OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Information-Theoretic Trust Regions for Stochastic Gradient-Based Optimization

Philipp Dahlinger[1] PHILIPP.DAHLINGER@KIT.EDU
Philipp Becker[1,2]
Maximilian Hüttenrauch[1]
Gerhard Neumann[1,2]
1: Karlsruhe Institute of Technology, Germany
2: FZI Research Center for Information Technology, Germany

Abstract

Stochastic gradient-based optimization is crucial to optimize neural networks. While popular approaches heuristically adapt the step size and direction by rescaling gradients, a more principled approach to improve optimizers requires second-order information. Such methods precondition the gradient using the objective's Hessian. Yet, computing the Hessian is usually expensive and effectively using second-order information in the stochastic gradient setting is non-trivial. We propose using Information-Theoretic Trust Region Optimization (arTuRO) for improved updates with uncertain second-order information. By modeling the network parameters as a Gaussian distribution and using a Kullback-Leibler divergence-based trust region, our approach takes bounded steps accounting for the objective's curvature and uncertainty in the parameters. Before each update, it solves the trust region problem for an optimal step size, resulting in a more stable and faster optimization process. We approximate the diagonal elements of the Hessian from stochastic gradients using a simple recursive least squares approach, constructing a model of the expected Hessian over time using only first-order information. We show that arTuRO combines the fast convergence of adaptive moment-based optimization with the generalization capabilities of SGD.

1. Introduction

In this work, we introduce Inform**a**tion-Theo**r**etic **Tru**st **R**egion **O**ptimization (arTuRO), an approach using approximate second-order information for stochastic optimization[1]. While using second-order information for preconditioning and step size selection is ubiquitous in classical optimization literature [8], it is not broadly adopted for stochastic optimization in deep learning as it suffers from two main problems. First, controlling the step size, using, e.g., a line search, breaks down if the second-order information is only approximate [7, 17]. Second, while modern automatic differentiation tools allow computing the Hessian, this is often impractical due to high memory and processing demands.

arTuRO addresses the uncertainty in the stochastic second-order information by limiting the maximal update step using trust regions. We take a distributional view of the network parameters and formalize a constrained optimization problem with an information-theoretic trust region [1, 20] that considers the objective's curvature. Furthermore, we estimate the Hessian's diagonal using gradient

1. Code available at https://github.com/ALRhub/arturo.

information to avoid computing it explicitly. Using only a single gradient per step, we iteratively approximate the Hessian using recursive least squares and use a drift model [23] to account for the parameters changing during updates. Assuming a Gaussian random walk of the Hessian, older gradients have a smaller weight in the regression.

We show that arTuRO profits from both the trust regions and its improved way of estimating the Hessian on several standard benchmark tasks and model architectures where it exhibits similar or better performance compared to optimizers with moment-based adaptations and SGD.

2. Information-Theoretic Trust Region Optimization

We study a typical supervised learning task where the aim is to solve an unconstrained non-convex optimization problem of the form

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} l_i(\boldsymbol{\theta}; x_i, y_i). \tag{1}$$

Here, $\boldsymbol{\theta} \in \mathbb{R}^n$ are the parameters of a neural network we aim to optimize, (x_i, y_i) are pairs of input data and output targets while l_i defines a differentiable loss function. The objective is usually approximated using mini-batches instead of the entire dataset.

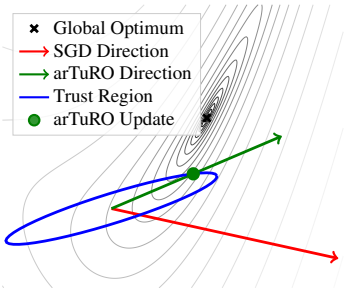

Figure 1: Visualization of the main idea behind arTuRO - Information-Theoretic Trust Regions. arTuRO's update direction (green arrow) is clearly better than simply stepping in the gradient direction (red). Yet, due to the stochastic nature of the optimization, fully relying on second-order information can be suboptimal. Thus, we use a trust region (blue) based on the current parameter uncertainty to restrict the update length.

Trust Region Optimization using Second-Order Information. We define a distribution over the parameters in order to introduce an information-theoretic trust region. A natural choice is a Gaussian distribution $\pi_t(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, with mean $\boldsymbol{\mu}_t$ and covariance $\boldsymbol{\Sigma}_t$. The probabilistic view allows us to define a principled trust region. Given a bound $\varepsilon \geq 0$, we bound the change in parameter distribution by the Kullback-Leibler (KL) divergence $\mathrm{KL}(\pi_t || \pi_{t-1}) \leq \varepsilon$. We can now solve Eq. (1) by repeatedly minimizing the expectation of the loss function over the distribution π_t under the trust-region constraint

$$\min_{\pi_t} \mathbb{E}_{\boldsymbol{\theta} \sim \pi_t}[\mathcal{L}(\boldsymbol{\theta})], \quad \text{s.t. } \mathrm{KL}(\pi_t || \pi_{t-1}) \leq \varepsilon.$$

The primal solution of this problem has a closed-form solution for quadratic functions $\mathcal{L}(\boldsymbol{\theta})$[1]. Thus, we use a second-order Taylor expansion of the loss function around the current mean $\boldsymbol{\mu}_t$ of the parameter distribution $\mathcal{L}(\boldsymbol{\theta}) \approx f_t(\boldsymbol{\theta}) = 0.5\boldsymbol{\theta}^T \boldsymbol{A}_t \boldsymbol{\theta} + \boldsymbol{\theta}^T \boldsymbol{b}_t + c_t$. Since the Hessian $\boldsymbol{H}_{\boldsymbol{\mu}_t}$ and the gradient $\nabla\mathcal{L}(\boldsymbol{\mu}_t)$ of the full dataset is unknown due to mini-batching, we have a further approximation of the Taylor coefficients $\boldsymbol{A}_t \approx \boldsymbol{H}_{\boldsymbol{\mu}_t}$ and $\boldsymbol{b}_t \approx \nabla\mathcal{L}_t(\boldsymbol{\mu}_t) - \boldsymbol{H}_{\boldsymbol{\mu}_t}\boldsymbol{\mu}_t$. This formulation allows for closed-form solutions of the expectation in the optimization problem as the distribution over parameters $\boldsymbol{\theta}$ is Gaussian.

Disentangled Trust Regions and Entropy Regularization. Previous work [2, 13] shows that disentangling the KL in independent constraints for the change of the mean and covariance improves the optimization procedure. The KL divergence for Gaussian distributions has a closed-form solution $\mathrm{KL}(\pi_t || \pi_{t-1}) = C_\mu + C_\Sigma$ with

$$C_\mu = 0.5(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1})^T \boldsymbol{\Sigma}_{t-1}^{-1} (\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}) \quad \text{and}$$
$$C_\Sigma = 0.5\big[\operatorname{tr}\big(\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\Sigma}_t\big) - n + \log\det(\boldsymbol{\Sigma}_{t-1}) - \log\det(\boldsymbol{\Sigma}_t)\big].$$

As we are ultimately interested in a maximum a posteriori (MAP) solution, i.e., the mean of the parameter distribution, we propose to use the covariance part of the KL divergence in the arTuRO objective instead of using it as a constraint. This regularizes the optimal updated covariance towards the old covariance $\boldsymbol{\Sigma}_{t-1}$. We further introduce a KL divergence $C_\lambda = \mathrm{KL}(\pi_t || \mathcal{N}(0, \lambda^{-1}\boldsymbol{I}))$ to a prior distribution and add it to the objective which regularizes the entropy of the distribution. The resulting objective of the arTuRO optimization problem is given as

$$\min_{\pi_t} \mathbb{E}_{\boldsymbol{\theta}\sim\pi_t}[f_t(\boldsymbol{\theta})] + \rho C_\lambda + \nu C_\Sigma \quad \text{s.t. } C_\mu \leq \varepsilon. \tag{2}$$

The factors $\rho, \nu \in \mathbb{R}$ weight the regularization, while the precision $\lambda \in \mathbb{R}$ defines the prior's scaling.

Solving the Trust Region Problem. We use the method of Lagrange multipliers to find a solution to Eq. (2). Taking the derivative of the Lagrangian with respect to the primal parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ and setting it to 0, we obtain the primal solutions

$$\boldsymbol{\mu}_t(\eta) = (\boldsymbol{A}_t + \eta\boldsymbol{\Sigma}_{t-1}^{-1} + \rho\lambda\boldsymbol{I}_n)^{-1}(\eta\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} - \boldsymbol{b}_t), \tag{3}$$

$$\boldsymbol{\Sigma}_t = (\rho+\nu)(\boldsymbol{A}_t + \rho\lambda\boldsymbol{I}_n + \nu\boldsymbol{\Sigma}_{t-1}^{-1})^{-1} \tag{4}$$

where $\eta \geq 0$ is a Lagrange multiplier. Thus, we see how η interpolates between fully trusting and discarding the second-order information. To compute the optimal value η^* we need to maximize the dual objective $g(\eta)$ corresponding to Eq. (2). For a detailed derivation, we refer to Appendix A. In order to scale to high dimensional problems we use a diagonal parameterization of the covariance $\boldsymbol{\Sigma}_t = \mathrm{diag}(\boldsymbol{\sigma}_t^2)$ and the matrix $\boldsymbol{A}_t = \mathrm{diag}(\boldsymbol{a}_t)$.

It remains to define the update of the surrogate model. The general idea is to accumulate the previous gradient information of the loss function into the current update using recursive least squares with a drift model. The solution is a Kalman update which is scalable to high-dimensional problems. In Appendix B we describe the derivation and the update equations of $\boldsymbol{A}_t$ and $\boldsymbol{b}_t$.

A crucial part of arTuRO is computing the Lagrange multiplier $\eta \geq 0$, which gives the step size of the update. Here, we use a bisection method due to its simplicity and amenability to a GPU implementation. As the change in η between iterations is small, we can initialize the bisection using tight bounds around the previous η. Using this method, we usually only need 2 to 4 iterations to find a sufficiently precise η. For hyperparameters and further details, see Appendix C.

3. Experiments

We test the performance of arTuRO on the well-established benchmarks Fashion-MNIST, CIFAR-10, and CIFAR-100. As the network architecture greatly affects the optimization procedure, we use a classical convolutional neural network (CNN), as well as ResNet architectures to evaluate the algorithms over different-sized networks. For architecture details, see Appendix C. We repeat each experiment over 10 seeds. In the Fashion-MNIST CNN experiment, we train the network for 120 epochs, while the other experiments run for 160 epochs.

Results. We compare arTuRO against the state-of-the-art optimization algorithms SGD with momentum, Adam [15], and AdamW [16]. All algorithms including arTuRO use weight decay. We also include a learning rate scheduler for SGD, Adam, and AdamW, while in arTuRO, we schedule

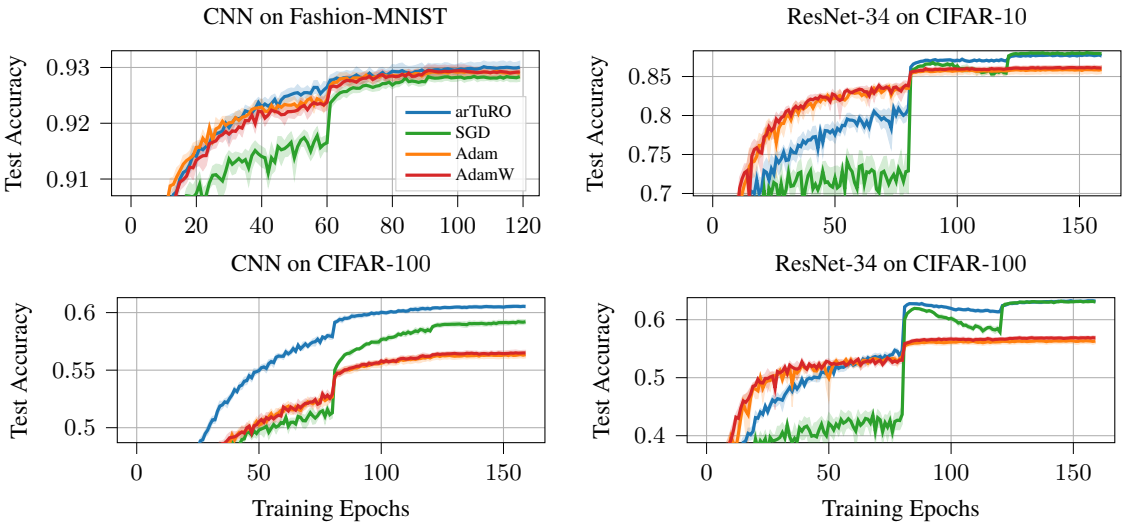

Figure 2: Test accuracies of arTuRO, SGD, Adam and AdamW.

	Fashion-MNIST		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
SGD	92.84 ± 0.08	$\mathbf{92.64 \pm 0.07}$	87.11 ± 0.15	$\mathbf{87.92 \pm 0.16}$	59.20 ± 0.18	$\mathbf{63.05 \pm 0.16}$
Adam	92.90 ± 0.10	91.76 ± 0.17	85.86 ± 0.15	85.89 ± 0.51	56.35 ± 0.27	56.32 ± 0.45
AdamW	92.92 ± 0.09	91.97 ± 0.11	85.88 ± 0.12	86.07 ± 0.30	56.50 ± 0.22	56.86 ± 0.28
arTuRO	$\mathbf{93.01 \pm 0.10}$	92.13 ± 0.12	$\mathbf{87.37 \pm 0.13}$	87.66 ± 0.19	$\mathbf{60.53 \pm 0.13}$	$\mathbf{63.19 \pm 0.24}$

Table 1: Results of a CNN and a ResNet architecture on Fashion-MNIST, CIFAR-10, and CIFAR-100. We report the mean and the doubled standard error of the test accuracy in [%].

the trust region bound ε. Twice during the optimization, the effective step size is reduced by a factor[2] of 0.1. For a fair comparison, we conduct an extensive hyperparameter optimization for each objective and algorithm. In Table 1, we report the resulting final accuracy on previously unseen test data. arTuRO performs superior on all CNN architectures and is comparable to SGD and clearly outperforms Adam and AdamW on the ResNet tasks. Furthermore, SGD outperforms the adaptive learning rate methods Adam and AdamW in every task besides Fashion-MNIST CNN. Appendix C lists the used hyperparameters for each experiment and algorithm. In Fig. 2, we present the accuracy on the test set throughout the optimization for four of the six experiments. arTuRO combines the faster convergence of Adam and AdamW with the generalization potential of SGD and benefits greatly from learning rate decay steps similar to SGD. Adam and AdamW can only compete on the CNN architecture on Fashion-MNIST (top-left). SGD on ResNet-34 shows a more unstable behavior as the test accuracy drops during a period of the optimization procedure. While SGD achieves a slightly higher final test accuracy, it is harder to tune. The remaining two figures, as well as the train loss curves, are given in Appendix D.

2. Since the trust region bound of arTuRO scales quadratically, we use a different scaling factor of 0.006. Empirically, this results in the same reduction of the step size.

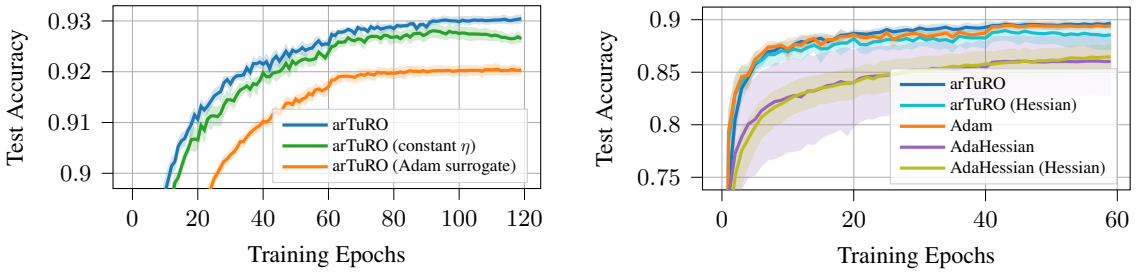

Figure 3: Ablation Results. **Left:** arTuRO ablation of the trust region and the surrogate computation. We report the test accuracy of the Fashion MNIST CNN task over the elapsed epochs. The blue line is the standard arTuRO algorithm, the green line uses a constant η instead of the solution to the dual problem, and the orange line replaces the recursive least squares surrogate computation with Adam's first and second moments estimation. **Right:** Ablation on Fashion MNIST with a smaller network where the computation of the analytic diagonal Hessian is feasible. *arTuRO(Hessian)* uses a running average over the Hessian and the gradient instead of our squared surrogate. *AdaHessian (Hessian)* uses the AdaHessian algorithm with the exact Hessian computation.

Ablations. First, we investigate the step size selection while using the identical update direction of the arTuRO algorithm. We compare the trust region approach with an ablation where the dual parameter η remains fixed. We carefully selected the fixed η with insights from the trust region approach as a general choice of η is not obvious. Additionally, we use a running average of the moment estimates from the Adam algorithm as the parameters of the quadratic surrogate to show that while the trust region approach is highly effective, it depends on an accurate model. We present the results on the left side in Fig. 3. We can conclude that a good surrogate is a key property for second-order algorithms. When having an adequate surrogate, the information-theoretic trust region gives an extra performance increase.

Second, we compare the second-order estimation of arTuRO on a smaller network architecture against other second-order estimation techniques. To this end, we compute the exact (stochastic) diagonal of the Hessian with a variation of the backpropagation algorithm [10]. The running average of the Hessian and the gradient is then used as the surrogate parameters. We compare against Adam and the diagonal Hessian approximation of AdaHessian [28]. To further evaluate AdaHessian approximation to the exact diagonal of the Hessian, we replace the AdaHessian approximation with the exact Hessian without altering the rest of the AdaHessian algorithm. The results of the ablation displayed in Fig. 3 (right side) indicate that correct Hessian information based on mini-batches in the stochastic setting does not result in better optimization results and we can outperform other second-order methods with our fit of the quadratic surrogate.

4. Related Work

First-Order Methods. First-order optimization methods like Stochastic Gradient Descent (SGD) [22] are widely used in deep learning for their simplicity and effectiveness. To improve performance, techniques like momentum [18, 25] and adaptive learning rates [11, 29] are have been introduced. The Adamalgorithm [15] is a popular variant that employs both of them. Still, there are challenges, such as choosing the correct learning rate, addressing convergence issues [21], and correctly including regularization [16]. While variants of Adam, such as AMSGRAD [21] and

AdamW [16] try to address these issues, there are remaining issues with generalization [14, 27], convergence [26] and the theoretical understanding [30] of Adam and its variants.

Second-Order Methods. Using second-order information to precondition the gradients and automatically setting the learning rate has many practical and theoretical benefits in classical optimization [3, 5]. However, classical Hessian approximation schemes[9] are not straightforwardly applicable in the stochastic optimization setting. While many approaches [17, 28] propose alternatives to approximate and use Hessians in this setting, devising a generally applicable, efficient second-order optimizer for large-scale stochastic optimization remains an open research question.

Information Theoretic Trust Regions. Information-theoretic trust regions find widespread use in reinforcement learning (RL)[19, 24]. The crucial difference is that arTuRO proposes a KL-bound in parameter space while the previous methods limit the KL-divergence in the output space. Inspired by zero-order stochastic-search algorithms [1, 12, 20], Arenz et al. [4] propose a first-order method to directly optimize model parameters, but their approach does not gracefully scale to even small neural networks with thousands of parameters and even less so to larger ones.

5. Conclusion

We introduced Information-Theoretic Trust Region Optimization (arTuRO), a novel approach to stochastic optimization of deep neural networks. First, arTuRO efficiently estimates second-order information from gradients using a recursive least squares approach. In the parameter update, arTuRO accounts for uncertainty in the second-order information and current parameters by limiting the updates using information-theoretic trust regions. arTuRO matches the convergence behavior of approaches using gradient statistics for preconditioning and step size control, e.g., Adam [15], while giving results comparable to those of a carefully tuned SGD with momentum. In future work, we aim to use our approach for Bayesian Deep Learning. There are similarities between our objective and the variational evidence lower bound which we could further exploit and learn a full distribution over network parameters.

6. Acknowledgments

This work was supported by funding from the pilot program Core Informatics of the Helmholtz Association (HGF). The authors acknowledge support by the state of Baden-Württemberg through bwHPC, as well as the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the German Federal Ministry of Education and Research.

References

[1] Abbas Abdolmaleki, Rudolf Lioutikov, Jan R Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[2] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[3] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *J. Mach. Learn. Res.*, 18:116:1–116:40, 2017. URL http://jmlr.org/papers/v18/16-491.html.

[4] Oleg Arenz, Philipp Dahlinger, Zihan Ye, Michael Volpp, and Gerhard Neumann. A unified perspective on natural gradient variational inference with gaussian mixture models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=tLBjsX4tjs.

[5] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 04 2018. ISSN 0272-4979. doi: 10.1093/imanum/dry009. URL https://doi.org/10.1093/imanum/dry009.

[6] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching l-bfgs method for machine learning. In *International Conference on Machine Learning*, pages 620–629. PMLR, 2018.

[7] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[8] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3. doi: 10.1017/CBO9780511804441. URL https://web.stanford.edu/%7Eboyd/cvxbook/.

[9] Richardh Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 02 2003. doi: 10.1137/0916069.

[10] Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlrF24twB.

[11] Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, pages 26–31, 2012. URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[12] Maximilian Hüttenrauch and Gerhard Neumann. Regret-aware black-box optimization with natural gradients, trust-regions and entropy control. *arXiv preprint arXiv:2206.06090*, 2022.

[13] Maximilian Hüttenrauch and Gerhard Neumann. Regret-aware black-box optimization with natural gradients, trust-regions and entropy control. *arXiv preprint arXiv:2206.06090*, 2022.

[14] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to SGD. *CoRR*, abs/1712.07628, 2017. URL http://arxiv.org/abs/1712.07628.

[15] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–15, 2015.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR*, 2019.

[17] Aryan Mokhtari and Alejandro Ribeiro. Res: Regularized stochastic bfgs algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, 2014.

[18] Juri Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983. URL https://ci.nii.ac.jp/naid/10029946121/en/.

[19] Fabian Otto, Philipp Becker, Vien Anh Ngo, Hanna Carolin Maria Ziesche, and Gerhard Neumann. Differentiable trust region layers for deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qYZD-AO1Vn.

[20] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[21] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

[22] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 00034851. URL http://www.jstor.org/stable/2236626.

[23] Simo Särkkä. *Bayesian Filtering and Smoothing*, volume 3 of *Institute of Mathematical Statistics textbooks*. Cambridge University Press, 2013. ISBN 978-1-10-761928-9.

[24] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897, 2015.

[25] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

[26] Tao Tan, Shiqun Yin, Kunling Liu, and Man Wan. On the convergence speed of amsgrad and beyond. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 464–470, 2019. doi: 10.1109/ICTAI.2019.00071.

[27] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

[28] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10665–10673, 2021.

[29] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

[30] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21285–21296, 2020.

Appendix A. Detailed solution of the Optimization problem

This section presents the detailed solution to the arTuRO optimization problem

$$\begin{aligned} \min_{\pi_t} \quad & \mathbb{E}_{\boldsymbol{\theta}\sim\pi_t}[f_t(\boldsymbol{\theta})] + \rho C_\lambda + \nu C_\Sigma \\ \text{s.t.} \quad & C_\mu \leq \varepsilon \end{aligned} \tag{5}$$

with

$$\begin{aligned} \pi_t(\boldsymbol{\theta}) &= \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \\ f_t(\boldsymbol{\theta}) &= \frac{1}{2}\boldsymbol{\theta}_t^T \boldsymbol{A}_t \boldsymbol{\theta}_t + \boldsymbol{\theta}^T \boldsymbol{b}_t + c_t, \\ C_\lambda &= \mathrm{KL}(\pi_t || \mathcal{N}(0, \lambda^{-1}\boldsymbol{I}_n) \\ C_\Sigma &= \frac{1}{2}\Big[\operatorname{tr}\big(\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\Sigma}_t\big) - n + \log\det(\boldsymbol{\Sigma}_{t-1}) - \log\det(\boldsymbol{\Sigma}_t)\Big] \\ C_\mu &= \frac{1}{2}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1})^T \boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}). \end{aligned}$$

The variables $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ are the primal parameters. We define a dual parameter $\eta \in \mathbb{R}$. Inserting the diagonal forms $\boldsymbol{\Sigma}_t = \operatorname{diag}(\boldsymbol{\sigma}_t^2)$ and $\boldsymbol{A}_t = \operatorname{diag}(\boldsymbol{a}_t)$, and solving the expectation over the quadratic surrogate, the Lagrangian is given by

$$\begin{aligned} L(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2, \eta) = &\frac{1}{2}\sum_{j=1}^{n}\Big[\mu_{t,j}^2 a_{t,j} + a_{t,j}\sigma_{t,j}^2 + \mu_{t,j}b_{t,j} + c_t\Big] \\ &+\rho\frac{1}{2}\sum_{j=1}^{n}\Big[\sigma_{t,j}^2\lambda - 1 + \mu_{t,j}^2\lambda + \log\big(\lambda^{-1}\big) - \log\big(\sigma_{t,j}^2\big)\Big] \\ &+\nu\frac{1}{2}\sum_{j=1}^{n}\Big[\frac{\sigma_{t,j}^2}{\sigma_{t-1,j}^2} - 1 + \log\big(\sigma_{t-1,j}^2\big) - \log\big(\sigma_{t,j}^2\big)\Big] \\ &+\eta\frac{1}{2}\sum_{j=1}^{n}\frac{(\mu_{t,j} - \mu_{t-1,j})^2}{\sigma_{t-1,j}^2} - \eta\varepsilon. \end{aligned}$$

Here, we sum over the parameter dimensions $j = 1, \dots, n$. Next, we compute the derivative with respect to the mean and set it to zero

$$\frac{\partial L}{\partial \mu_{t,j}} = \mu_{t,j} a_{t,j} + b_{t,j} + \rho\lambda\mu_{t,j} + \eta \frac{\mu_{t,j} - \mu_{t-1,j}}{\sigma^2_{t-1,j}} \stackrel{!}{=} 0.$$

This results in the primal solution of the mean

$$\boldsymbol{\mu}_t(\eta) = (\boldsymbol{A}_t + \eta\boldsymbol{\Sigma}_{t-1}^{-1} + \rho\lambda\boldsymbol{I}_n)^{-1}(\eta\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} - \boldsymbol{b}_t).$$

In the same way, we compute the primal solution of the variance

$$\begin{aligned} \frac{\partial L}{\partial \sigma^2_{t,j}} &= \frac{1}{2}\Big[a_{t,j} + \rho\lambda + \frac{\rho}{\sigma^2_{t,j}} + \frac{\nu}{\sigma^2_{t-1,j}} - \frac{\nu}{\sigma^2_{t,j}}\Big] \stackrel{!}{=} 0 \\ \implies \quad \boldsymbol{\Sigma}_t &= (\rho + \nu)\Big(\boldsymbol{A}_t + \rho\lambda\boldsymbol{I}_n + \nu\boldsymbol{\Sigma}_{t-1}^{-1}\Big)^{-1}. \end{aligned}$$

Note, that $\boldsymbol{\Sigma}_t$ is independent of the Lagrange multiplier η since we have dropped the covariance part of the KL divergence from the constraints.

It remains to solve the dual optimization problem to obtain η. The dual function is given by inserting the primal solution into the Lagrangian, resulting in

$$g(\eta) = L(\boldsymbol{\mu}_t(\eta), \boldsymbol{\Sigma}_t(\eta), \eta).$$

We solve the convex dual optimization problem

$$\eta^* = \text{argmax}_\eta \;\; g(\eta), \quad \text{s.t. } \eta \geq 0 \tag{6}$$

by finding an η^* with the derivative

$$g'(\eta^*) = 0.$$

The derivative of the dual has a simple form given by

$$g'(\eta) = \frac{1}{2}(\boldsymbol{\mu}_t(\eta) - \boldsymbol{\mu}_{t-1})^T\boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t(\eta) - \boldsymbol{\mu}_{t-1}) - \varepsilon. \tag{7}$$

We find η^* using a bisection method. It is possible that the derivative in Eq. (7) is always negative. This happens when the optimal solution lies inside the trust region. In that case, the solution to Eq. (6) is given by $\eta^* = 0$.

Appendix B. Derivation of the Surrogate Model Fitting

This section describes the computation of the quadratic surrogate

$$\mathcal{L}(\boldsymbol{\theta}) \approx f_t(\boldsymbol{\theta}) = 0.5\boldsymbol{\theta}^T\boldsymbol{A}_t\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{b}_t + c_t \tag{8}$$

from previous gradient evaluations $\boldsymbol{g}_0 = \nabla\mathcal{L}_0(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\mu}_0}, \dots, \boldsymbol{g}_t = \nabla\mathcal{L}_t(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\mu}_t}$. The general idea is to accumulate the information about the loss function into the current update using recursive least

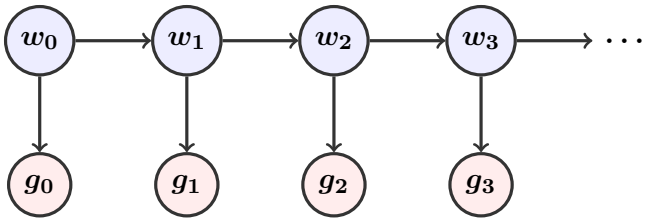

Figure 4: Probabilistic state space model for fitting the quadratic surrogate. The drift of the surrogate parameters $\boldsymbol{w}_t$ over time is modeled using a Gaussian random walk. The observed gradients g_t in each step are modeled using a linear model with Gaussian noise.

squares with a drift model. Taking the derivative on both sides of Eq. (8), the quadratic surrogate of the objective function is equivalent to a linear surrogate of its gradient

$$\boldsymbol{A}_t\boldsymbol{\theta} + \boldsymbol{b}_t \approx \nabla_\theta \mathcal{L}(\boldsymbol{\theta}). \tag{9}$$

As we have seen in Section 2, the update of the parameter distribution $\pi_t(\boldsymbol{\theta})$ does not depend on the scalar parameter c_t of the surrogate. Therefore, the linear surrogate of the gradient contains all the necessary information. Selecting a diagonal matrix $\boldsymbol{A}_t = \operatorname{diag}(\boldsymbol{a}_t)$ leads to independent one-dimensional regression problems for fitting the surrogate. We can find the values $a_t \in \mathbb{R}$ and $b_t \in \mathbb{R}$ from the gradients $g_0, \ldots, g_t \in \mathbb{R}$ evaluated at the parameters $\mu_0, \ldots \mu_t \in \mathbb{R}$ for each dimension of the parameter space independently. To utilize matrix-vector formulations, we define the weight vector

$$\boldsymbol{w}_t = (a_t, b_t)^T.$$

A recursive least squares approach [23] is a capable framework for solving this problem. In the recursive setting, we get one new observation and update our current belief accordingly. The new observation is the gradient g_t that we want to approximate.

In general, there are two stochastic processes we have to model here. First, we allow the surrogate to drift over time since we are not evaluating the gradients at the same point in the parameter space throughout the optimization process. Therefore, older observations may not align with the current surrogate. Second, we have to deal with the fact that the gradient information is noisy due to the noisy objective $\mathcal{L}_t(\boldsymbol{\theta})$. This is a crucial detail since other second-order algorithms like L-BFGS [9] have suffered from this problem [6]. A formulation that addresses these issues builds upon a Bayesian view and utilizes a probabilistic state space model given in Fig. 4. The goal is to compute the parameters $\boldsymbol{m}_t \in \mathbb{R}^2$ and $\boldsymbol{P}_t \in \mathbb{R}^{2\times 2}$ of the filtering distribution

$$p(\boldsymbol{w}_t | g_{0:t}) := \mathcal{N}(\boldsymbol{w}_t | \boldsymbol{m}_t, \boldsymbol{P}_t). \tag{10}$$

We have no information about the development of $\boldsymbol{w}_t$ throughout the optimization. Hence, we model its dynamics as a Gaussian random walk

$$p(\boldsymbol{w}_t | \boldsymbol{w}_{t-1}) = \mathcal{N}(\boldsymbol{w}_t | \boldsymbol{w}_{t-1}, q\boldsymbol{I})$$

with drift variance $q \in \mathbb{R}$. To address the noisy gradient evaluations, we define a measurement model which describes the observation g_t given the current state $\boldsymbol{w}_t$

$$p(g_t|\boldsymbol{w}_t) = \mathcal{N}(g_t|\boldsymbol{H}_t\boldsymbol{w}_t, r).$$

The matrix $\boldsymbol{H}_t = (\mu_t, 1) \in \mathbb{R}^2$ displays the linear relationship of the gradient from Eq. (9) while the value $r \in \mathbb{R}$ models the noise of the gradients. The noise values q and r are hyperparameters for the arTuRO algorithm.

Given a previous state space distribution $p(\boldsymbol{w}_{t-1}|g_{0:t-1}) = \mathcal{N}(\boldsymbol{w}_{t-1}|\boldsymbol{m}_{t-1}, \boldsymbol{P}_{t-1})$, the update equation to obtain the filtering distribution Eq. (10) at step t are the following [23]

$$\begin{aligned}
\boldsymbol{P}_t^- &= \boldsymbol{P}_{t-1} + q\boldsymbol{I} \\
v_t &= \boldsymbol{H}_t\boldsymbol{P}_t^-\boldsymbol{H}_t^T + r \\
\boldsymbol{K}_t &= \boldsymbol{P}_t^-\boldsymbol{H}_t^T v_t^{-1} \\
\boldsymbol{m}_t &= \boldsymbol{m}_{t-1} + \boldsymbol{K}_t(g_t - \boldsymbol{H}_t\boldsymbol{m}_{t-1}) \\
\boldsymbol{P}_t &= \boldsymbol{P}_t^- - v_t\boldsymbol{K}_t\boldsymbol{K}_t^T.
\end{aligned} \tag{11}$$

We use the maximum a posteriori solution $(a_t, b_t)^T := \boldsymbol{m}_t$ to approximate the parameters of the quadratic surrogate Eq. (2). The described one-dimensional computation of $(a_t, b_t)^T$ from the current gradient evaluation g_t shown in Eq. (11) is easily vectorizable and can be implemented efficiently even for high dimensional parameter spaces with millions of parameters.

Appendix C. Experiment Details

This section lists the details for the empirical evaluation in Section 3. We present an illustration of the CNN architectures used for the experiments in Fig. 5. Regarding the ResNet architectures, we use the untrained architectures implemented by PyTorch in the torchvision package. For the Fashion-MNIST task, we change the first layer of the ResNet architecture to accept images with only one channel.

To obtain a fair comparison, we tune all hyperparameters for all algorithms on all available tasks. We use the Weights and Biases sweep functionality (Biewald, 2020). We apply k-fold cross-validation (Kohavi et al., 1995) with $k = 8$ for the HPO since the datasets do not contain a separate validation set. For each configuration of task and algorithm, we test 50 hyperparameter configurations and evaluate each configuration on 3 different seeds. The seed has an impact on the initial weights of the architecture and the shuffling of the dataset. We use a Bayesian optimization (BO) which uses an initial distribution of hyperparameters and proposes new configurations of hyperparameters based on the validation accuracy of previous runs. After obtaining all the hyperparameter configurations per experiment, we cross-evaluated the combinations to see if another combination can perform better than the one found by the BO.

For arTuRO, we have to define the trust region bound ε, the initial variance $\boldsymbol{\Sigma}_0$, the scaling values ρ, ν of the objective, the prior precision λ and the noise values r, q for fitting the surrogate. We use default values for the initial variance $\boldsymbol{\Sigma}_0 = 0.01$ of the parameter distribution, the prior precision $\lambda = 0.0015$, and the scaling $\nu = 1.3$ of the covariance part of the KL since they did not have a big

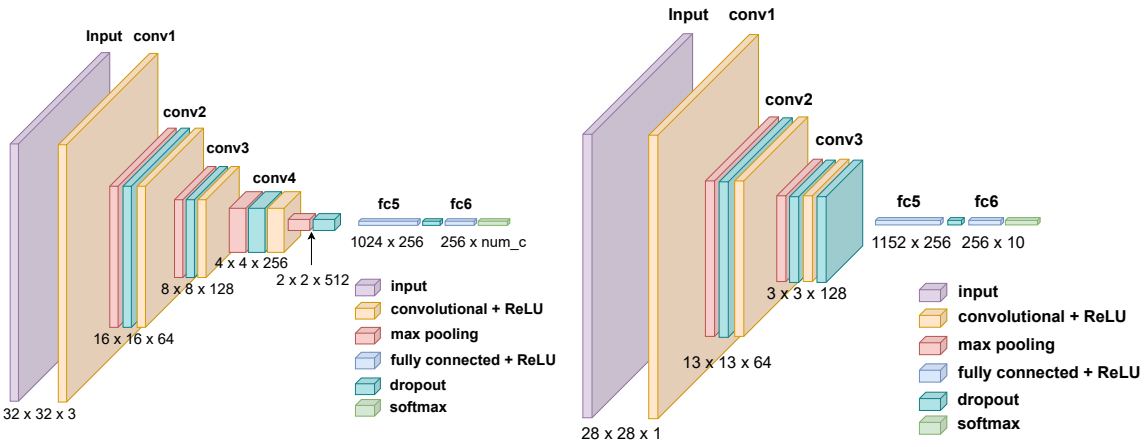

Figure 5: CNN architectures for the CIFAR-10/100 dataset (left) and for the Fashion-MNIST dataset (right).

arTuRO	**Fashion-MNIST**		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
ε	0.085675	0.085675	0.002787	0.007133	0.002787	0.007234
ρ	0.058657	0.058657	0.296786	1.597354	0.296786	1.676770
r	2.816791	2.816791	1.219750	9.328440	1.219750	4.822032
q	0.017393	0.017393	0.002455	0.089381	0.002455	0.009779
Weight Decay	0.000002	0.000002	0.000000	0.000703	0.000000	0.000000

Table 2: Tuned hyperparameter for the arTuRO algorithm

impact upon the results of the optimization. We further select a small value for the initial variance of the filtering distribution $\boldsymbol{P}_0 = \mathrm{diag}((0.00005, 0.00005))$. The bisection method uses a warm start method for the interval bounds based on the previous computation of η^*. The lower and upper bound C_l and C_u are initialized as

$$C_l = \frac{\eta^*}{3},$$
$$C_u = 3\eta^*.$$

The bisection method either stops when

$$C_u - C_l < 0.5,$$

or when

$$|g'(\eta)| < 0.1\varepsilon.$$

This leaves us with four relevant hyperparameters, the trust region bound ε, the prior scaling ρ, and the noise values r and q required for the update of the surrogate model.

The tuned hyperparameters for all algorithms are given in Tables 2 to 5.

SGD	**Fashion-MNIST**		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
Learning Rate	0.071049	0.137031	0.017834	0.056480	0.017834	0.067994
Momentum	0.865730	0.854087	0.946762	0.866487	0.946762	0.867370
Weight Decay	0.000225	0.001963	0.000163	0.001697	0.000163	0.001800

Table 3: Tuned hyperparameter for the SGD algorithm

Adam	**Fashion-MNIST**		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
Learning Rate	0.001012	0.045115	0.001129	0.006652	0.001129	0.001612
β_1	0.945256	0.907895	0.851157	0.890313	0.851157	0.864582
β_2	0.990342	0.999999	0.998940	0.999387	0.998940	0.999953
Weight Decay	0.000000	0.000002	0.001090	0.000447	0.001090	0.001941

Table 4: Tuned hyperparameter for the Adam algorithm

AdamW	**Fashion-MNIST**		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
Learning Rate	0.001004	0.018744	0.001129	0.006652	0.001129	0.001245
β_1	0.922247	0.862748	0.851157	0.890313	0.851157	0.858643
β_2	0.999945	0.999999	0.998940	0.999387	0.998940	0.998802
Weight Decay	0.000142	0.000040	0.001090	0.000447	0.001090	0.001804

Table 5: Tuned hyperparameter for the AdamW algorithm

	Fashion-MNIST		**CIFAR-10**		**CIFAR-100**	
	CNN	ResNet-18	CNN	ResNet-34	CNN	ResNet-34
SGD	1.23	4.19	6.60	9.05	7.09	12.08
Adam	1.33	5.16	6.71	9.83	7.21	11.33
AdamW	1.28	5.13	6.80	9.92	7.18	11.57
arTuRO	1.85	5.42	7.40	15.59	7.92	15.87

Table 6: Average time per training epoch in seconds.

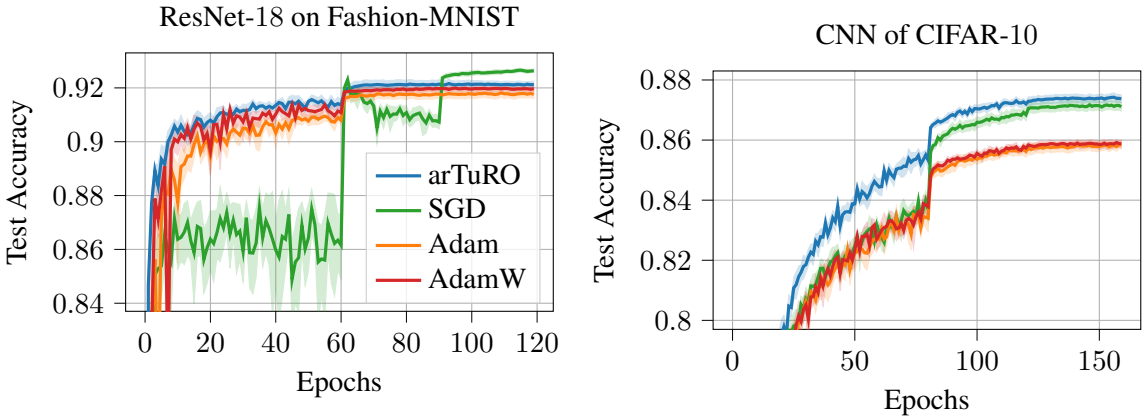

Figure 6: Test accuracies of the competing algorithms arTuRO, SGD, Adam and AdamW on the two remaining experiments ResNet-18 on Fashion-MNIST (left) and CNN on CIFAR-10 (right). arTuRO outperforms Adam and AdamW which is consistent to the other experiments. SGD beats arTuRO on the ResNet-18 task applied to FashionMNIST, however its learning curve shows unstable behavior.

Appendix D. Additional Results

This section lists the missing figures of the test accuracies for the experiments ResNet-18 on Fashion-MNIST and CNN on CIFAR-10 in Fig. 6. We also present the train loss curves of all experiments in Fig. 7.

As mentioned in the paper, the arTuRO algorithm needs a longer computation time compared to state-of-the-art optimizer. We present the average time per training epoch in Table 6. All experiments are run on a single NVIDIA GeForce RTX 3080.

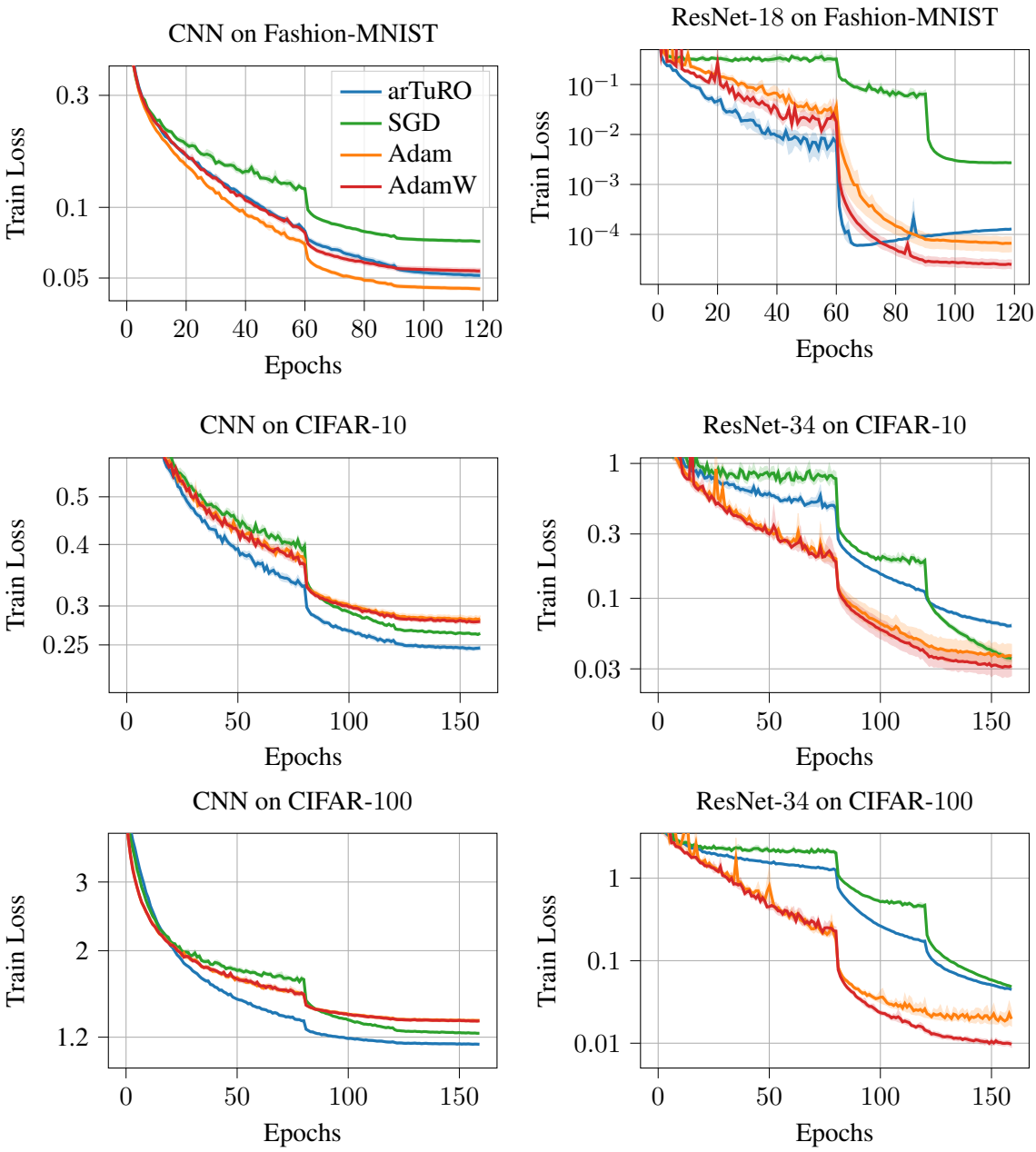

Figure 7: Train loss curves of the competing algorithms arTuRO, SGD, Adam and AdamW.