KOALA: A Kalman Optimization Algorithm with Loss Adaptivity - Supplementary Material -

Aram Davtyan, Sepehr Sameni, Llukman Cerkezi, Givi Meishvili, Adam Bielski, Paolo Favaro
Computer Vision Group, University of Bern, Switzerland
{aram.davtyan, sepehr.sameni, llukman.cerkezi, givi.meishvili, adam.bielski, paolo.favaro}@inf.unibe.ch

With KOALA we introduce a Kalman filtering framework for stochastic optimization that is suitable for training large scale neural networks on machine learning tasks such as image classification, image generation and language modeling. This supplementary material provides details that could not be included in the main paper due to space limitations. In section A, we provide convergence analysis. In section B, we describe the implementation details of the *KOALA-M* (Momentum) algorithm. In section C, we evaluate the performance of different models trained with KOALA and other optimization algorithms to classify images from the CIFAR-10/100 datasets. In section D, we provide numerical results and some visual data from experiments with GANs. In section E, we describe experiments on language modeling.

A Convergence analysis

As mentioned in the main paper, our convergence analysis for KOALA builds on the framework introduced in (Bertsekas and Tsitsiklis 2000). In summary, we show that the gradient of the empirical loss $\hat{L}(x_k)$ goes to 0 as $k \to \infty$ when x_k is computed with KOALA.

In our algorithm, we skip all update steps when the norm of the gradient of a minibatch loss is lower than a positive threshold. This is because such observations provide almost no information to the state. Because of this rule we can thus guarantee that $\forall x$, $|\nabla \hat{L}_k(x)| \geq g$ for some $g > 0$. Notice that although the sequence $\{x_k\}_{k=1,\dots,T}$ may have reached the minimum of $\hat{L}$, so that $\nabla \hat{L}(x_T) = 0$, the gradient of a minibatch loss $\nabla \hat{L}_k(x_T) \neq 0$ in general. In fact, the minibatch losses need only to satisfy $\sum_{k=1}^{T} \nabla \hat{L}_k(x_T) = 0$ and because each minibatch loss is a noisy version of $\hat{L}$, x_T is unlikely to be at one of their minima.

Given these settings, we now analyze the evolution of $\hat{P}_k$ in the case of KOALA-V, and show that it lies between two positive bounds. Recall the update equations (13) from the main paper.

$$\begin{aligned}\hat{P}_k &= P_{k-1} + Q_k \\ P_k &= \frac{R_k}{\hat{P}_k |\nabla \hat{L}_k(x_k)|^2 + R_k} \hat{P}_k.\end{aligned} \tag{31}$$

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The first equation increases $\hat{P}_k$, while the second one decreases it. Notice that all quantities are positive values. Thus, it is straightforward to see that the minimum value for $\hat{P}_k$ is Q_k. Also, it is possible to see that $\hat{P}_k$ will always tend towards

$$\hat{P}_{\text{lim}} \doteq \frac{Q_k + \sqrt{Q_k^2 + 4\frac{R_k Q_k}{|\nabla \hat{L}_k(x_k)|^2}}}{2}. \tag{32}$$

In fact, assume that at both steps k and $k+1$ the covariances Q_k and R_k and $|\nabla L_k(x_k)|^2$ remain constant. When $\hat{P}_k$ is larger than $\hat{P}_{\text{lim}}$ then $\hat{P}_{k+1}$ will decrease and, vice versa, if it is smaller than $\hat{P}_{\text{lim}}$ then $\hat{P}_{k+1}$ will increase. Also, we can see that $\hat{P}_{\text{lim}} \leq 1/2 \left(Q_k + \sqrt{Q_k^2 + 4\frac{R_k Q_k}{g^2}}\right) < \infty$ because R_k and Q_k are finite and $g > 0$.

Now, we are ready to state the convergence of KOALA-V for two choices of the target risks. The proof for KOALA-M requires an extension of (Bertsekas and Tsitsiklis 2000) to non-scalar learning rates. We leave this to future work.

Theorem 1. *Let $\hat{L}(x)$ be a continuously differentiable function and $\nabla \hat{L}(x)$ be Lipschitz-continuous. Assume that $g \leq |\nabla \hat{L}_k(x)| \leq G$ for all x and k, where $g, G > 0$ and $\hat{L}_k(x)$ is the minibatch loss. Additionally, let all x_k lie in a compact space. Let us choose the target risk*

$$\textit{(a)} \qquad \hat{L}_k^{\textit{target}} = \hat{L}_k(x_k) - \varepsilon_k, \tag{33}$$

$$\textit{(b)} \qquad \hat{L}_k^{\textit{target}} = (1 - \varepsilon_k)\hat{L}_k(x_k), \tag{34}$$

where ε_k is any sequence satisfying $\sum_{k=0}^{\infty} \varepsilon_k = \infty$ and $\sum_{k=0}^{\infty} \varepsilon_k^2 < \infty$. Then $\hat{L}(x_k)$ converges to a finite value and $\lim_{k\to\infty} \nabla \hat{L}(x_k) = 0$.

Proof. The analysis is based on Proposition 3 in (Bertsekas and Tsitsiklis 2000) for the general descent algorithm

$$x_{k+1} = x_k + \gamma_k(s_k + u_k), \tag{35}$$

where γ_k is the step size, s_k is a descent direction and u_k is a noise term. Here, s_k is related to the gradient of the empirical risk $\hat{L}$ through two bounds. Similarly, u_k satisfies some regularity conditions. First, let us recall the formula for the learning rate of KOALA-V

$$\eta_k \doteq \frac{\hat{P}_k(\hat{L}_k(x_k) - \hat{L}_k^{\text{target}})}{\hat{P}_k|\nabla \hat{L}_k(x_k)|^2 + R_k} = \frac{\alpha_k(\hat{L}_k(x_k) - \hat{L}_k^{\text{target}})}{|\nabla \hat{L}_k(x_k)|^2}, \tag{36}$$

where $\alpha_k \doteq \frac{\hat{P}_k|\nabla \hat{L}_k(x_k)|^2}{\hat{P}_k|\nabla \hat{L}_k(x_k)|^2 + R_k}$ and we have immediately that $0 < \frac{Q_k g^2}{Q_k g^2 + R_k} \leq \alpha_k \leq 1$. KOALA-V can be seen as a special case of the incremental gradient method shown in Section 5 of (Bertsekas and Tsitsiklis 2000) with

$$s_k = -\mathbb{E}[\phi_k \nabla \hat{L}_k(x_k) \mid \mathcal{F}_k], \tag{37}$$

$$u_k = \mathbb{E}[\phi_k \nabla \hat{L}_k(x_k) \mid \mathcal{F}_k] - \phi_k \nabla \hat{L}_k(x_k), \tag{38}$$

$$\gamma_k = \varepsilon_k, \tag{39}$$

where $\mathcal{F}_k$ is a σ-algebra filtration induced by $x_0, s_0, u_0, \ldots, x_{k-1}, s_{k-1}, u_{k-1}, x_k, s_k$ and $\phi_k \doteq \frac{\eta_k}{\varepsilon_k}$. In can be shown that ϕ_k is positive and bounded from above and below for both target risks *(a)* and *(b)* considered in the theorem. In the target risk *(a)* we have

$$\phi_k = \frac{\alpha_k}{|\nabla \hat{L}_k(x_k)|^2} \tag{40}$$

and in *(b)* we have

$$\phi_k = \frac{\alpha_k}{|\nabla \hat{L}_k(x_k)|^2} \hat{L}_k(x_k). \tag{41}$$

In the second case, it could be that $\hat{L}_k(x_k)$ is zero or negative. To avoid these cases, we can, without loss of generality, add a sufficiently large positive constant to the minibatch loss. This will not change the overall optimization, but will ensure that $\hat{L}_k(x_k)$ is positive. Moreover, the loss is bounded on a compact space due to its continuity. Next, we need to show that s_k, u_k and γ_k satisfy the assumptions of Proposition 3 in (Bertsekas and Tsitsiklis 2000). γ_k satisfies those conditions due to the assumptions on ε_k. An example of a sequence ϵ_k that satisfies both conditions is $\epsilon_k = 1/k$.

Since ϕ_k only depends on the norm of the gradient, and $\nabla \hat{L}_k(x_k)$ is an unbiased estimate of $\nabla \hat{L}(x_k)$, s_k is simply a positive and bounded scaling of the gradient of the empirical loss. Hence, it satisfies the bounds in Proposition 3.

Finally, it is easy to see, that

$$\mathbb{E}[u_k \mid \mathcal{F}_k] = 0, \tag{42}$$

and

$$\begin{aligned}
&\mathbb{E}[\|u_k\|^2 \mid \mathcal{F}_k] = \\
&\quad = \mathbb{E}[\|\phi_k \nabla \hat{L}_k(x_k)\|^2 \mid \mathcal{F}_k] - \|\mathbb{E}[\phi_k \nabla \hat{L}_k(x_k) \mid \mathcal{F}_k]\|^2 \\
&\quad \leq \mathbb{E}[\|\phi_k \nabla \hat{L}_k(x_k)\|^2 \mid \mathcal{F}_k] \\
&\quad \leq \sup_{x_k} \phi_k^2 \cdot \mathbb{E}[\|\nabla \hat{L}_k(x_k)\|^2 \mid \mathcal{F}_k].
\end{aligned} \tag{43}$$

In Section 5 of (Bertsekas and Tsitsiklis 2000) the authors show that $\mathbb{E}[\|\nabla \hat{L}_k(x_k)\|^2 \mid \mathcal{F}_k]$ satisfies the bounds required by Proposition 3 and, hence, u_k also satisfies those bounds. □

B KOALA-M Algorithm

In this section we provide the derivations and formulas for the *KOALA-M* algorithm that was used in the main paper as the base for the *KOALA* algorithm.

The dynamic system is given by the following equations

$$\bar{x}_k = F\bar{x}_{k-1} + \zeta_{k-1} \tag{44}$$

$$\hat{L}_k^{\text{target}} = \hat{L}_k(\Pi\bar{x}_k) + v_k, \tag{45}$$

where $F = \begin{bmatrix} I_{n\times n} & I_{n\times n} \\ \mathbf{0} & \kappa I_{n\times n} \end{bmatrix}$, $\Pi = \begin{bmatrix} I_{n\times n} & \mathbf{0} \end{bmatrix}$ and $\bar{x}_k = (x_k, p_k)$ is a concatenated vector of the weights and the velocities.

Here the posterior error covariance is a $2n \times 2n$ matrix, therefore we approximate it with the following form

$$P_k = \begin{bmatrix} \sigma_{x,k}^2 I_{n\times n} & \sigma_{c,k}^2 I_{n\times n} \\ \sigma_{c,k}^2 I_{n\times n} & \sigma_{p,k}^2 I_{n\times n} \end{bmatrix}, \tag{46}$$

where $\sigma_{x,k}^2$, $\sigma_{c,k}^2$, $\sigma_{p,k}^2$ are scalars. This approximation allows us to consider only 2×2 matrix operations, which leads to much more time- and space- efficient calculations.

In this formulation the measurement function does not depend on the velocities, hence we have

$$H_k = \begin{bmatrix} \nabla \hat{L}_k^\top & \mathbf{0}^\top \end{bmatrix} \tag{47}$$

and thus our approximation for the Kalman update of the posterior covariance will use

$$H_k^\top H_k \approx \begin{bmatrix} |\nabla \hat{L}_k|^2 I_{n\times n} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{48}$$

Recall the adaptive R_k and Q_k estimations, as well as layer-wise updates. The KOALA-M algorithm consists of these equations and the original Kalman Filter recursive equations. The pseudocode of this algorithm is given in Algorithm 2.

C Comparison of CIFAR-10/100 Classification across Different Architectures

In section 5.1 of the main paper we describe the settings for the experiments with classification on CIFAR-10/100 (Krizhevsky and Hinton 2009). Here we provide more figures for a better comparison of KOALA's performance across different architectures. The plots from the experiments on CIFAR-10 and CIFAR-100 are given in Figures 3 and 4 respectively. Moreover, in addition to the process noise ablation we show its evolution over time in Figure 1.

D Generative Adversarial Networks Training

GANs (Goodfellow et al. 2014) are models trained to generate new samples from a given data distribution. A common practice for training GANs is to use adaptive methods like Adam or RMSProp. Following (Zhuang et al. 2020), we test our method on Wasserstein GAN (Arjovsky, Chintala, and

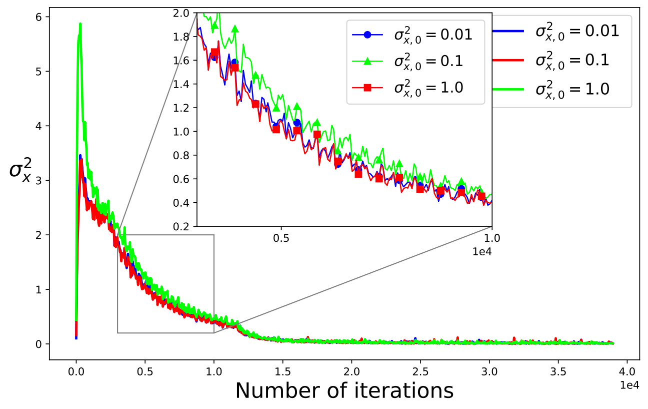


Figure 1: Evolution of σ_x^2 over time depending on the values we initialize it with. It can be seen that the parameter converges quite quickly to the same value regardless of the initialization.

Bottou 2017) with gradient penalty (WGAN-GP) (Gulrajani et al. 2017). We control the measurements equations in KOALA by adjusting the $\hat{L}_k^{\text{target}}$, as was done to obtain learning rate schedules. For a given minibatch $\mathcal{S}_k$ the target is set to $\hat{L}_k^{\text{target}} = \hat{L}_k(x_k) - u_k$, for some $u_k > 0$, which we set before the training. In our experiments, we fix $u_k = 4$. We also set $\kappa = 0.5$ similarly to a common choice of $\beta_1 = 0.5$ for Adam in GAN training.

We use a DCGAN (Radford, Metz, and Chintala 2015) architecture with WGAN-GP loss. For each optimizer, we train the model for 100 epochs on CIFAR-10 and evaluate the FID score (Heusel et al. 2017). For the training we use the code provided by the authors of (Zhuang et al. 2020) in the following repository: https://github.com/juntang-zhuang/Adabelief-Optimizer/tree/update_0.2.0/PyTorch_Experiments/wgan-gp.

We report the mean and standard deviation among 4 runs with different random seeds. Usually GANs are trained in an alternating way, that is the generator is updated every n_{critic} iterations. We test KOALA on two settings: $n_{\text{critic}} = 1, 5$. On both settings KOALA outperforms Adam and is more stable. The results and sample images are shown in Table 2 and Figure 2 respectively.

E Language modeling

Given the recent success of transfer learning in NLP with pretrained language models (Peters et al. 2018; Yang et al. 2019; Devlin et al. 2019; Clark et al. 2020), we trained both an LSTM (Hochreiter and Schmidhuber 1997) and also a Transformer (Vaswani et al. 2017) for language modeling on the Penn TreeBank dataset (Marcus, Santorini, and Marcinkiewicz 1993) and Wikitext-2 (Merity et al. 2017). We use the default data splits for training and validation and report the perplexity on the test set in Table 1.

We used a two layer LSTM with 200 hidden neurons and input embedding size of 200 for our tiny-LSTM (the same settings as (Bernstein et al. 2020)) and increased the sizes to 1500 for the larger-LSTM experiment (the same as (Zhang, Mitliagkas, and Ré 2017)). The learning rate for Adam and SGD were picked based on a grid search and we used the default learning rate of 1.0 for our optimizer. In order to prevent overfitting we used an aggressive dropout (Srivastava et al. 2014) rate of 0.65 and tied the input and output embeddings (Press and Wolf 2017). Since we are using small datasets, we use only a two-layer masked multi-head self-attention transformer with two heads, which performs worse than LSTM. We find that even in these settings, KOALA is better or on-par with other optimizers.

Table 1: Language modeling perplexity (lower is better) on different datasets, architectures and optimizers.

Model	Optimizer	PTB ppl	WikiText-2 ppl
tiny-LSTM	SGD	207.7	216.62
	Adam	123.6	166.49
	KOALA-M	**110.1**	**124.69**
larger-LSTM	SGD	112.19	145.7
	Adam	**81.27**	94.39
	KOALA-M	81.57	**89.64**
tiny-Transformer	SGD	134.41	**169.04**
	Adam	140.13	189.66
	KOALA-M	**129.45**	179.81

Table 2: FID (mean±std, lower is better) for different optimizers in GAN training on CIFAR-10.

Optimizer	FID ↓ $n_{\text{critic}} = 1$	FID ↓ $n_{\text{critic}} = 5$
Adam	78.21±15.91	79.05±7.89
KOALA-M	**67.17±14.93**	**75.65±5.38**

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR.

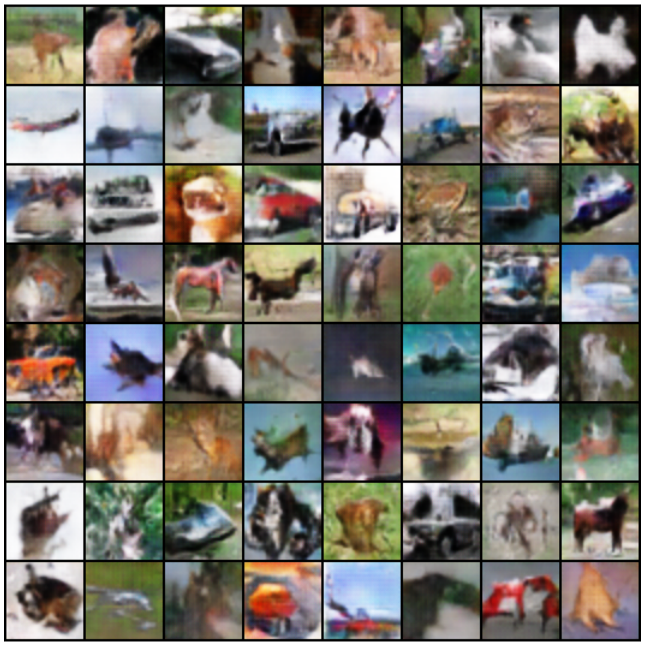
Bernstein, J.; Vahdat, A.; Yue, Y.; and Liu, M.-Y. 2020. On the distance between two neural networks and the stability of learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*.

Bertsekas, D. P.; and Tsitsiklis, J. N. 2000. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, (3).

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ArXiv*, abs/2003.10555.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. *Advances in Neural Information Processing Systems*.


(a) Adam, FID=64.71

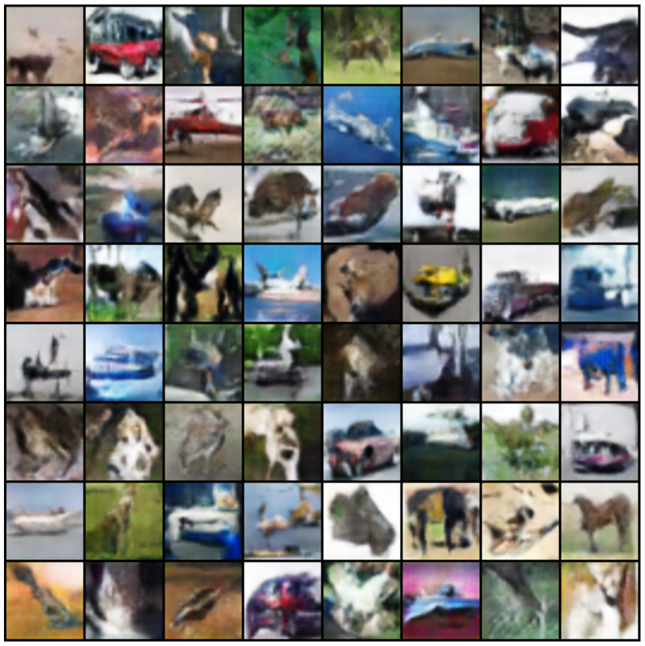

(b) KOALA, FID=55.41

Figure 2: Samples generated by GANs trained on CIFAR- 10; best run across different random seeds for each optimizer.

Table 3: Class. err. & wall-time (CIFAR-100/WideResNet50-2).

Optimizer	Top-1 error (wall-time)	
	after 3h of training	after 100 epochs
SGD	24.98 (3h)	23.92 (3h 40m)
Adam	23.86 (3h)	23.42 (4h 21m)
KOALA	21.79 (3h)	21.43 (4h 43m)

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images.

Marcus, M. P.; Santorini, B.; and Marcinkiewicz, M. A. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*.

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer Sentinel Mixture Models. *ArXiv*, abs/1609.07843.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana.

Press, O.; and Wolf, L. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 157–163. Valencia, Spain.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. *ArXiv*, abs/1706.03762.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*.

Zhang, J.; Mitliagkas, I.; and Ré, C. 2017. YellowFin and the Art of Momentum Tuning. *arXiv preprint arXiv:1706.03471*.

Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S. C.; Dvornek, N.; Papademetris, X.; and Duncan, J. 2020. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. In *Advances in Neural Information Processing Systems*.

Algorithm 2: KOALA-M (Momentum)

```
Init. $x_0$, $P_0 = \begin{bmatrix} \sigma^2_{x,0} & \sigma^2_{c,0} \\ \sigma^2_{c,0} & \sigma^2_{p,0} \end{bmatrix}$, $F = \begin{bmatrix} 1 & 1 \\ 0 & \kappa \end{bmatrix}$, $R_0$, $q_p^2$.
for $k$ in range(1, $T$) do
    Sample a minibatch of data $\mathcal{S}_k$. Skip iteration if for any
    $i \in [1, B]$, $|\nabla_{b_i} L_k(x_k)| < \theta$, where $\theta > 0$ is a small
    scalar.
    Update noise estimates:
    $R_k = \beta_R R_k + (1 - \beta_R)\frac{1}{|\mathcal{S}_k|}\sum_{i \in \mathcal{S}_k}\left(\ell(\xi_i; x_k) - \hat{L}_k^{\text{target}}\right)^2$   (49)
    $x_{\text{avg}} = \beta_x x_{\text{avg}} + (1 - \beta_x) x_{k-1}$   (50)
    $q^2_{x,k} = \frac{1}{n}|x_{k-1} - x_{\text{avg}}|^2$   (51)
    $Q_k = \begin{bmatrix} q^2_{x,k} & 0 \\ 0 & q_p^2 \end{bmatrix}$   (52)
    Predict:
    $\hat{x}_k = x_{k-1} + p_{k-1}$   (53)
    $\hat{p}_k = \kappa p_{k-1}$   (54)
    $\hat{P}_k = F P_{k-1} F^T + Q_k$   (55)
    Update:
    for $i$ in range(1, $B$) do
        $S_k^{(i)} = |\nabla_{b_i} \hat{L}_k|^2 \hat{P}_k[1,1] + R_k$   (56)
        $K_{k,x} = \frac{\hat{P}_k[1,1]\left(\hat{L}_k(\hat{x}_k) - \hat{L}_k^{\text{target}}\right)}{S_k^{(i)}}$   (57)
        $K_{k,p} = \frac{\hat{P}_k[2,1]\left(\hat{L}_k(\hat{x}_k) - \hat{L}_k^{\text{target}}\right)}{S_k^{(i)}}$   (58)
        $x_{k,b_i} = \hat{x}_{k,b_i} - K_{k,x}\nabla_{b_i}\hat{L}_k$   (59)
        $p_{k,b_i} = \hat{p}_{k,b_i} - K_{k,p}\nabla_{b_i}\hat{L}_k$   (60)
    end for
    $\widetilde{h}_k = \max_{1 \le i \le B} \frac{|\nabla_{b_i}\hat{L}_k|^2}{S_k^{(i)}}$   (61)
    $\widetilde{H}_k = \begin{bmatrix} \widetilde{h}_k & 0 \\ 0 & 0 \end{bmatrix}$   (62)
    $P_k = \hat{P}_k - \hat{P}_k \widetilde{H}_k \hat{P}_k^T$   (63)
end for
return $x_K$
```

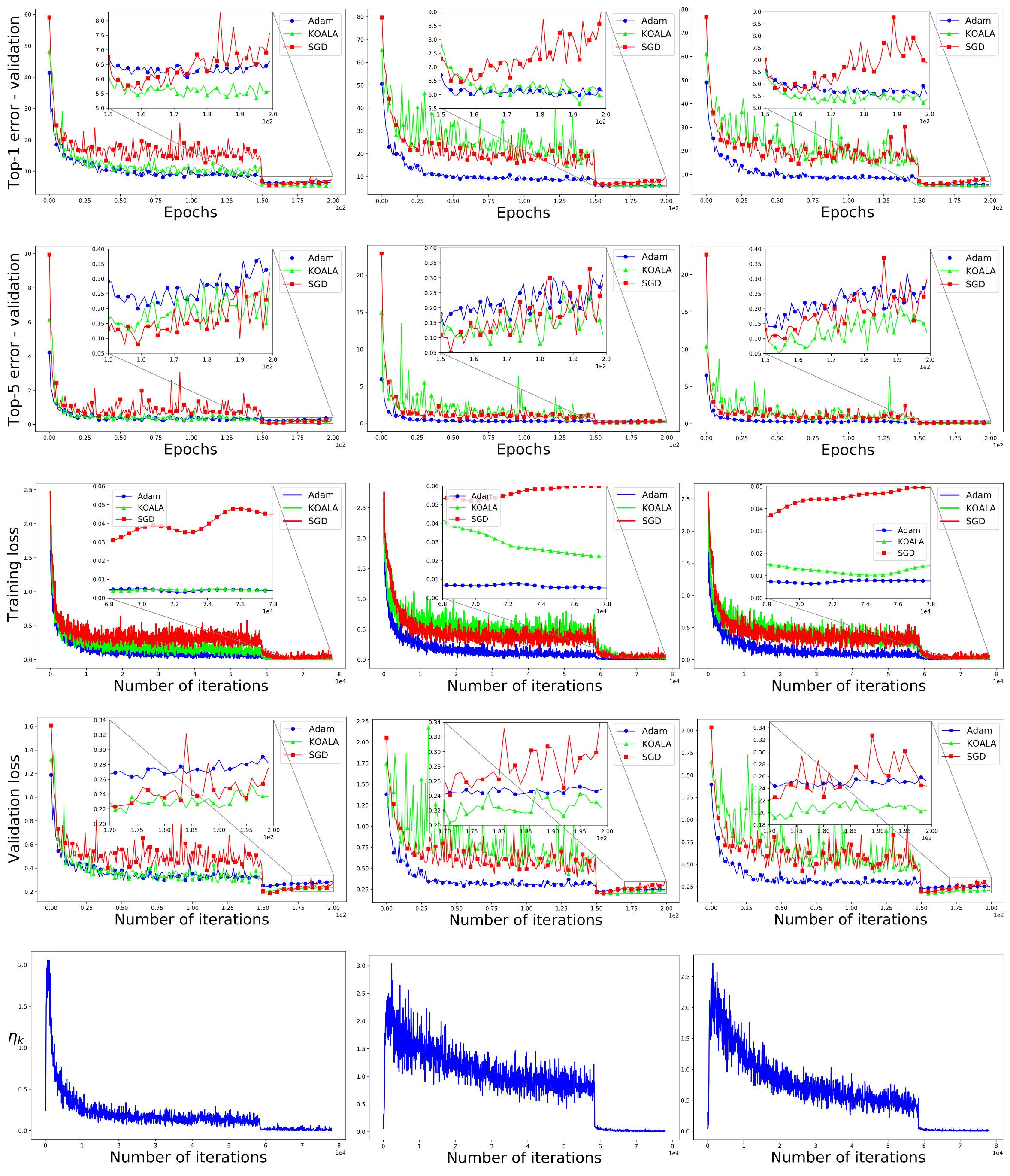


Figure 3: Classification on the CIFAR-10 dataset. 200-epochs run. Columns (from right to left): Resnet-18, Resnet-50, WideResnet-50-2. Rows (from top to bottom): Top-1 error, Top-5 error, training loss, validation loss, KOALA's step size η_k, *i.e.*, the value scaling the gradient in the update equation. We show the best runs for all models across different random seeds.

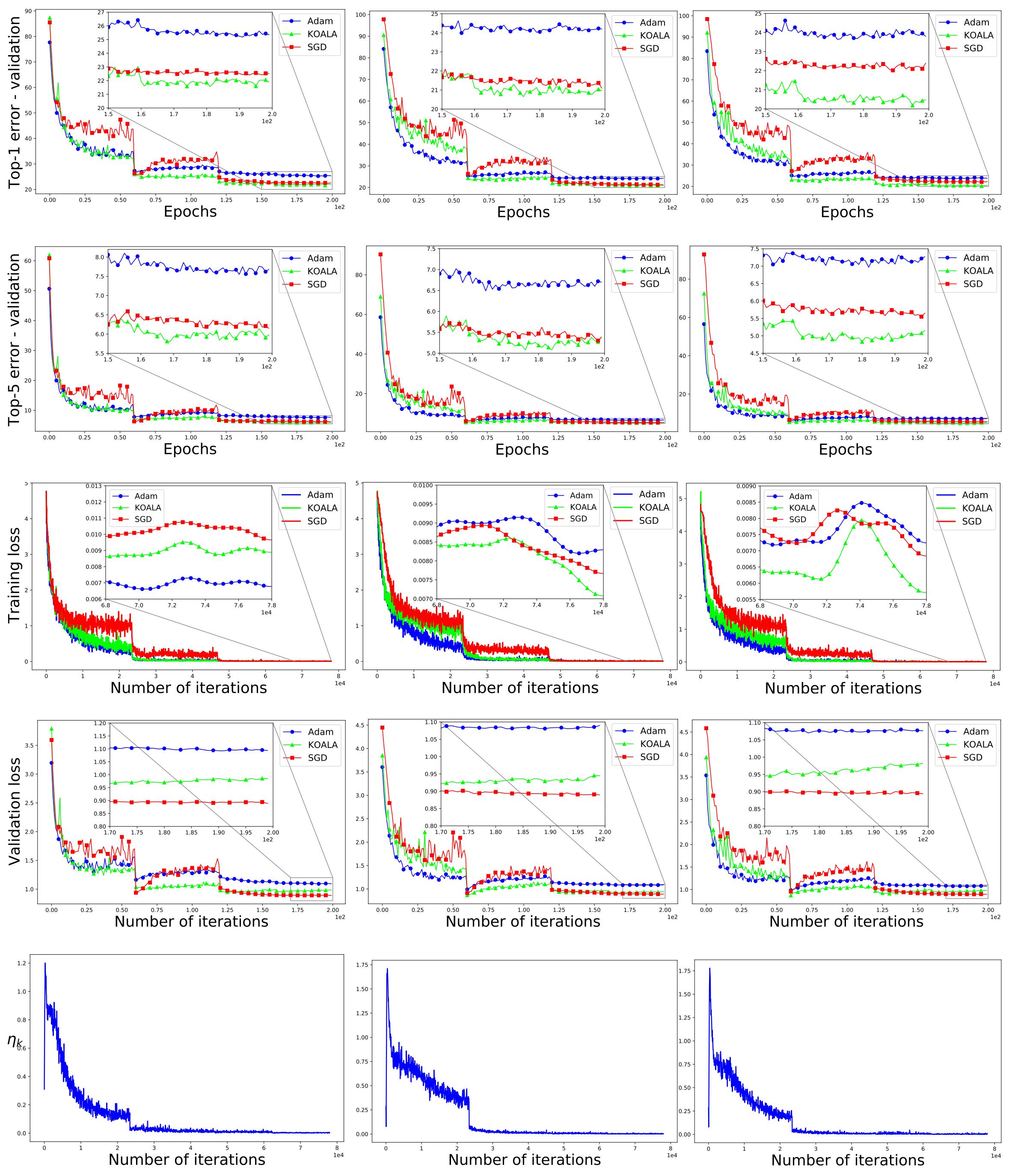


Figure 4: Classification on the CIFAR-100 dataset. 200-epochs run. Columns (from right to left): Resnet-18, Resnet-50, WideResnet-50-2. Rows (from top to bottom): Top-1 error, Top-5 error, training loss, validation loss, KOALA's step size η_k, *i.e.*, the value scaling the gradient in the update equation. We show the best runs for all models across different random seeds.