches that step, the corresponding state is added to $\mathcal{B}$. If $\mathcal{B}$ is already full, the oldest entry at the head of the queue is removed before the new state is appended to the tail. Details descriptions of how replay buffer works are introduce in **Algorithm 1**.

3.2. Composition of Optimization Trajectories

Our goal is to expose the LO to sufficiently long unrolls across meta-training, which motivates setting a relatively small threshold P_{th}. However, generating long unrolls in the early stages of training can severely impair convergence. At this stage, the LO remains underfit and tends to produce low-quality optimization trajectories. Errors from these low-quality updates quickly accumulate across steps, leading to noisy and biased gradient signals. As the unroll length increases, this compounding effect becomes more severe, making the meta-objective unstable and in many cases causing meta-training to collapse altogether.

Inspired by imitation learning, we employ Adam as an *expert* in the inner loop to improve the quality of the trajectories. Concretely, for any inner step n, we construct an adaptive weighted sum between the trajectories produced by the LO and those produced by Adam, as expressed in the following formulation:

$$\tau_n = (1 - \alpha_t)\, \tau_{\mathcal{H}} + \alpha_t\, \tau_{\mathcal{O}}, \tag{1}$$

where $\alpha_t = \frac{t}{T-1} (t \in \mathbb{Z},\ 0 \leq t < T)$, with $\mathcal{H}$ and $\mathcal{O}$ denoting Adam and the LO, respectively. As the outer loop progresses, the quality of the trajectories generated by the LO are expected to gradually improve. Accordingly, the weighting gradually shifts from relying entirely on Adam to relying fully on the LO. This adaptive transition ensures that every inner-loop trajectory during meta-training remains of high quality, thereby avoiding redundant meta-training and instability that noisy trajectories would otherwise introduce.

3.3. Combining Meta-Learning and Supervises Loss

Directly relying on the fused trajectories makes the meta-loss (marked as $\mathcal{L}^{\text{task}}$) less representative of the LO's performance, causing meta-gradients to become noisy. This issue is particularly pronounced when α_t is large.

Given that Adam can produce high-quality trajectories, we can leverage a regularization loss $\mathcal{L}^{\text{bc}}$ to guide the LO at each n, replacing the noisy $\mathcal{L}^{\text{task}}$ with a more accurate signal. $\mathcal{L}^{\text{bc}}$ is defined as:

$$\mathcal{L}^{bc} = \sum_{n=1}^{N} \|\theta_n^{\mathcal{O}} - \theta_n^{\mathcal{H}}\|_2^2, \tag{2}$$

where $\theta_n^{\mathcal{O}}$ and $\theta_n^{\mathcal{H}}$ denote the parameters of the optimizee updated at unroll step $n-1$ by the LO and Adam, respectively. However, if the LO is trained solely under $\mathcal{L}^{\text{bc}}$, its performance will be inherently bounded by that of Adam. Our ultimate goal, however, is to train an LO capable of surpassing hand-designed optimizers on specific tasks. To this end, an advanced idea is to combine

$\mathcal{L}^{\text{bc}}$ with the $\mathcal{L}^{\text{task}}$ through a convex combination:

$$\mathcal{L}^{meta} = (1 - \alpha_t)\, \mathcal{L}^{\text{bc}} + \alpha_t\, \mathcal{L}^{\text{task}}, \tag{3}$$

where α_t is set the same as in **Eq.** 1. In this way, the meta-gradients are initially dominated by $\mathcal{L}^{\text{bc}}$, ensuring stable training signals in the early stage, and gradually shift towards being fully driven by $\mathcal{L}^{\text{task}}$, encouraging the LO to discover superior optimization rules.

Nevertheless, using such loss fusion implicitly assumes that the magnitudes of $\nabla_\phi \mathcal{L}^{bc}$ and $\nabla_\phi \mathcal{L}^{task}$ remain comparable. Empirically, the relative scales of these two terms vary considerably, with observed ratios frequently surpassing 10^3 and at times exceeding 10^4, which undermines the intended balance and makes the optimization very unstable.

To resolve this, we propose to align the magnitudes of $\nabla_\phi \mathcal{L}^{bc}$ and $\nabla_\phi \mathcal{L}^{task}$ by simply applying second moment normalization to both $\Delta_\epsilon \mathcal{L}_n^{\text{bc}}$ and $\Delta_\epsilon \mathcal{L}_n^{\text{task}}$ at each n, so that the contributions of $\mathcal{L}^{\text{bc}}$ and $\mathcal{L}^{\text{task}}$ are constantly governed by α_t. Concretely, we define $\Delta_\epsilon \mathcal{L}_n^{\text{meta}}$ as:

$$\Delta_\epsilon \mathcal{L}_n^{\text{meta}} = (1 - \alpha_t) \cdot \psi(\Delta_\epsilon \mathcal{L}_n^{\text{bc}}) + \alpha_t \cdot \psi(\Delta_\epsilon \mathcal{L}_n^{\text{task}}), \tag{4}$$

where the normalization operator is defined as:

$$\psi(x_n) = \frac{x_n}{\sqrt{\frac{1}{N}\sum_{n=1}^{N} x_n^2} + \epsilon_0}. \tag{5}$$

From Appendix **Eq.** 9, we know that when estimating $\nabla\phi\mathcal{L}_n^*$ using ES or its variants, $\Delta_\epsilon \mathcal{L}_n^*$ is linearly related to $\nabla\phi\mathcal{L}_n^*$. Consequently, $\psi(\Delta_\epsilon \mathcal{L}_n^*)$ is as effective as $\psi(\nabla\phi\mathcal{L}_n^*)$. This ensures that both loss terms contribute at comparable scales, allowing α_t to precisely modulate the influence of imitation and task-driven signals throughout meta-training. Detailed pipelines are in **Algorithm 1**.

4. Empirical Studies

In this section, we provide a rigorous validation and analysis of our proposed method. LoLO achieves state-of-the-art performance across various vision and natural language tasks, while also demonstrating way stronger long unroll generalization than naive LO. Furthermore, we conduct a series of ablation studies to disentangle the respective contributions of the replay buffer and behavior cloning components.

4.1. Experimental Details

Our experimental setup and meta-training pipelin largely follow that of [12, 20]. Specifically, both LoLO and the naive-LO baselines use 3-layer MLP with a hidden width of 32, and take a variety of input features inspired by [9]. Unlike naive-LO, LoLO integrates a replay buffer to extend unroll exposure, while leveraging Adam (lr=1×10^{-3}) as an *expert* to ensure stable training. For both methods, we applied best hyperparameter settings. Meta-training was conducted on the ImageNet-1k[4] (32×32) dataset for 5K outer steps with an unroll length of 1K steps. We estimate meta-gradients using persistent evolution strategies (PES) [22] with truncations of length 50.

For AdamW [8] baselines, we adopted a classic configuration with a learning rate of 1×10^{-3} and weight decay of 1×10^{-4}. For vision tasks, we used ImageNet-1k (32×32) as the benchmark dataset. We experimented with a 3-layer MLP (hidden width 128) and with ResNet-18 [5], using

Algorithm 1: Long-horizon Resilient Learned Optimizer (LoLO)

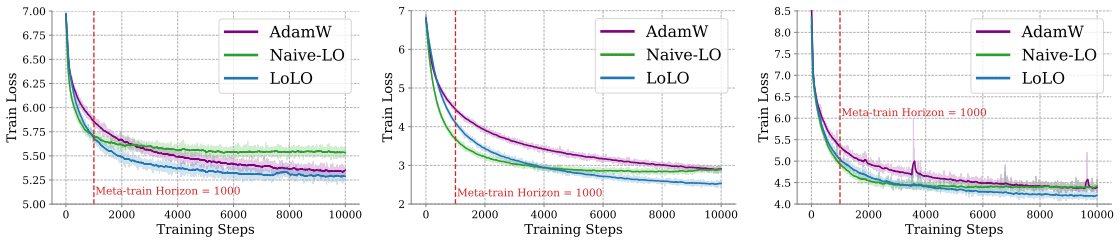
Input: The size M of replay buffer, the threshold P_{th} of applying buffer initialization.
Initialize: Replay buffer $\mathcal{B} = \{s_1, s_2, \ldots, s_m\}, m \leq M$, where s_* indicates a inner state.
Notations: The notations used are defined in Appendix A.

for $t = 0, 1, 2, ..., T-1$ **do**
  Sample $P_{\mathcal{B}} \sim \text{Uniform}(0, 1)$;
  if $(P_{\mathcal{B}} > P_{th}) \wedge (t > 0)$ **then**
    Select s_1 from $\mathcal{B}$ for inner initialization ($K := \pi_\zeta(s_1)$)
  else
    Randomly initialize inner state ($K = 0$)
  end
  sample $N_{\text{push}} \in [K, N+K) \cap \mathbb{Z}$;
  for $n = K, K+1, K+2, ..., K+N-1$ **do**
    $\theta^{\mathcal{H}}_{n+1} = \theta_n + \Delta\theta^{\mathcal{H}}_{n+1}$;
    $\theta^{\mathcal{O}}_{n+1} = \theta_n + \Delta\theta^{\mathcal{O}}_{n+1}$;
    $\theta_{n+1} = (1-\alpha_t)\theta^{\mathcal{H}}_{n+1} + \alpha_t\theta^{\mathcal{O}}_{n+1}$;
    $\Delta_\epsilon \mathcal{L}^{bc}_n = \mathcal{L}^{bc}_n(\phi + \epsilon) - \mathcal{L}^{bc}_n(\phi - \epsilon)$;
    $\Delta_\epsilon \mathcal{L}^{task}_n = \mathcal{L}^{task}_n(\phi + \epsilon) - \mathcal{L}^{task}_n(\phi - \epsilon)$;
    $s_{n+1} = (\theta_{n+1}, \zeta_{n+1})$;
    if $n == N_{push}$ **then**
      $\mathcal{B} \leftarrow \begin{cases} \text{enqueue}(\mathcal{B}, s_n), & m < M \\ \text{enqueue}(\text{dequeue}(\mathcal{B}), s_n), & m = M \end{cases}$;
    end
  end
  $g_t = \sum_{n=K+1}^{K+N} \textbf{PES}\Big((1-\alpha_t)\psi(\Delta_\epsilon \mathcal{L}^{bc}_n) + \alpha_t \psi(\Delta_\epsilon \mathcal{L}^{task}_n)\Big)$;
  $\phi_{t+1} = \mathcal{U}(g_t, t; \phi_t)$
end

a practical batch size of 4,096. For language tasks, we employed the FineWeb-10B [14] dataset along with the GPT-2-mini [15] model. Due to computational resource constraints, the batch size for language experiments was set to 512.

4.2. Comparison experiments.

We aim for LoLO to be applicable in practice rather than remaining confined to research settings. To this end, we evaluate its unroll generalization ability on widely adopted real-world tasks. Specifically, we take LOs that were meta-trained with unrolls of only 1K steps and assess them on 10K-step unroll training using multiple models across the ImageNet and FineWeb datasets. As shown in **Figure** 1, LoLO consistently achieves the lowest final train-loss compared to other methods across all tasks. Moreover, LoLO maintains convergence stability on longer unrolls (beyond 1K steps) comparable to AdamW, while naive-LO exhibits flattened curves, and even slightly diverges finally.



(*a*) Train Loss on In1K, MLP (*b*) Train Loss on In1K, ResNet18 (*c*) Train Loss on FineWeb10B, GPT2-mini

Figure 1: Comparing LoLO with other baselines on ImageNet-1K using 3-layer MLP and ResNet-18, and on FineWeb-10B using GPT-2-mini.

To further study generalization, we also track performance on test sets. As reported in Appendix **Figure** 3, the trends observed on test data closely mirror those on the training data.

4.3. Ablations.

We further aim to disentangle the individual contributions of LoLO's components. We conduct ablation studies on the replay buffer and behavior cloning, in order to establish their respective roles and investigate whether their combination yields synergistic effects. Specifically, we first evaluate the performance of Naive-LO, then verify the results of incorporating behavior cloning (since testing only with replay buffer may cause unstable convergence during meta-training), and finally we examine the complete LoLO method. As shown in Figure 2, the naive LO, by imitating Adam online, already achieves reasonable unroll generalization, while the replay buffer further strengthens this property.

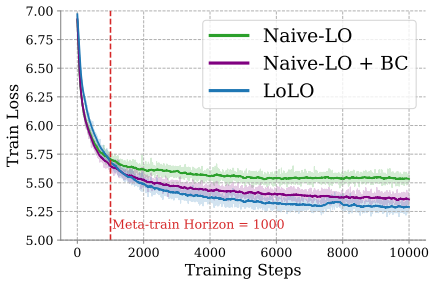


Figure 2: Ablation study on ImagNet1k using 3-layer MLP.

5. Conclusion.

In this work, we introduced LoLO, a memory-guided imitation framework for learned optimization. By coupling replay buffers with on-policy imitation, LoLO enables stable meta-training while exposing the optimizer to longer unrolls, thus improving its generalization to long unrolls in downstream tasks. Our experiments across vision and language benchmarks demonstrate that learned optimizers meta-learned with LoLO consistently outperform strong baselines, highlighting LoLO's potential as a practical and effective meta-training framework for improving learned optimizers' generalization to longer training horizons.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by

gradient descent. *Advances in neural information processing systems*, 29, 2016.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer, 2010.

[3] Tianlong Chen, Weiyi Zhang, Jingyang Zhou, Shiyu Chang, Sijia Liu, Lisa Amini, and Zhangyang Wang. Training stronger baselines for learning to optimize. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/51f4efbfb3e18f4ea053c4d3d282c4e2-Abstract.html.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. In *2018 56th annual allerton conference on communication, control, and computing (Allerton)*, pages 478–485. IEEE, 2018.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[9] Niru Maheswaranathan, David Sussillo, Luke Metz, Ruoxi Sun, and Jascha Sohl-Dickstein. Reverse engineering learned optimizers reveals known and novel mechanisms. *Advances in neural information processing systems*, 34:19910–19922, 2021.

[10] Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. Velo: Training versatile learned optimizers by scaling up, 2022. *URL https://arxiv. org/abs/2211.09760*.

[11] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565. PMLR, 2019.

[12] Luke Metz, C Daniel Freeman, James Harrison, Niru Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022. URL http://github.com/google/learned_optimization.

[13] Abhinav Moudgil, Boris Knyazev, Guillaume Lajoie, and Eugene Belilovsky. Celo: Training versatile learned optimizers on a compute diet.

[14] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[16] Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. *Advances in Neural Information Processing Systems*, 33:2914–2924, 2020.

[17] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[18] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[19] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[20] Benjamin Thérien, Charles Étienne Joseph, Boris Knyazev, Edouard Oyallon, Irina Rish, and Eugene Belilovsky. μlo: Compute-efficient meta-generalization of learned optimizers, 2024. URL https://arxiv.org/abs/2406.00153.

[21] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[22] Paul Vicol, Luke Metz, and Jascha Sohl-Dickstein. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10553–10563. PMLR, 2021.

[23] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando De Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[24] Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

Appendix A. Notation for Algorithm 1

$\mathcal{H}$: Indicating hand-designed optimizer, here is Adam
$\mathcal{O}$: Indicating LO
θ : Parameters of the optimizee
ϕ : Parameters of the optimizer (LO)
$P_{\mathcal{B}}$: Probability of selecting a state from $\mathcal{B}$ for initialization
K : Start step of each inner loop
N_{push} : Inner step index at which the state is pushed into $\mathcal{B}$
ζ_n : Auxiliary accumulators (e.g., momentum, step count)
$\pi_\zeta(s)$: Projection operator extracting the step index from state s
ψ : Second moment normalization
$\mathcal{U}$: Generic update operator

Appendix B. Background

Behavior Cloning. Behavior Cloning (BC) is one of the most widely used paradigms in offline imitation learning, where the objective is to approximate an expert policy by directly regressing from observed states to expert actions. Formally, given a dataset of expert demonstrations $\mathcal{D} = \{(x_i, a_i^\star)\}_{i=1}^N$, where $x_i \in \mathcal{X}$ denotes the state and $a_i^\star \in \mathcal{A}$ is the corresponding expert action, the goal is to learn a policy $\pi_\theta : \mathcal{X} \to \mathcal{A}$, parameterized by θ, that closely approximates the expert's behavior. A simple version of behavior cloning solves the supervised regression problem:

$$\hat{\pi}_\theta = \arg\min \pi_\theta \, \frac{1}{N} \sum_{i=1}^{N} \|\pi_\theta(x_i) - a_i^\star\|_2^2 \,. \tag{6}$$

Learned Optimization. In this work we adopt the `small_fc_lopt` architecture of [12], a three-layer MLP with ReLU activations. The optimizer takes as input a feature vector ($\boldsymbol{u}$) for each parameter in the optimizee and outputs an update direction, d, and magnitude, m. That is, $f\phi(\cdot) = [d, m]$, where f is the learned optimizer and ϕ are its parameters. The optimizee's parameters (θ) are then updated as follows:

$$\theta_t = \theta_{t-1} - \lambda_1 d e^{\lambda_2 m}. \tag{7}$$

In general, learning the meta-parameters, ϕ, involves solving an optimization problem of the form [20]:

$$\min_{\phi} \; \mathbb{E}_{(\mathcal{D},\mathcal{L},\boldsymbol{w}_0)\sim\mathcal{T}} \left[\mathbb{E}_{(X,Y)\sim\mathcal{D}} \left[\frac{1}{T} \sum_{t=0}^{T-1} \mathcal{L}(X, Y; f_\phi(\boldsymbol{u}_t), \boldsymbol{w}_t) \right] \right]. \tag{8}$$

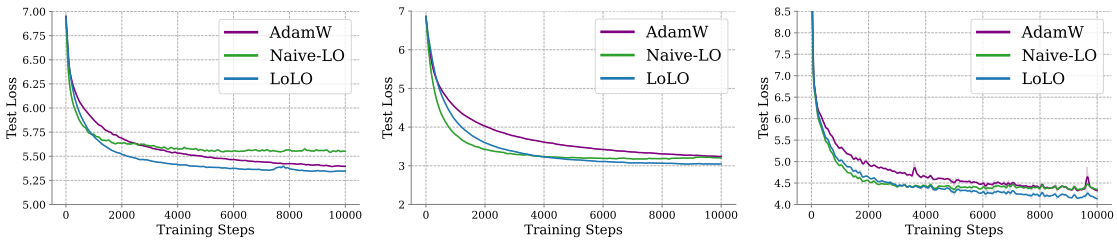
Here, $\mathcal{T}$ is a distribution of tasks defined as a distribution, an objective function, and an initialization triple. The objective seeks to minimize the sum of per-timestep losses over the training horizon T.

Persistent evolution strategies estimator Evolution strategies (ES) estimators are helpful gradient estimators for unrolled optimization problems as they do not explode or vanish as T is increased [11]. A standard antithetic ES gradient can be computed as follows:

$$\hat{\mathbf{g}}^{\text{ES-A}} = \frac{1}{N\sigma^2} \sum_{i=1}^{N/2} \epsilon^{(i)} \Big(\mathcal{L}(\phi + \epsilon^{(i)}) - \mathcal{L}(\phi - \epsilon^{(i)}) \Big). \tag{9}$$

Where $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ is a perturbation and N is the number of antithetic perturbations sampled. When the meta-loss is evaluated after T steps, the estimator is unbiased with respect to our objective in equation 8. However, this can be computationally expensive when T is large. Alternatively, we can consider updating our meta-parameters, ϕ, at an intermediate point during the full unroll. This biased algorithm, known as Truncated ES, follows equation 9 but evaluates the loss every k steps, where k is the truncation length. Persistent evolution strategies [22] is an unbiased alternative to ES, which we use to estimate the meta-gradients in our work.

Appendix C. Test results on vision and language tasks



(*a*) Test Loss on In1K, MLP (*b*) Test Loss on In1K, ResNet18 (*c*) Test Loss on FineWeb10B, GPT2-mini

Figure 3: Comparing LoLO with other baselines on ImageNet-1K using 3-layer MLP and ResNet-18, and on FineWeb-10B using GPT-2-mini.