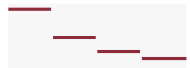
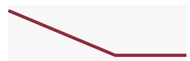
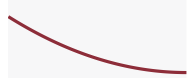
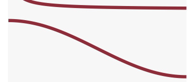
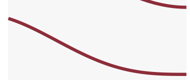
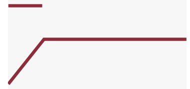
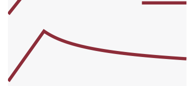
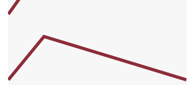
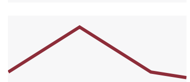
A. List of optimizers and schedules considered

Table 2: List of optimizers considered for our benchmark. This is only a subset of all existing methods for deep learning.

Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KF-QN-CNN	(Ren & Goldfarb, 2021)
AdaBatch	(Devarakonda et al., 2017)	KFAC	(Martens & Grosse, 2015)
AdaBayes/AdaBayes-SS	(Aitchison, 2020)	KFLR/KFRA	(Botev et al., 2017)
AdaBelief	(Zhuang et al., 2020)	L4Adam/L4Momentum	(Rolínek & Martius, 2018)
AdaBlock	(Yun et al., 2019)	LAMB	(You et al., 2020)
AdaBound	(Luo et al., 2019)	LaProp	(Ziyin et al., 2020)
AdaComp	(Chen et al., 2018)	LARS	(You et al., 2017)
Adadelta	(Zeiler, 2012)	LHOPT	(Almeida et al., 2021)
Adafactor	(Shazeer & Stern, 2018)	LookAhead	(Zhang et al., 2019)
AdaFix	(Bae et al., 2019)	M-SVAG	(Balles & Hennig, 2018)
AdaFom	(Chen et al., 2019a)	MADGRAD	(Defazio & Jelassi, 2021)
AdaFTRL	(Orabona & Pál, 2015)	MAS	(Landro et al., 2020)
Adagrad	(Duchi et al., 2011)	MEKA	(Chen et al., 2020b)
ADAHESSIAN	(Yao et al., 2020)	MTAdam	(Malkiel & Wolf, 2020)
Adai	(Xie et al., 2020)	MVRC-1/MVRC-2	(Chen & Zhou, 2020)
AdaLoss	(Teixeira et al., 2019)	Nadam	(Dozat, 2016)
Adam	(Kingma & Ba, 2015)	NAMSB/NAMSG	(Chen et al., 2019b)
Adam$^+$	(Liu et al., 2020b)	ND-Adam	(Zhang et al., 2017a)
AdamAL	(Tao et al., 2019)	Nero	(Liu et al., 2021b)
AdaMax	(Kingma & Ba, 2015)	Nesterov	(Nesterov, 1983)
AdamBS	(Liu et al., 2020c)	Noisy Adam/Noisy K-FAC	(Zhang et al., 2018)
AdamNC	(Reddi et al., 2018)	NosAdam	(Huang et al., 2019)
AdaMod	(Ding et al., 2019)	Novograd	(Ginsburg et al., 2019)
AdamP/SGDP	(Heo et al., 2021)	NT-SGD	(Zhou et al., 2021b)
AdamT	(Zhou et al., 2020)	Padam	(Chen et al., 2020a)
AdamW	(Loshchilov & Hutter, 2019)	PAGE	(Li et al., 2020b)
AdamX	(Tran & Phong, 2019)	PAL	(Mutschler & Zell, 2020)
ADAS	(Eliyahu, 2020)	PolyAdam	(Orvieto et al., 2019)
AdaS	(Hosseini & Plataniotis, 2020)	Polyak	(Polyak, 1964)
AdaScale	(Johnson et al., 2020)	PowerSGD/PowerSGDM	(Vogels et al., 2019)
AdaSGD	(Wang & Wiens, 2020)	Probabilistic Polyak	(de Roos et al., 2021)
AdaShift	(Zhou et al., 2019)	ProbLS	(Mahsereci & Hennig, 2017)
AdaSqrt	(Hu et al., 2019)	PStorm	(Xu, 2020)
Adathm	(Sun et al., 2019)	QHAdam/QHM	(Ma & Yarats, 2019)
AdaX/AdaX-W	(Li et al., 2020a)	RAdam	(Liu et al., 2020a)
AEGD	(Liu & Tian, 2020)	Ranger	(Wright, 2020b)
ALI-G	(Berrada et al., 2020)	RangerLars	(Grankin, 2020)
AMSBound	(Luo et al., 2019)	RMSProp	(Tieleman & Hinton, 2012)
AMSGrad	(Reddi et al., 2018)	RMSterov	(Choi et al., 2019)
AngularGrad	(Roy et al., 2021)	S-SGD	(Sung et al., 2020)
ArmijoLS	(Vaswani et al., 2019)	SAdam	(Wang et al., 2020b)
ARSG	(Chen et al., 2019b)	Sadam/SAMSGrad	(Tong et al., 2019)
ASAM	(Kwon et al., 2021)	SALR	(Yue et al., 2020)
AutoLRS	(Jin et al., 2021)	SAM	(Foret et al., 2021)
AvaGrad	(Savarese et al., 2019)	SC-Adagrad/SC-RMSProp	(Mukkamala & Hein, 2017)
BAdam	(Salas et al., 2018)	SDProp	(Ida et al., 2017)
BGAdam	(Bai & Zhang, 2019)	SGD	(Robbins & Monro, 1951)
BPGrad	(Zhang et al., 2017b)	SGD-BB	(Tan et al., 2016)
BRMSProp	(Aitchison, 2020)	SGD-G2	(Ayadi & Turinici, 2020)
BSGD	(Hu et al., 2020)	SGDEM	(Ramezani-Kebrya et al., 2021)
C-ADAM	(Tutunov et al., 2020)	SGDHess	(Tran & Cutkosky, 2021)
CADA	(Chen et al., 2021)	SGDM	(Liu & Luo, 2020)
Cool Momentum	(Borysenko & Byshkin, 2020)	SGDR	(Loshchilov & Hutter, 2017)
CProp	(Preechakul & Kijsirikul, 2019)	SHAdagrad	(Huang et al., 2020)
Curveball	(Henriques et al., 2019)	Shampoo	(Anil et al., 2020; Gupta et al., 2018)
Dadam	(Nazari et al., 2019)	SignAdam++	(Wang et al., 2019a)
DeepMemory	(Wright, 2020a)	SignSGD	(Bernstein et al., 2018)
DGNOpt	(Liu et al., 2021a)	SKQN/S4QN	(Yang et al., 2020)
DiffGrad	(Dubey et al., 2020)	SM3	(Anil et al., 2019)
EAdam	(Yuan & Gao, 2020)	SMG	(Tran et al., 2020)
EKFAC	(George et al., 2018)	SNGM	(Zhao et al., 2020)
Eve	(Hayashi et al., 2018)	SoftAdam	(Fetterman et al., 2019)
Expectigrad	(Daley & Amato, 2020)	SRSGD	(Wang et al., 2020a)
FastAdaBelief	(Zhou et al., 2021a)	Step-Tuned SGD	(Castera et al., 2021)
FRSGD	(Wang & Ye, 2020)	SWATS	(Keskar & Socher, 2017)
G-AdaGrad	(Chakrabarti & Chopra, 2021)	SWNTS	(Chen et al., 2019c)
GADAM	(Zhang & Gouza, 2018)	TAdam	(Ilboudo et al., 2020)
Gadam	(Granziol et al., 2020)	TEKFAC	(Gao et al., 2020)
GOALS	(Chae et al., 2021)	VAdam	(Khan et al., 2018)
GOLS-I	(Kafka & Wilke, 2019)	VR-SGD	(Shang et al., 2020)
Grad-Avg	(Purkayastha & Purkayastha, 2020)	vSGD-b/vSGD-g/vSGD-l	(Schaul et al., 2013)
GRAPES	(Dellaferrera et al., 2021)	vSGD-fd	(Schaul & LeCun, 2013)
Gravilon	(Kelterborn et al., 2020)	WNGrad	(Wu et al., 2018)
Gravity	(Bahrami & Zadeh, 2021)	YellowFin	(Zhang & Mitliagkas, 2019)
HAdam	(Jiang et al., 2019)	Yogi	(Zaheer et al., 2018)

Table 3: Overview of commonly used parameter schedules. Note, while we list the schedules parameters, it isn't clearly defined what aspects of a schedule are (tunable) parameters and what is a-priori fixed. In this column, α_0 denotes the initial learning rate, α_{lo} and α_{up} the lower and upper bound, Δt indicates an epoch count at which to switch decay styles, k denotes a decaying factor.

Name		Ref.	Illustration	Parameters
Constant				α_0
Step Decay	constant factor			$\alpha_0, \Delta t_1, \ldots, k$
	multi-step			$\alpha_0, \Delta t_1, \ldots, k_1, \ldots$
Smooth Decay	linear decay	e.g. (Goodfellow et al., 2016)		$\alpha_0, (\Delta t, \alpha_{\text{lo}})$
	polynomial decay			$\alpha_0, k, (\alpha_{\text{lo}})$
	exponential decay			$\alpha_0, k, (\alpha_{\text{lo}})$
	inverse time decay	e.g. (Bottou, 2012)		$\alpha_0, k, (\alpha_{\text{lo}})$
	cosine decay	(Loshchilov & Hutter, 2017)		$\alpha_0, (\alpha_{\text{lo}})$
	linear cosine decay	(Bello et al., 2017)		$\alpha_0, (\alpha_{\text{lo}})$
Cyclical	triangular	(Smith, 2017)		$\alpha_{\text{lo}}, \alpha_{\text{up}}, \Delta t$
	triangular + decay	(Smith, 2017)		$\alpha_{\text{lo}}, \alpha_{\text{up}}, \Delta t, k$
	triangular + exponential decay	(Smith, 2017)		$\alpha_{\text{lo}}, \alpha_{\text{up}}, \Delta t$
	cosine + warm restarts	(Loshchilov & Hutter, 2017)		$\alpha_{\text{up}}, \Delta t, (\alpha_{\text{lo}})$
	cosine + warm restarts + decay	(Loshchilov & Hutter, 2017)		$\alpha_{\text{up}}, \Delta t, k, (\alpha_{\text{lo}})$
Warmup	constant warmup	e.g. (He et al., 2016)		$\alpha_{\text{lo}}, \alpha_0, \Delta t$
	gradual warmup	(Goyal et al., 2017)		$\alpha_0, \Delta t, (\alpha_{\text{lo}})$
	gradual warmup + multi-step decay	(Goyal et al., 2017)		$\alpha_0, \Delta t, \Delta t_{\text{steps}}, k_1, \ldots, (\alpha_{\text{lo}})$
	gradual warmup + step number decay	(Vaswani et al., 2017)		$\alpha_0, \Delta t, (\alpha_{\text{lo}})$
	slanted triangular	(Howard & Ruder, 2018)		$\alpha_0, \Delta t, (\alpha_{\text{lo}})$
	long trapezoid	(Xing et al., 2018)		$\alpha_0, \Delta t_{\text{up}}, \Delta t_{\text{down}}, (\alpha_{\text{lo}})$
Super-Convergence	1cycle	(Smith & Topin, 2017)		$\alpha_{\text{up}}, \Delta t, \Delta t_{\text{cutoff}}, (\alpha_{\text{lo}})$

B. List of optimizers selected

Table 4: Selected optimizers for our benchmarking process with their respective color, hyperparameters, default values, tuning distributions and scheduled hyperparameters. Here, $\mathcal{LU}(\cdot,\cdot)$ denotes the log-uniform distribution while $\mathcal{U}\{\cdot,\cdot\}$ denotes the discrete uniform distribution.

Optimizer	Ref.	Parameters	Default	Tuning Distribution	Scheduled
AMSBound	(Luo et al., 2019)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		α_l	0.1	$\mathcal{LU}(10^{-3}, 0.5)$	
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		γ	10^{-3}	$\mathcal{LU}(10^{-4}, 10^{-1})$	
		ε	10^{-8}	✗	
AMSGrad	(Reddi et al., 2018)	α	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-8}	✗	
AdaBelief	(Zhuang et al., 2020)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-14}	✗	
AdaBound	(Luo et al., 2019)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		α_l	0.1	$\mathcal{LU}(10^{-3}, 0.5)$	
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		γ	10^{-3}	$\mathcal{LU}(10^{-4}, 10^{-1})$	
AdaDelta	(Zeiler, 2012)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		ε	10^{-8}	✗	
		$1-\rho$	0.95	$\mathcal{LU}(10^{-4}, 1)$	
AdaGrad	(Duchi et al., 2011)	α	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓
		ε	10^{-7}	✗	
Adam	(Kingma & Ba, 2015)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-8}	✗	
Lookahead Momentum abbr. LA(Mom.)	(Zhang et al., 2019)	α	0.5	$\mathcal{LU}(10^{-4}, 1)$	
		α_f	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓
		k	5	$\mathcal{U}\{1, 20\}$	
		$1-\rho$	0.99	$\mathcal{LU}(10^{-4}, 1)$	
Lookahead RAdam abbr. LA(RAdam)	(Zhang et al., 2019)	α	0.5	$\mathcal{LU}(10^{-4}, 1)$	
		α_f	10^{-3}	$\mathcal{LU}(1e-4, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-7}	✗	
		k	5	$\mathcal{U}\{1, 20\}$	
Momentum	(Polyak, 1964)	α	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓
		$1-\rho$	0.99	$\mathcal{LU}(10^{-4}, 1)$	
NAG	(Nesterov, 1983)	α	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓
		$1-\rho$	0.99	$\mathcal{LU}(10^{-4}, 1)$	
Nadam	(Dozat, 2016)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-7}	✗	
RAdam	(Liu et al., 2020a)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		β_1	0.9	$\mathcal{LU}(0.5, 0.999)$	
		β_2	0.999	$\mathcal{LU}(0.8, 0.999)$	
		ε	10^{-7}	✗	
RMSProp	(Tieleman & Hinton, 2012)	α	10^{-3}	$\mathcal{LU}(10^{-4}, 1)$	✓
		ε	10^{-10}	✗	
		$1-\rho$	0.9	$\mathcal{LU}(10^{-4}, 1)$	
SGD	(Robbins & Monro, 1951)	α	10^{-2}	$\mathcal{LU}(10^{-4}, 1)$	✓

C. Robustness to random seeds

Data subsampling, random weight initialization, dropout and other aspects of deep learning introduce stochasticity to the training process. As such, judging the performance of an optimizer on a single run may be misleading due to random fluctuations. In our benchmark we use 10 different seeds of the final setting for each budget in order to judge the stability of the optimizer and the results. However, to keep the magnitude of this benchmark feasible, we only use a single seed while tuning, analogously to how a single user would progress. This means that our tuning process can sometimes choose hyperparameter settings which might not even converge for seeds other than the one used for tuning.

Figure 5 illustrates this behavior on an example problem where we used 10 seeds throughout a tuning process using grid search. The figure shows that in the beginning performance increases when increasing the learning rate, followed by an area were it sometimes works but other times diverges. Picking hyperparameters from this "danger zone" can lead to unstable results. In this case, where we only consider the learning rate, it is clear that decreasing the learning rate a bit to get away from this "danger zone" would lead to a more stable, but equally well-performing algorithm. In more complicated cases, however, we are unable to use a simple heuristic such as this. This might be the case, for example, when tuning multiple hyperparameters or when the effect of the hyperparameter on the performance is less straight forward. Thus, this is a problem not created by improperly using the tuning method, but by an unstable optimization method.

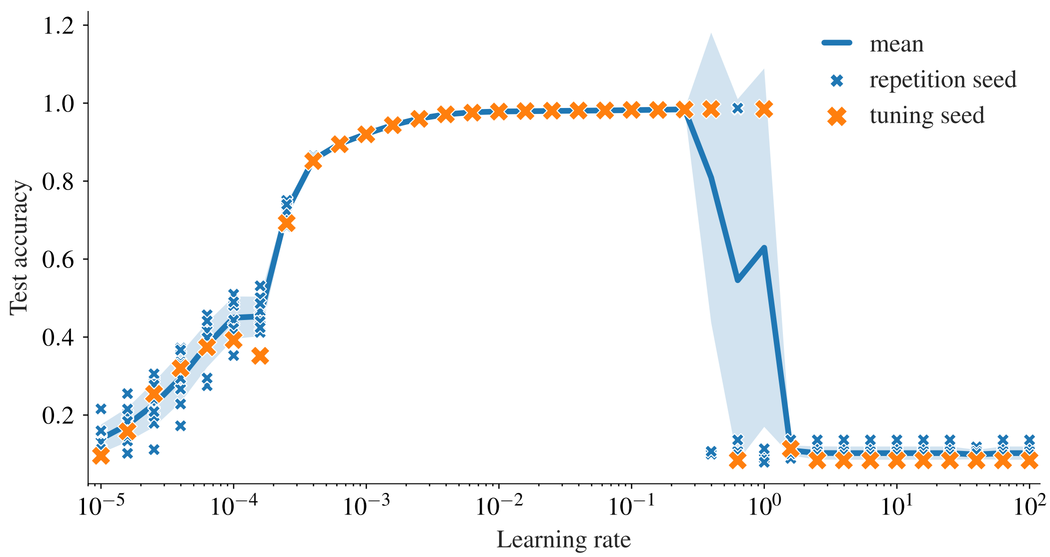


Figure 5: Performance of SGD on a simple multilayer perceptron. For each learning rate, markers in orange (✖) show the initial seed which would be used for tuning, blue markers (✖) illustrate nine additional seeds with otherwise unchanged settings. The mean over all seeds is plotted as a blue line (—), showing one standard deviation as a shaded area ( ).

In our benchmark, we observe a total of 18, 24, and 17 divergent seeds for the small, medium, and large budget respectively. This amounts to roughly 0.5% of the runs in each budget. Most of them occur when using SGD (10, 15, and 7 cases for the small, medium and large budget respectively), ADAGRAD (5, 3, and 5 cases for the small, medium and large budget respectively) or ADADELTA (3, 5, and 3 cases for the small, medium and large budget respectively), which might indicate that modern adaptive methods are less prone to this kind of behavior. None of these cases occur when using a constant schedule, and most of them occur when using the *trapezoidal* schedule (11, 11, and 9 cases for the small, medium and large budget respectively). However, as our data on diverging seeds is very limited, it is not conclusive enough to draw solid conclusions.

D. Re-Tuning experiments

In order to test the stability of our benchmark and especially the tuning method, we selected two optimizers in our benchmark and re-tuned them on all problems a second time. We used completely independent random seeds for both tuning and the 10 repetitions with the final setting. Figure 6 and Figure 7 show the distribution of all 10 random seeds for both the original tuning as well as the re-tuning runs for RMSProp and AdaDelta. It is evident, that re-tuning results in a shift of this distribution, since small (stochastic) changes during tuning can result in a different chosen hyperparameter setting.

These differences also highlight how crucial it is to look at multiple problems. Individually, small changes, such as re-doing the tuning with different seeds can lead to optimization methods changing rankings. However, they tend to average out when looking at an unbiased list of multiple problems. These results also further supports the statement made in Section 3 that there is no optimization method clearly domination the competition, as small performance margins might vanish when re-tuning.

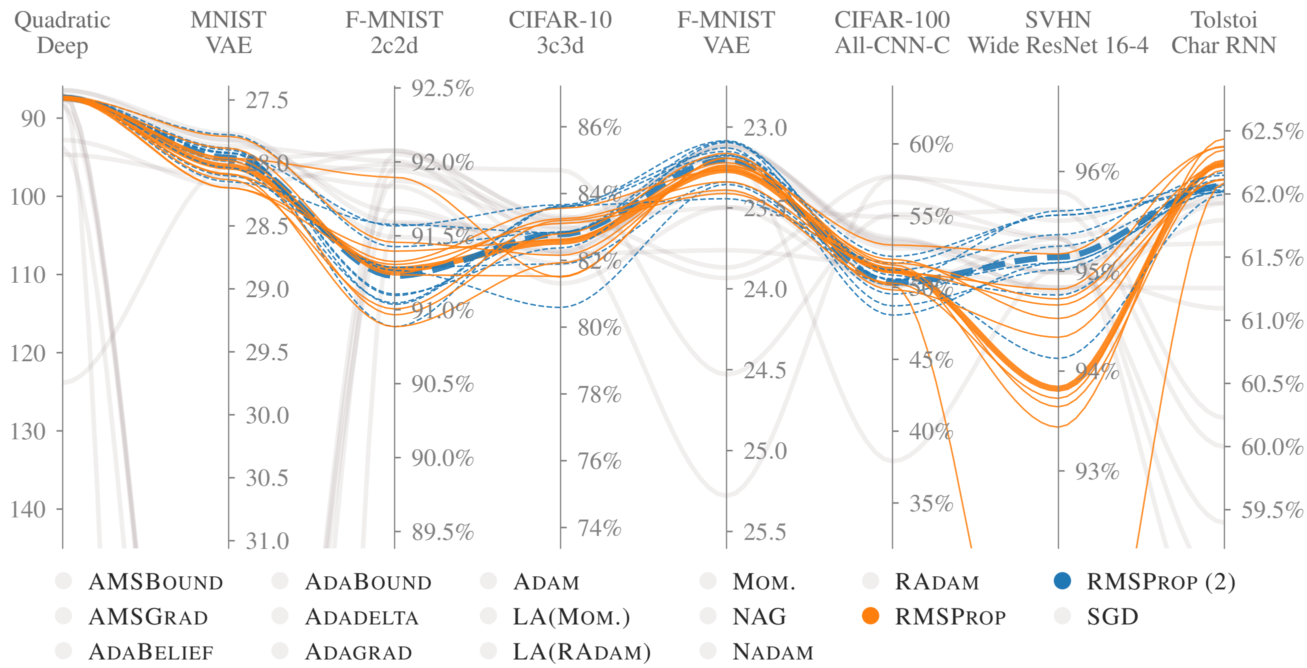


Figure 6: Mean test set performance of all 10 seeds of RMSProp (—) on all eight optimization problems using the *small budget* for tuning and *no learning rate schedule*. The mean is shown with a thicker line. We repeated the full tuning process on all eight problems using different random seeds, which is shown in dashed lines blue (- -). The mean performance of all other optimizers is shown in transparent gray lines.

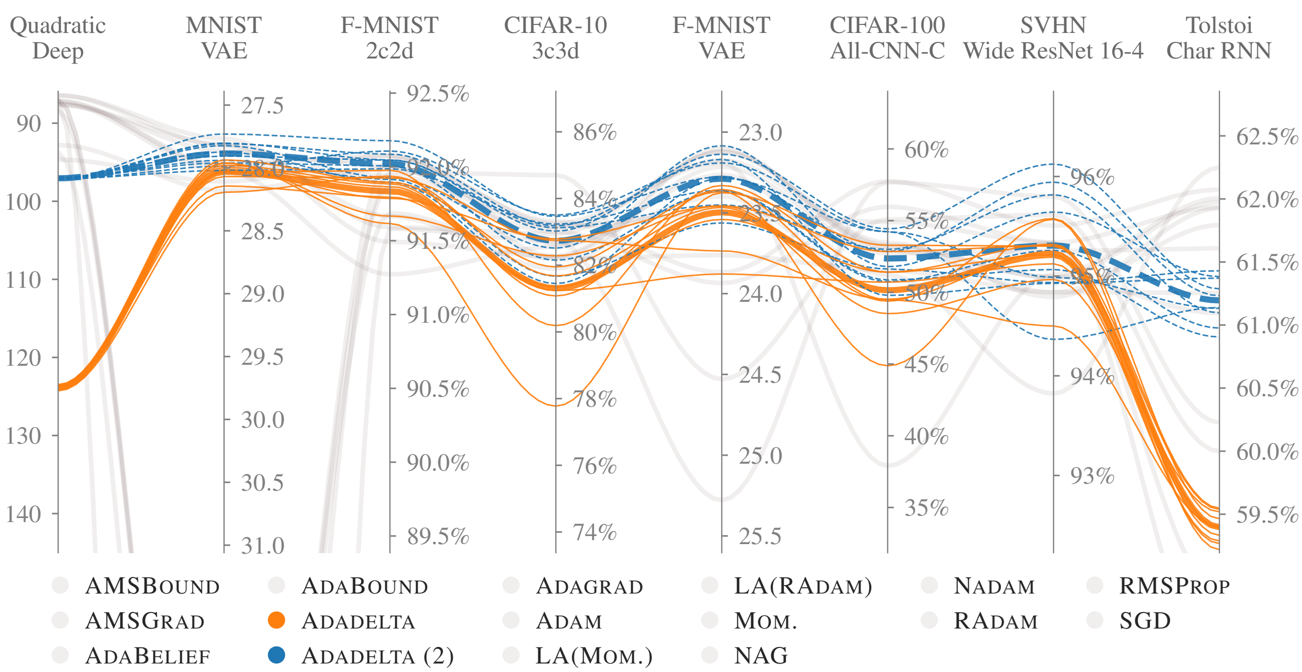


Figure 7: Mean test set performance of all 10 seeds of AdaDelta (—) on all eight optimization problems using the *small budget* for tuning and *no learning rate schedule*. The mean is shown with a thicker line. We repeated the full tuning process on all eight problems using different random seeds, which is shown in dashed lines blue (- -). The mean performance of all other optimizers is shown in transparent gray lines.

E. List of schedules selected

The schedules selected for our benchmark are illustrated in Figure 8. All learning rate schedules are multiplied by the initial learning rate found via tuning or picked as the default choice.

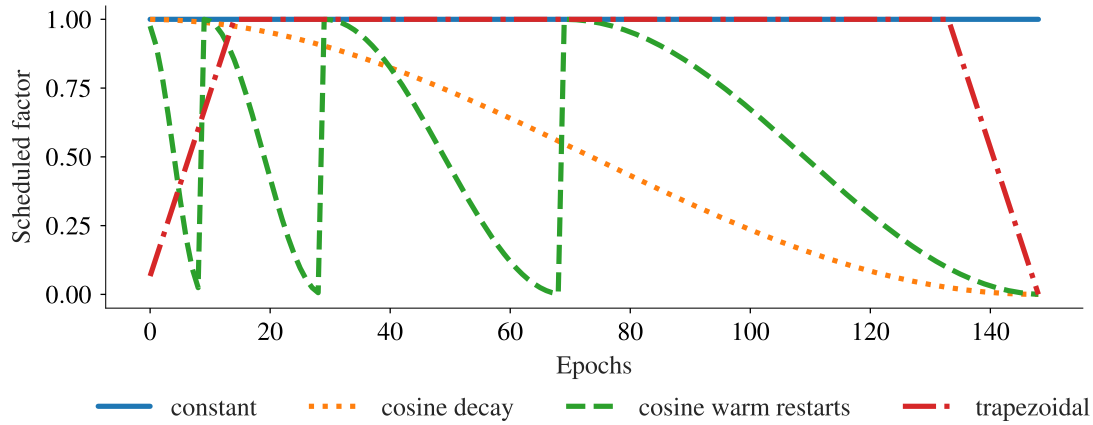


Figure 8: Illustration of the selected learning rate schedules for a training duration of 150 epochs.

We use a *cosine decay* (Loshchilov & Hutter, 2017) that starts at 1 and decays in the form of a half period of a cosine to 0. As an example of a cyclical learning rate schedule, we test a *cosine with warm restarts* schedule with a cycle length $\Delta t = 10$ which increases by a factor of 2 after each cycle without any discount factor. Depending on the number of epochs we train our model, it is possible that training stops shortly after one of those warm restarts. Since performance typically declines shortly after increasing the learning rate, we don't report the final performance for this schedule, but instead the performance achieved after the last complete period (just before the next restart). This approach is suggested by the original work of Loshchilov & Hutter (2017). However, we still use the final performance while tuning.

A representation of a schedule including warm-up is the *trapezoidal* schedule from Xing et al. (2018). For our benchmark we set a warm-up and cool-down period of $1/10$ the training time.

F. ArXiv Mentions

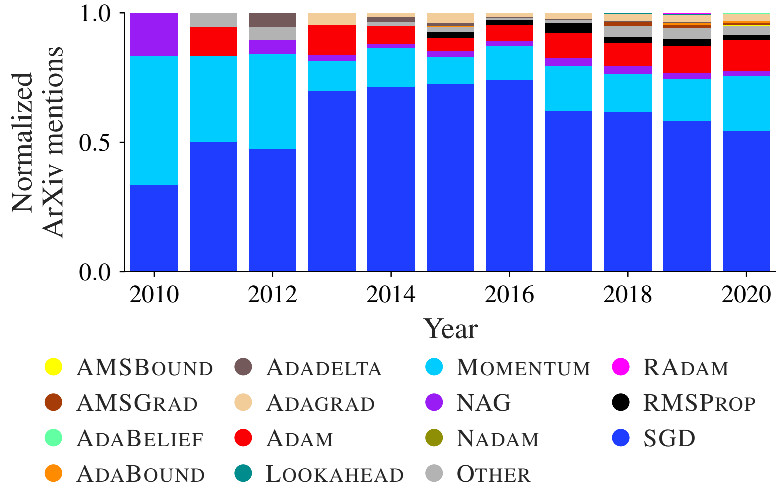


Figure 9: Percentage of times ArXiv titles and abstracts mention specific optimizer per year. This is a normalized version of Figure 1. The data for this figure is shown in Table 5.

Table 5: Mentions of each optimizer in titles and abstracts of papers on ArXiv per year. All non-selected optimizers from Table 2 in the appendix are grouped into *Other*.

Optimizer	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**
AMSBound	0	0	0	0	0	0	0	0	0	1	0
AMSGrad	0	0	0	0	0	0	0	0	7	9	11
AdaBelief	0	0	0	0	0	0	0	0	0	0	3
AdaBound	0	0	0	0	0	0	0	0	0	4	4
AdaDelta	0	0	1	0	1	2	0	1	2	3	3
AdaGrad	0	0	0	2	1	5	3	8	16	22	24
Adam	0	2	0	5	4	7	11	31	47	83	119
Lookahead	0	0	0	0	0	0	0	0	0	2	1
Momentum	3	6	7	5	9	14	23	57	76	124	205
NAG	1	0	1	1	1	3	3	11	17	18	19
Nadam	0	0	0	0	0	0	0	0	1	2	0
Other	0	1	1	0	1	3	2	4	22	34	36
RAdam	0	0	0	0	0	0	0	0	0	2	1
RMSProp	0	0	0	0	0	3	3	13	13	18	18
SGD	2	9	9	30	42	98	129	205	326	451	532

G. Improvement after tuning

When looking at Figure 2, one might realize that few diagonal entries contain negative values. Since diagonal entries reflect the intra-optimizer performance change when tuning on the respective task, this might feel quite counterintuitive at first. *In theory*, this can occur if the respective tuning distributions is chosen poorly, the tuning randomness simply got "unlucky", or we observe significantly worse results for our additional seeds (see Figure 5).

If we compare Figures 10 and 11 to Figures 12 and 13 we can see most negative diagonal entries vanish or at least diminish in magnitude. For the latter two figures we allow for more tuning runs and only consider the seed that has been used for this tuning process. The fact that the effect of negative diagonal entries reduces is an indication that they mostly result from the two latter reasons mentioned.

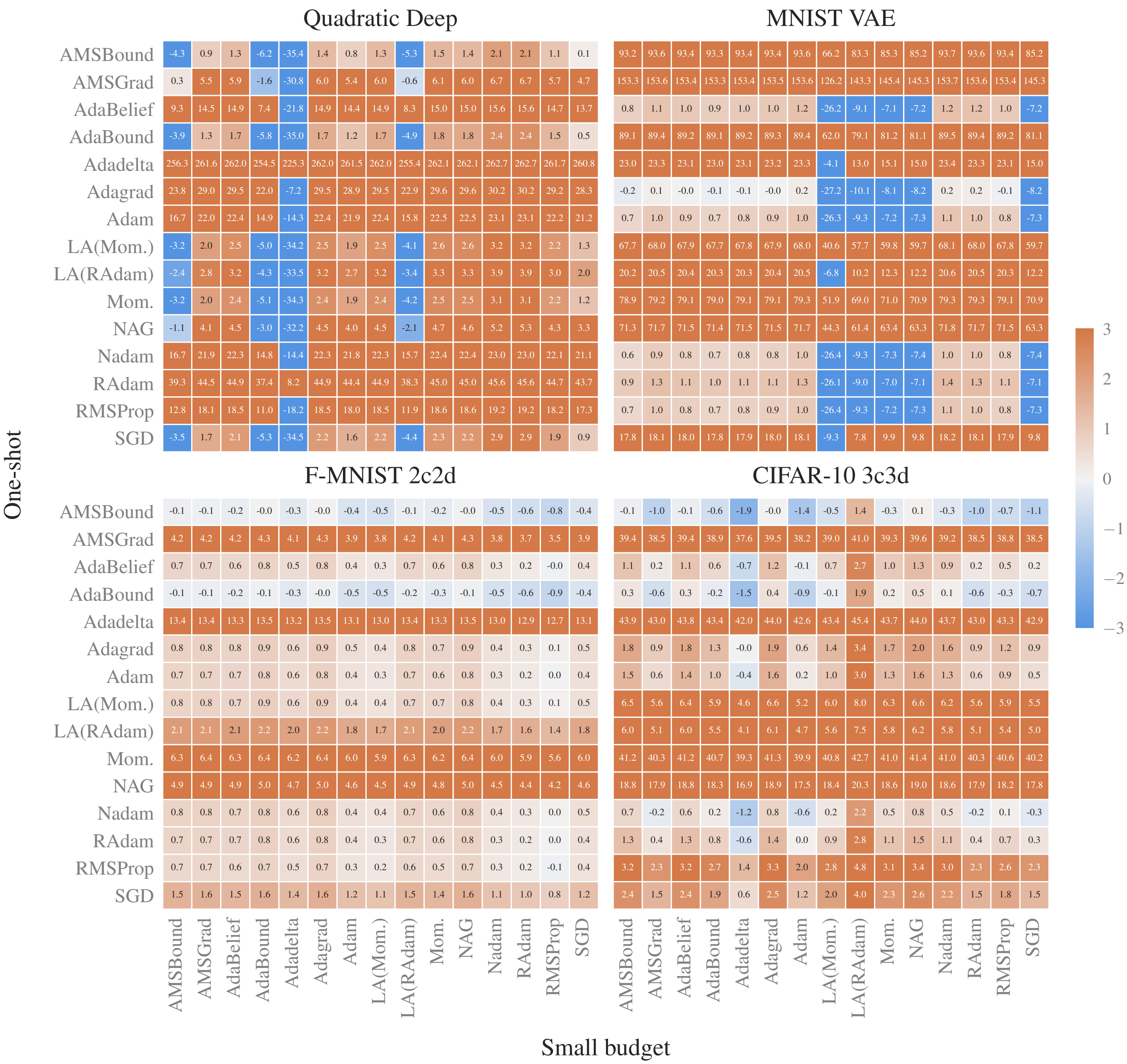


Figure 10: The absolute test set performance improvement after switching from any untuned optimizer (y-axis, *one-shot*) to any tuned optimizer (x-axis, *small budget*) as an average over 10 random seeds for the *constant* schedule. This is a detailed version of Figure 2 in the main text showing the first four problems.

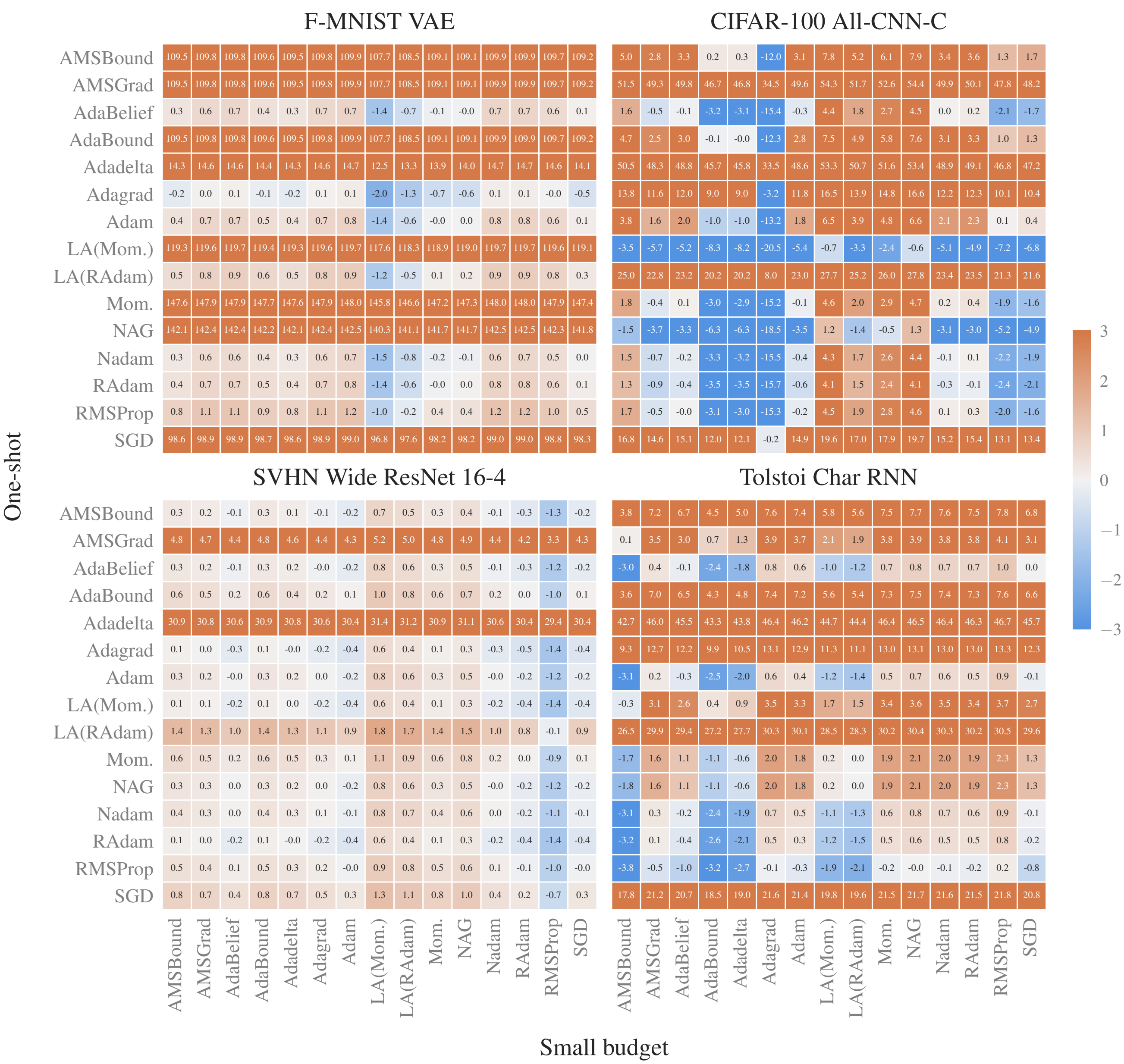


Figure 11: The absolute test set performance improvement after switching from any untuned optimizer (y-axis, *one-shot*) to any tuned optimizer (x-axis, *small budget*) as an average over 10 random seeds for the *constant* schedule. This is a detailed version of Figure 2 in the main text showing the last four problems.

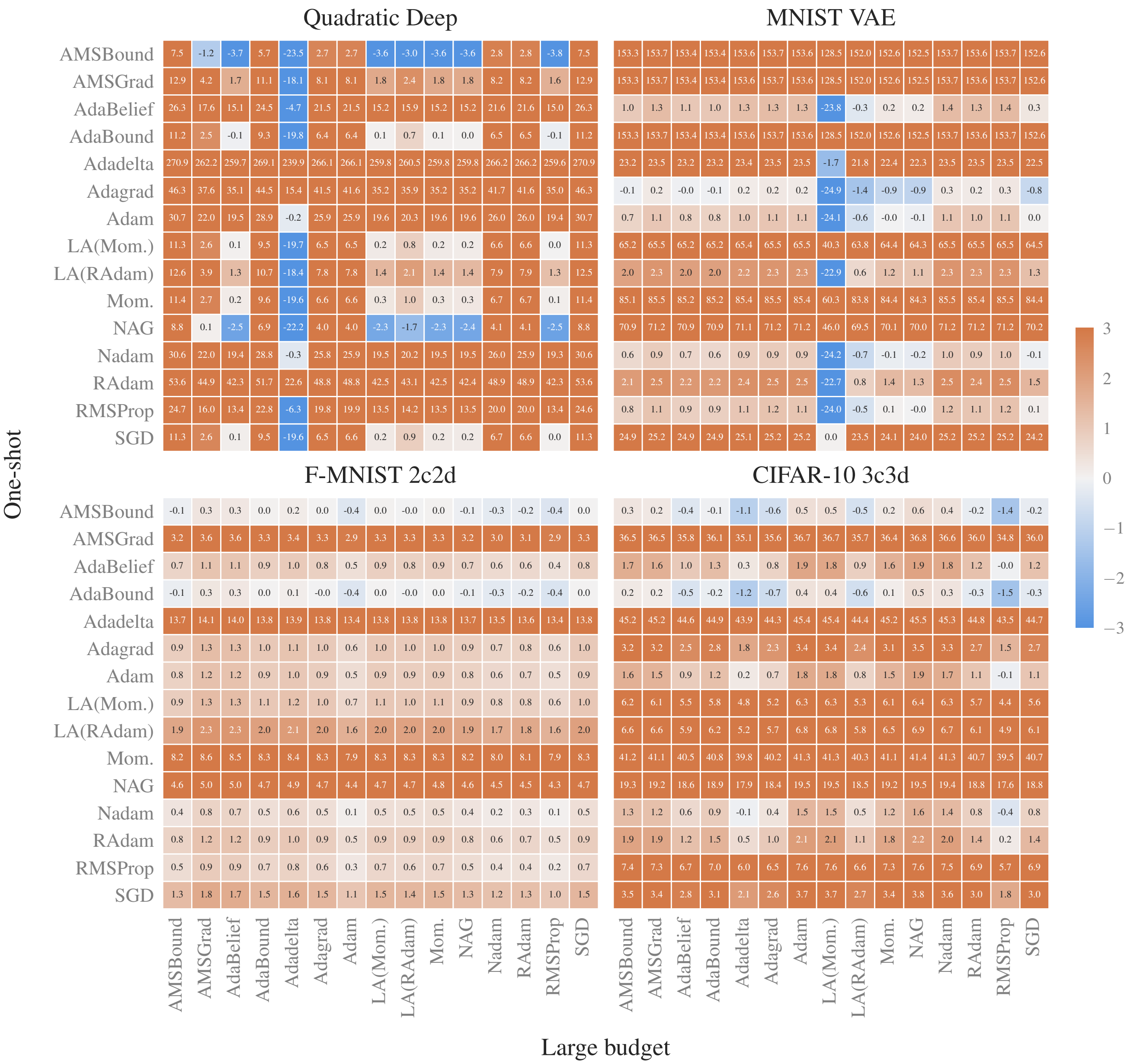


Figure 12: The absolute test set performance improvement after switching from any untuned optimizer (y-axis, *one-shot*) to any tuned optimizer (x-axis, *large budget*) for the *constant* schedule. This is structurally the same plot as Figure 10 but comparing to the *large budget* and only considering the seed that has been used for tuning.

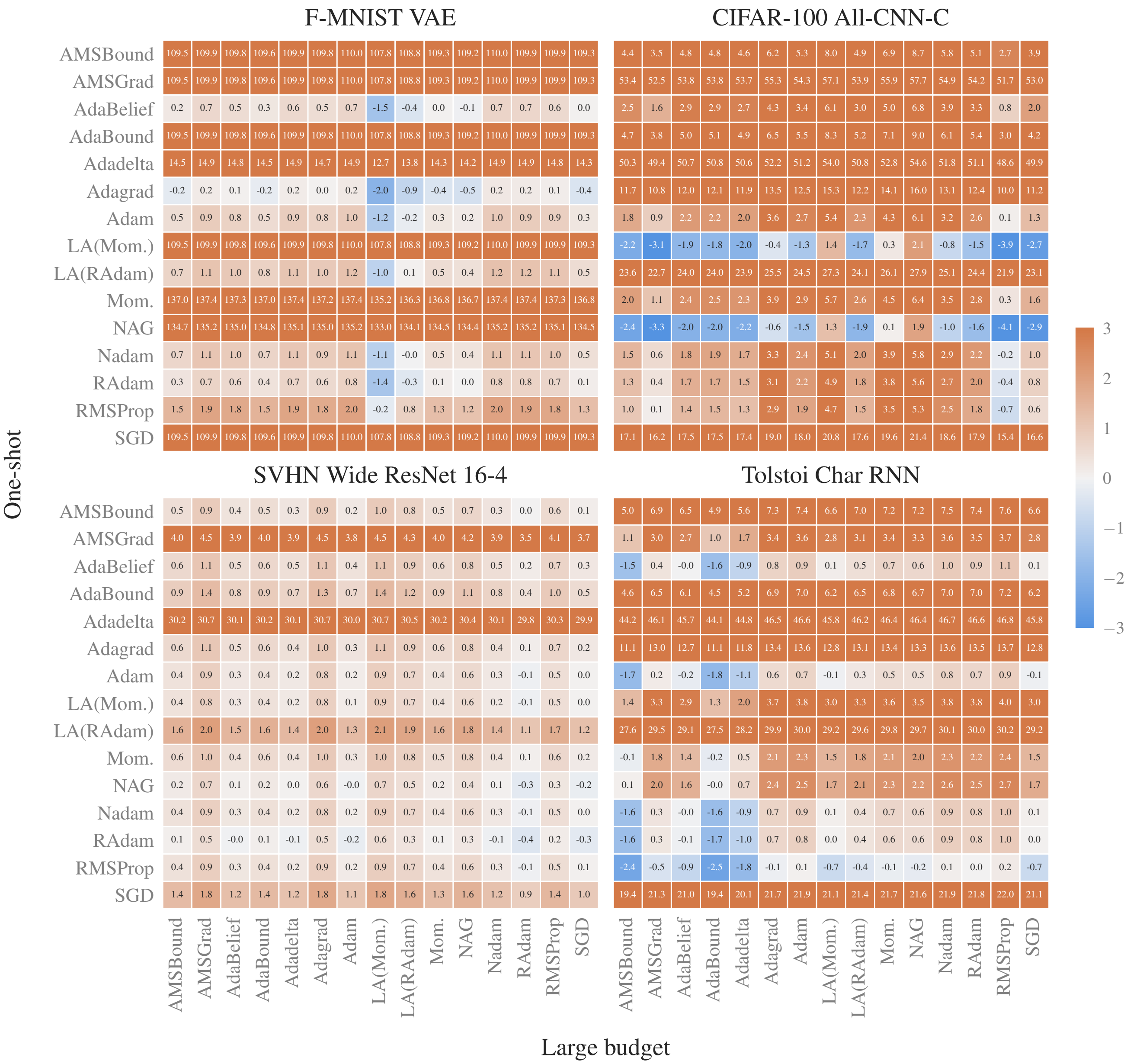


Figure 13: The absolute test set performance improvement after switching from any untuned optimizer (y-axis, *one-shot*) to any tuned optimizer (x-axis, *large budget*) for the *constant* schedule. This is structurally the same plot as Figure 11 but comparing to the *large budget* and only considering the seed that has been used for tuning.

H. Optimizer performance across test problems

Similarly to Figure 4, we show the corresponding plots for the *small budget* with *no learning rate schedule* in Figure 14 and the *medium budget* with the *cosine* and *trapezoidal learning rate schedule* in Figures 15 and 16. Additionally, in Figure 17 we show the same setting as Figure 4 but showing the training loss instead of the test loss/accuracy.

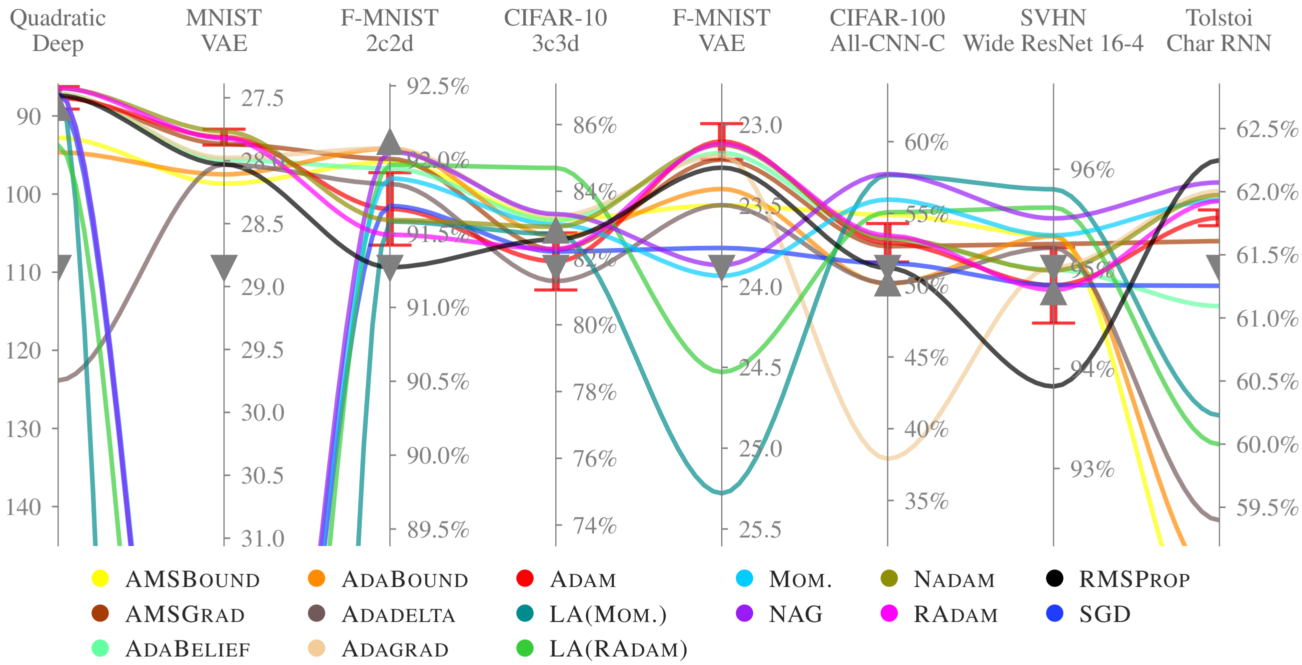


Figure 14: Mean test set performance over 10 random seeds of all tested optimizers on all eight optimization problems using the *small budget* for tuning and *no learning rate schedule*. One standard deviation for the tuned Adam optimizer is shown with a red error bar (I). The performance of the untuned versions of Adam (▼) and AdaBound (▲) are marked for reference. Note, the upper bound of each axis represents the best performance achieved in the benchmark, while the lower bound is chosen in relation to the performance of Adam with default parameters. Tabular version available in the Appendix as Table 7.

The high-level trends mentioned in Section 3 also hold for the smaller tuning budget in Figure 14. Namely, taking the winning optimizer for several untuned algorithms (here marked for Adam and AdaBound) will result in a decent performance in most problems with much less effort. Adding a tuned version Adam (or variants thereof) to this selection would result in a very competitive performance. The absolute top-performance however, is achieved by changing optimizers across different problems.

Note, although the *medium budget* is a true superset of the *small budget* it is not given that it will always perform better. Our tuning procedure guarantees that the *validation* performance on the seed that has been used for tuning is as least as good on the medium budget than on the small budget. But due to averaging over multiple seeds and reporting *test* performance instead of *validation* performance, this hierarchy is no longer guaranteed. We discuss the possible effects of averaging over multiple seeds further in Appendix C.

The same high-level trends also emerge when considering the *cosine* or *trapezoidal learning rate schedule* in Figures 15 and 16. We can also see that the top performance in general increase when adding a schedule (cf. Figure 4 and Figure 16).

Comparing Figure 4 and Figure 17 we can assess the generalization performance of the optimization method not only to an unseen test set, but also to a different performance metric (accuracy instead of loss). Again, the overall picture of varying performance across different problems remains consistent when considering the training loss performance. Similarly to the figures showing test set performance we cannot identify a clear winner, although Adam ands its variants, such as RAdam perform near the top consistently. Note that while Figure 17 shows the training loss, the optimizers have still be tuned to achieve the best validation performance (i.e. accuracy if available, else the loss).

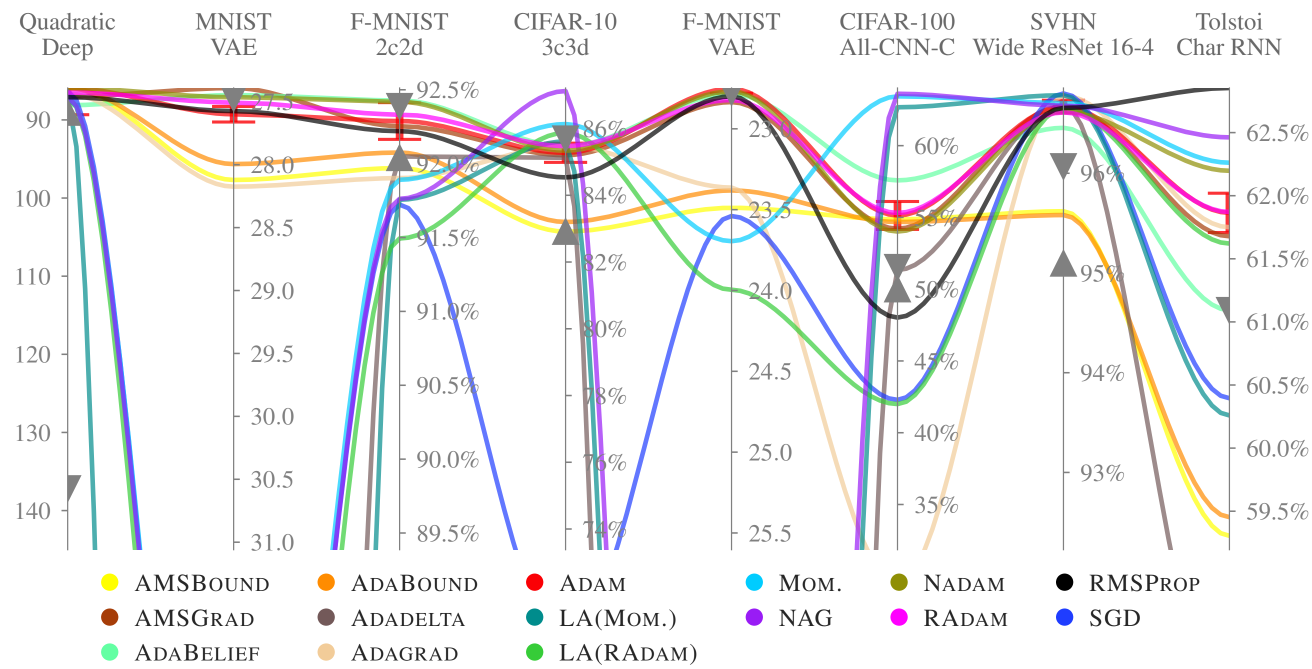


Figure 15: Mean test set performance over 10 random seeds of all tested optimizers on all eight optimization problems using the *medium budget* for tuning and the *cosine learning rate schedule*. One standard deviation for the tuned ADAM optimizer is shown with a red error bar (I). The performance of the untuned versions of ADAM (▼) and ADABOUND (▲) are marked for reference (this time with the *cosine* learning rate schedule). Note, the upper bound of each axis represents the best performance achieved in the benchmark, while the lower bound is chosen in relation to the performance of ADAM with default parameters (and no schedule). Tabular version available in the Appendix as Table 8.

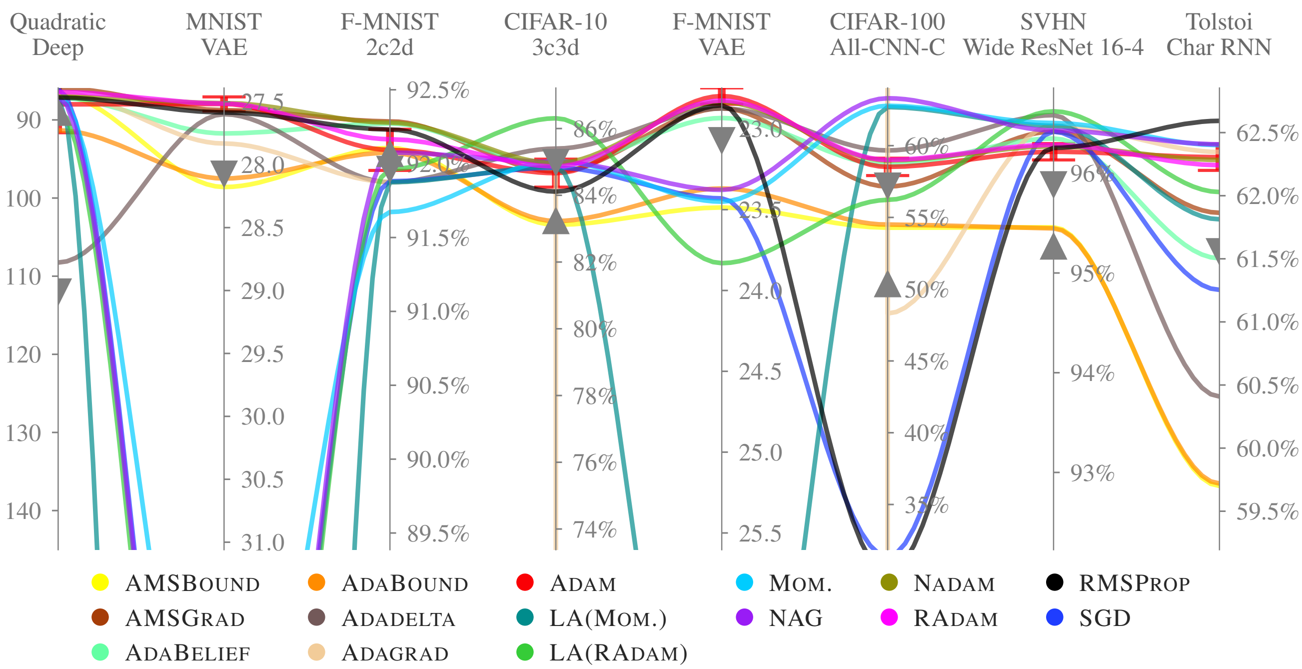


Figure 16: Mean test set performance over 10 random seeds of all tested optimizers on all eight optimization problems using the *large budget* for tuning and the *trapezoidal learning rate schedule*. One standard deviation for the tuned Adam optimizer is shown with a red error bar (I). The performance of the untuned versions of Adam (▼) and AdaBound (▲) are marked for reference (this time with the *trapezoidal* learning rate schedule). Note, the upper bound of each axis represents the best performance achieved in the benchmark, while the lower bound is chosen in relation to the performance of Adam with default parameters (and no schedule). Tabular version available in the Appendix as Table 9.

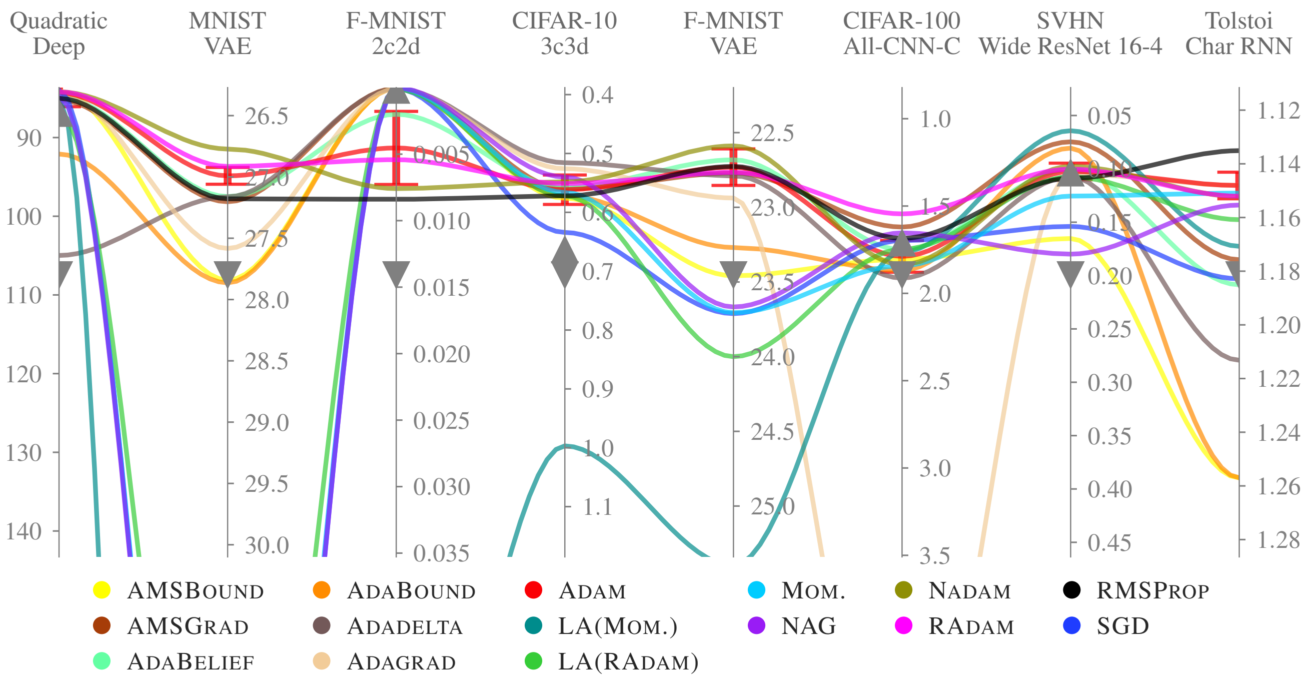


Figure 17: Mean *training* loss performance over 10 random seeds of all tested optimizers on all eight optimization problems using the *large budget* for tuning and *no learning rate schedule*. One standard deviation for the tuned Adam optimizer is shown with a red error bar (I). The performance of the untuned versions of Adam (▼) and AdaBound (▲) are marked for reference. Note, the upper bound of each axis represents the best performance achieved in the benchmark, while the lower bound is chosen in relation to the performance of Adam with default parameters (and no schedule). This figure is very similar to Figure 4, but showing the *training loss* performance instead of the *test accuracy* (or *test loss* if no accuracy is available). Tabular version available in the Appendix as Table 10.

I. Tabular version

Table 6: Tabular version of Figure 4. Mean test set performance and standard deviation over 10 random seeds of all tested optimizers on all eight optimization problems using the *large budget* for tuning and *no learning rate schedule*. For comprehensability, mean and standard deviation are rounded.

Optimizer	Quadratic Deep	MNIST VAE	F-MNIST 2c2d	CIFAR-10 3c3d	F-MNIST VAE	CIFAR-100	SVHN	Tolstoi
AMSBound	86.35 ± 3.47	28.14 ± 0.15	92.15 ± 0.13	82.99 ± 0.78	23.55 ± 0.18	54.64 ± 1.33	95.31 ± 0.31	59.70 ± 0.16
AMSGrad	87.64 ± 1.00	27.85 ± 0.07	**92.26** ± 0.16	83.42 ± 0.65	23.11 ± 0.10	52.34 ± 1.03	95.58 ± 0.31	61.52 ± 0.13
AdaBelief	87.17 ± 0.03	28.01 ± 0.06	92.06 ± 0.24	82.85 ± 0.59	23.22 ± 0.08	53.76 ± 1.35	95.09 ± 0.30	61.26 ± 0.17
AdaBound	94.66 ± 6.25	28.14 ± 0.13	92.03 ± 0.13	83.39 ± 0.53	23.38 ± 0.09	54.77 ± 0.94	95.40 ± 0.29	59.73 ± 0.20
AdaDelta	106.95 ± 0.14	27.87 ± 0.10	92.07 ± 0.11	83.34 ± 0.74	23.18 ± 0.13	53.18 ± 2.48	95.30 ± 0.60	60.54 ± 0.15
AdaGrad	86.70 ± 1.99	28.04 ± 0.29	92.05 ± 0.17	83.08 ± 0.41	23.16 ± 0.04	43.63 ± 21.35	95.34 ± 0.49	62.01 ± 0.10
Adam	86.58 ± 1.95	27.77 ± 0.03	91.69 ± 0.16	82.95 ± 0.70	23.06 ± 0.10	54.84 ± 0.65	94.84 ± 0.30	61.97 ± 0.12
LA(Mom.)	87.17 ± 0.07	52.86 ± 0.84	91.74 ± 0.19	74.01 ± 3.70	25.37 ± 0.35	57.32 ± 0.80	**95.82** ± 0.11	61.44 ± 0.17
LA(RAdam)	89.03 ± 0.87	34.26 ± 9.37	92.05 ± 0.16	83.00 ± 0.64	24.04 ± 0.25	54.92 ± 0.97	95.67 ± 0.11	61.73 ± 0.10
Momentum	87.04 ± 0.02	36.00 ± 11.09	91.87 ± 0.12	83.16 ± 0.56	23.86 ± 0.15	56.21 ± 0.67	95.37 ± 0.27	61.97 ± 0.12
NAG	87.08 ± 0.02	36.16 ± 10.99	91.87 ± 0.12	83.30 ± 0.88	23.85 ± 0.22	**57.85** ± 0.77	95.28 ± 0.23	61.74 ± 0.12
NAdam	86.45 ± 1.94	**27.73** ± 0.09	91.75 ± 0.42	**83.58** ± 0.45	**23.00** ± 0.07	53.44 ± 1.27	95.00 ± 0.25	62.01 ± 0.11
RAdam	86.43 ± 1.93	27.81 ± 0.06	91.63 ± 0.24	82.85 ± 0.52	23.10 ± 0.11	53.98 ± 1.00	94.83 ± 0.38	61.98 ± 0.13
RMSProp	87.38 ± 0.12	27.86 ± 0.08	91.79 ± 0.36	82.16 ± 0.65	23.11 ± 0.08	52.16 ± 0.99	95.25 ± 0.34	**62.24** ± 0.07
SGD	**86.29** ± 3.44	36.17 ± 10.97	91.80 ± 0.23	82.64 ± 0.91	23.83 ± 0.22	50.58 ± 1.49	95.11 ± 0.31	61.29 ± 0.14

Table 7: Tabular version of Figure 14. Mean test set performance and standard deviation over 10 random seeds of all tested optimizers on all eight optimization problems using the *small budget* for tuning and *no learning rate schedule*. For comprehensability, mean and standard deviation are rounded.

Optimizer	Quadratic Deep	MNIST VAE	F-MNIST 2c2d	CIFAR-10 3c3d	F-MNIST VAE	CIFAR-100	SVHN	Tolstoi
AMSBound	92.80 ± 5.99	28.18 ± 0.14	91.99 ± 0.10	83.15 ± 0.65	23.50 ± 0.11	54.91 ± 0.54	95.33 ± 0.17	58.25 ± 0.19
AMSGrad	87.58 ± 0.71	27.87 ± 0.08	92.01 ± 0.09	82.25 ± 0.54	23.21 ± 0.06	52.71 ± 0.97	95.25 ± 0.21	61.61 ± 0.14
AdaBelief	87.18 ± 0.03	27.99 ± 0.06	91.94 ± 0.33	83.13 ± 0.60	23.17 ± 0.07	53.17 ± 1.15	94.99 ± 0.31	61.09 ± 0.09
AdaBound	94.66 ± 6.25	28.11 ± 0.09	**92.08** ± 0.20	82.64 ± 1.03	23.40 ± 0.06	50.10 ± 16.39	95.33 ± 0.16	58.88 ± 0.16
AdaDelta	123.86 ± 0.24	28.03 ± 0.08	91.84 ± 0.11	81.31 ± 1.40	23.50 ± 0.17	50.14 ± 2.29	95.21 ± 0.29	59.40 ± 0.11
AdaGrad	87.14 ± 1.02	27.98 ± 0.16	**92.08** ± 0.23	83.25 ± 0.51	23.19 ± 0.08	37.90 ± 24.22	95.02 ± 0.21	62.01 ± 0.11
Adam	87.68 ± 1.44	27.81 ± 0.06	91.67 ± 0.25	81.90 ± 0.86	**23.10** ± 0.11	52.96 ± 1.34	94.84 ± 0.38	61.79 ± 0.06
LA(Mom.)	87.16 ± 0.06	55.20 ± 0.86	91.58 ± 0.15	82.72 ± 1.24	25.28 ± 0.23	57.68 ± 0.60	**95.80** ± 0.10	60.23 ± 0.26
LA(RAdam)	93.75 ± 3.15	38.11 ± 9.73	91.97 ± 0.22	**84.70** ± 0.30	24.53 ± 0.15	55.09 ± 0.98	95.62 ± 0.19	60.00 ± 0.11
Momentum	87.03 ± 0.02	36.08 ± 11.04	91.87 ± 0.16	83.00 ± 0.71	23.93 ± 0.30	55.96 ± 0.92	95.34 ± 0.23	61.93 ± 0.10
NAG	87.08 ± 0.02	36.18 ± 10.97	92.05 ± 0.13	83.32 ± 0.57	23.87 ± 0.33	**57.75** ± 0.71	95.51 ± 0.21	62.07 ± 0.10
NAdam	86.45 ± 1.94	**27.77** ± 0.06	91.59 ± 0.25	82.94 ± 0.61	23.12 ± 0.06	53.30 ± 0.90	94.99 ± 0.18	61.97 ± 0.08
RAdam	**86.43** ± 1.93	27.82 ± 0.06	91.49 ± 0.40	82.27 ± 0.53	23.12 ± 0.07	53.47 ± 0.86	94.79 ± 0.38	61.93 ± 0.14
RMSProp	87.40 ± 0.14	28.03 ± 0.13	91.27 ± 0.28	82.56 ± 0.71	23.26 ± 0.08	51.20 ± 0.89	93.82 ± 1.64	**62.25** ± 0.12
SGD	88.37 ± 3.55	36.18 ± 10.96	91.69 ± 0.15	82.20 ± 1.32	23.76 ± 0.25	51.53 ± 1.37	94.84 ± 0.56	61.25 ± 0.12

Table 8: Tabular version of Figure 15. Mean test set performance and standard deviation over 10 random seeds of all tested optimizers on all eight optimization problems using the *medium budget* for tuning and the *cosine learning rate schedule*. For comprehensability, mean and standard deviation are rounded.

Optimizer	**Quadratic Deep**	**MNIST VAE**	**F-MNIST 2c2d**	**CIFAR-10 3c3d**	**F-MNIST VAE**	**CIFAR-100**	**SVHN**	**Tolstoi**
AMSBound	85.94 ± 3.41	28.12 ± 0.19	91.97 ± 0.15	82.91 ± 0.83	23.49 ± 0.07	54.87 ± 0.70	95.62 ± 0.15	59.31 ± 0.36
AMSGrad	87.00 ± 0.55	**27.39** ± 0.04	92.25 ± 0.22	85.20 ± 0.34	22.83 ± 0.06	54.21 ± 1.99	96.68 ± 0.07	61.68 ± 0.17
AdaBelief	88.12 ± 0.04	27.45 ± 0.05	**92.43** ± 0.14	85.47 ± 0.26	22.78 ± 0.04	57.58 ± 0.57	96.46 ± 0.08	61.09 ± 0.17
AdaBound	**85.92** ± 3.41	28.00 ± 0.09	92.08 ± 0.17	83.20 ± 0.62	23.38 ± 0.08	54.68 ± 0.81	95.58 ± 0.10	59.45 ± 0.36
AdaDelta	164.58 ± 0.35	58.46 ± 61.52	92.05 ± 0.08	85.12 ± 0.28	60.55 ± 49.27	51.34 ± 0.64	96.68 ± 0.05	57.77 ± 0.19
AdaGrad	86.61 ± 1.94	28.17 ± 0.27	91.90 ± 0.23	85.48 ± 0.35	23.36 ± 0.05	29.40 ± 28.41	96.78 ± 0.07	61.75 ± 0.07
Adam	**85.92** ± 3.41	27.60 ± 0.06	92.29 ± 0.12	85.27 ± 0.29	**22.75** ± 0.03	55.14 ± 0.97	96.67 ± 0.06	61.86 ± 0.16
LA(Mom.)	87.06 ± 0.02	76.78 ± 24.04	91.76 ± 0.20	85.61 ± 0.24	46.09 ± 21.85	62.67 ± 0.81	96.78 ± 0.08	60.26 ± 0.23
LA(RAdam)	87.08 ± 0.42	37.41 ± 10.15	91.49 ± 0.24	85.87 ± 0.18	24.00 ± 0.12	42.00 ± 27.55	96.65 ± 0.09	61.62 ± 0.16
Momentum	87.06 ± 0.02	36.33 ± 10.85	91.89 ± 0.12	86.13 ± 0.19	23.70 ± 0.18	63.43 ± 0.56	96.71 ± 0.05	62.26 ± 0.13
NAG	87.06 ± 0.02	36.53 ± 10.71	91.76 ± 0.13	**87.12** ± 0.19	41.41 ± 21.65	**63.61** ± 0.46	96.68 ± 0.08	62.46 ± 0.10
NAdam	85.93 ± 3.41	27.46 ± 0.10	92.42 ± 0.12	85.34 ± 0.34	22.77 ± 0.07	54.02 ± 0.71	96.62 ± 0.07	62.20 ± 0.12
RAdam	86.49 ± 1.94	27.51 ± 0.05	92.33 ± 0.10	85.47 ± 0.36	22.82 ± 0.08	55.31 ± 0.86	96.61 ± 0.07	61.87 ± 0.19
RMSProp	87.09 ± 0.01	27.57 ± 0.05	92.22 ± 0.18	84.54 ± 0.25	22.80 ± 0.04	48.02 ± 15.69	96.65 ± 0.06	**62.85** ± 0.06
SGD	86.30 ± 3.41	36.47 ± 10.76	91.72 ± 0.21	70.50 ± 30.76	23.54 ± 0.13	42.29 ± 27.05	**96.80** ± 0.08	60.40 ± 0.11

Table 9: Tabular version of Figure 16. Mean test set performance and standard deviation over 10 random seeds of all tested optimizers on all eight optimization problems using the *large budget* for tuning and *trapezoidal learning rate schedule*. For comprehensability, mean and standard deviation are rounded.

Optimizer	**Quadratic Deep**	**MNIST VAE**	**F-MNIST 2c2d**	**CIFAR-10 3c3d**	**F-MNIST VAE**	**CIFAR-100**	**SVHN**	**Tolstoi**
AMSBound	86.78 ± 2.04	28.18 ± 0.19	92.11 ± 0.16	83.11 ± 0.84	23.49 ± 0.11	54.28 ± 1.23	95.46 ± 0.21	59.70 ± 0.14
AMSGrad	**85.94** ± 3.42	27.57 ± 0.06	**92.29** ± 0.12	84.71 ± 0.31	22.87 ± 0.06	57.15 ± 0.89	96.42 ± 0.06	61.86 ± 0.14
AdaBelief	87.19 ± 0.02	27.75 ± 0.05	92.27 ± 0.10	84.90 ± 0.32	22.93 ± 0.07	58.66 ± 0.50	96.35 ± 0.07	61.50 ± 0.15
AdaBound	91.34 ± 5.60	28.11 ± 0.09	92.08 ± 0.14	83.23 ± 0.58	23.37 ± 0.05	54.50 ± 1.23	95.45 ± 0.18	59.72 ± 0.17
AdaDelta	108.26 ± 0.14	27.60 ± 0.08	91.87 ± 0.20	85.40 ± 0.17	22.87 ± 0.08	59.67 ± 0.38	96.58 ± 0.07	60.41 ± 0.11
AdaGrad	86.51 ± 1.95	27.83 ± 0.15	91.88 ± 0.12	84.84 ± 0.23	7e23 ± 2e24	48.31 ± 23.66	96.48 ± 0.10	62.35 ± 0.16
Adam	88.01 ± 3.63	27.52 ± 0.06	92.09 ± 0.14	84.66 ± 0.42	**22.80** ± 0.05	58.52 ± 0.61	96.22 ± 0.08	62.31 ± 0.10
LA(Mom.)	87.12 ± 0.02	52.89 ± 0.00	91.87 ± 0.17	84.85 ± 0.60	28.24 ± 13.23	62.69 ± 0.42	96.48 ± 0.10	61.81 ± 0.17
LA(RAdam)	88.67 ± 1.24	36.14 ± 10.99	91.96 ± 0.14	**86.31** ± 0.25	23.83 ± 0.14	56.22 ± 18.42	**96.62** ± 0.08	62.03 ± 0.14
Momentum	87.06 ± 0.02	33.77 ± 9.62	91.67 ± 0.22	85.02 ± 0.30	23.45 ± 0.22	62.78 ± 0.34	96.50 ± 0.08	62.40 ± 0.08
NAG	87.06 ± 0.02	35.80 ± 11.20	92.08 ± 0.16	85.00 ± 0.44	23.38 ± 0.16	**63.30** ± 0.31	96.43 ± 0.11	62.41 ± 0.09
NAdam	87.03 ± 3.66	**27.51** ± 0.08	92.28 ± 0.11	84.96 ± 0.37	22.83 ± 0.08	58.96 ± 0.77	96.27 ± 0.10	62.28 ± 0.11
RAdam	86.43 ± 1.93	**27.51** ± 0.05	92.17 ± 0.17	84.86 ± 0.32	22.83 ± 0.07	59.01 ± 0.73	96.29 ± 0.09	62.24 ± 0.13
RMSProp	87.14 ± 0.03	27.58 ± 0.07	92.23 ± 0.13	84.11 ± 0.16	22.85 ± 0.05	30.15 ± 29.15	96.25 ± 0.09	**62.59** ± 0.11
SGD	86.05 ± 3.40	35.71 ± 11.26	91.88 ± 0.17	84.83 ± 0.27	23.43 ± 0.19	31.36 ± 30.38	96.42 ± 0.07	61.25 ± 0.11

Table 10: Tabular version of Figure 17. Mean *training* set performance and standard deviation over 10 random seeds of all tested optimizers on all eight optimization problems using the *large budget* for tuning and *no learning rate schedule*. For comprehensability, mean and standard deviation are rounded.

Optimizer	**Quadratic Deep**	**MNIST VAE**	**F-MNIST 2c2d**	**CIFAR-10 3c3d**	**F-MNIST VAE**	**CIFAR-100**	**SVHN**	**Tolstoi**
AMSBound	84.10 ± 3.34	27.84 ± 0.15	**0.00** ± 0.00	0.58 ± 0.02	23.46 ± 0.22	1.80 ± 0.09	0.17 ± 0.00	1.26 ± 0.01
AMSGrad	85.24 ± 1.21	27.20 ± 0.09	**0.00** ± 0.00	0.56 ± 0.01	22.77 ± 0.10	1.62 ± 0.11	0.07 ± 0.00	1.18 ± 0.01
AdaBelief	84.87 ± 0.30	27.16 ± 0.07	**0.00** ± 0.00	0.56 ± 0.02	22.68 ± 0.07	1.75 ± 0.11	0.10 ± 0.00	1.19 ± 0.01
AdaBound	92.10 ± 5.64	27.86 ± 0.17	**0.00** ± 0.00	0.57 ± 0.02	23.27 ± 0.14	1.87 ± 0.09	0.08 ± 0.01	1.26 ± 0.01
Adadelta	104.99 ± 0.30	27.16 ± 0.12	**0.00** ± 0.00	**0.52** ± 0.01	22.79 ± 0.15	1.91 ± 0.17	0.11 ± 0.01	1.21 ± 0.01
Adagrad	84.40 ± 1.73	27.58 ± 0.34	**0.00** ± 0.00	0.53 ± 0.02	22.94 ± 0.06	5.25 ± 6.85	0.10 ± 0.01	1.15 ± 0.01
Adam	84.33 ± 1.76	26.99 ± 0.07	**0.00** ± 0.00	0.56 ± 0.03	22.73 ± 0.12	1.79 ± 0.09	0.10 ± 0.01	1.15 ± 0.00
LA(Mom.)	84.85 ± 0.30	52.85 ± 0.74	0.06 ± 0.02	1.00 ± 0.16	25.40 ± 0.39	1.76 ± 0.06	**0.06** ± 0.00	1.17 ± 0.01
LA(RAdam)	86.68 ± 1.10	34.33 ± 9.29	**0.00** ± 0.00	0.57 ± 0.02	24.00 ± 0.26	1.75 ± 0.08	0.11 ± 0.00	1.16 ± 0.00
Momentum	84.77 ± 0.30	35.98 ± 11.06	**0.00** ± 0.00	0.57 ± 0.02	23.71 ± 0.13	1.84 ± 0.07	0.13 ± 0.00	1.15 ± 0.01
NAG	84.77 ± 0.30	36.15 ± 10.96	**0.00** ± 0.00	0.54 ± 0.02	23.67 ± 0.22	1.65 ± 0.03	0.18 ± 0.00	1.16 ± 0.01
Nadam	84.19 ± 1.74	**26.77** ± 0.12	0.01 ± 0.01	0.55 ± 0.02	**22.59** ± 0.08	1.84 ± 0.08	0.10 ± 0.00	1.15 ± 0.01
RAdam	84.18 ± 1.74	26.91 ± 0.10	0.01 ± 0.00	0.55 ± 0.02	22.77 ± 0.08	**1.54** ± 0.10	0.10 ± 0.00	1.15 ± 0.01
RMSProp	85.02 ± 0.32	27.18 ± 0.13	0.01 ± 0.01	0.57 ± 0.02	22.73 ± 0.08	1.69 ± 0.13	0.11 ± 0.01	**1.14** ± 0.00
SGD	**83.95** ± 3.25	36.15 ± 10.95	**0.00** ± 0.00	0.63 ± 0.02	23.71 ± 0.23	1.70 ± 0.10	0.15 ± 0.01	1.18 ± 0.00

Appendix References

Aitchison, L. Bayesian filtering unifies adaptive and non-adaptive neural network optimization methods. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Almeida, D., Winter, C., Tang, J., and Zaremba, W. A Generalizable Approach to Learning Optimizers. *arXiv preprint:* `2106.00958`, 2021.

Anil, R., Gupta, V., Koren, T., and Singer, Y. Memory Efficient Adaptive Optimization. In *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.

Anil, R., Gupta, V., Koren, T., Regan, K., and Singer, Y. Second Order Optimization Made Practical. *arXiv preprint:* `2002.09018`, 2020.

Ayadi, I. and Turinici, G. Stochastic Runge-Kutta methods and adaptive SGD-G2 stochastic gradient descent. *arXiv preprint:* `2002.09304`, 2020.

Bae, K., Ryu, H., and Shin, H. Does Adam optimizer keep close to the optimal point?. *arXiv preprint:* `1911.00289`, 2019.

Bahrami, D. and Zadeh, S. P. Gravity Optimizer: a Kinematic Approach on Optimization in Deep Learning. *arXiv preprint:* `2101.09192`, 2021.

Bai, J. and Zhang, J. BGADAM: Boosting based Genetic-Evolutionary ADAM for Convolutional Neural Network Optimization. *arXiv preprint:* `1908.08015`, 2019.

Balles, L. and Hennig, P. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. In *35th International Conference on Machine Learning, ICML*, 2018.

Bello, I., Zoph, B., Vasudevan, V., and Le, Q. V. Neural Optimizer Search with Reinforcement Learning. In *34th International Conference on Machine Learning, ICML*, 2017.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. SIGNSGD: Compressed Optimisation for Non-Convex Problems. In *35th International Conference on Machine Learning, ICML*, 2018.

Berrada, L., Zisserman, A., and Kumar, M. P. Training Neural Networks for and by Interpolation. In *37th International Conference on Machine Learning, ICML*, 2020.

Borysenko, O. and Byshkin, M. CoolMomentum: A Method for Stochastic Optimization by Langevin Dynamics with Simulated Annealing. *arXiv preprint:* `2005.14605`, 2020.

Botev, A., Ritter, H., and Barber, D. Practical Gauss-Newton Optimisation for Deep Learning. In *34th International Conference on Machine Learning, ICML*, 2017.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*. Springer, 2012.

Castera, C., Bolte, J., Févotte, C., and Pauwels, E. Second-order step-size tuning of SGD for non-convex optimization. *arXiv preprint:* `2103.03570`, 2021.

Chae, Y., Wilke, D. N., and Kafka, D. GOALS: Gradient-Only Approximations for Line Searches Towards Robust and Consistent Training of Deep Neural Networks. *arXiv preprint:* `2105.10915`, 2021.

Chakrabarti, K. and Chopra, N. Generalized AdaGrad (G-AdaGrad) and Adam: A State-Space Perspective. *arXiv preprint:* `2106.00092`, 2021.

Chen, C., Choi, J., Brand, D., Agrawal, A., Zhang, W., and Gopalakrishnan, K. AdaComp: Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *32nd AAAI Conference on Artificial Intelligence, AAAI*, 2018.

Chen, J., Zhou, D., Tang, Y., Yang, Z., Cao, Y., and Gu, Q. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. In *29th International Joint Conference on Artificial Intelligence, IJCAI*, 2020a.

Chen, R. T. Q., Choi, D., Balles, L., Duvenaud, D., and Hennig, P. Self-Tuning Stochastic Optimization with Curvature-Aware Gradient Filtering. *arXiv preprint:* `2011.04803`, 2020b.

Chen, T., Guo, Z., Sun, Y., and Yin, W. CADA: Communication-Adaptive Distributed Adam. In *24th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2021.

Chen, X., Liu, S., Sun, R., and Hong, M. On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization. In *7th International Conference on Learning Representations, ICLR*, 2019a.

Chen, Y., Jing, H., Zhao, W., Liu, Z., Li, O., Qiao, L., Xue, W., Fu, H., and Yang, G. An Adaptive Remote Stochastic Gradient Method for Training Neural Networks. *arXiv preprint:* 1905.01422, 2019b.

Chen, Y., Jing, H., Zhao, W., Liu, Z., Qiao, L., Xue, W., Fu, H., and Yang, G. NAMSG: An Efficient Method For Training Neural Networks. *arXiv preprint:* 1905.01422, 2019c.

Chen, Z. and Zhou, Y. Momentum with Variance Reduction for Nonconvex Composition Optimization. *arXiv preprint:* 2005.07755, 2020.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. On Empirical Comparisons of Optimizers for Deep Learning. *arXiv preprint:* 1910.05446, 2019.

Daley, B. and Amato, C. Expectigrad: Fast Stochastic Optimization with Robust Convergence Properties. *arXiv preprint:* 2010.01356, 2020.

de Roos, F., Jidling, C., Wills, A., Schön, T., and Hennig, P. A Probabilistically Motivated Learning Rate Adaptation for Stochastic Optimization. *arXiv preprint:* 2102.10880, 2021.

Defazio, A. and Jelassi, S. Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization. *arXiv preprint:* 2101.11075, 2021.

Dellaferrera, G., Wozniak, S., Indiveri, G., Pantazi, A., and Eleftheriou, E. Learning in Deep Neural Networks Using a Biologically Inspired Optimizer. *arXiv preprint:* 2104.11604, 2021.

Devarakonda, A., Naumov, M., and Garland, M. AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks. *arXiv preprint:* 1712.02029, 2017.

Ding, J., Ren, X., Luo, R., and Sun, X. An Adaptive and Momental Bound Method for Stochastic Learning. *arXiv preprint:* 1910.12249, 2019.

Dozat, T. Incorporating Nesterov Momentum into Adam. In *4th International Conference on Learning Representations, ICLR*, 2016.

Dubey, S. R., Chakraborty, S., Roy, S. K., Mukherjee, S., Singh, S. K., and Chaudhuri, B. B. diffGrad: An Optimization Method for Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Duchi, J., Hazan, E., and Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research, JMLR*, 12, 2011.

Eliyahu, Y. ADAS Optimzier. https://github.com/YanaiEliyahu/AdasOptimizer, 2020.

Fetterman, A. J., Kim, C. H., and Albrecht, J. SoftAdam: Unifying SGD and Adam for better stochastic gradient descent. https://openreview.net/forum?id=Skgfr1rYDH, 2019.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware Minimization for Efficiently Improving Generalization. In *9th International Conference on Learning Representations, ICLR*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Gao, K.-X., Liu, X.-L., Huang, Z.-H., Wang, M., Wang, S., Wang, Z., Xu, D., and Yu, F. Eigenvalue-corrected Natural Gradient Based on a New Approximation. *arXiv preprint:* 2011.13609, 2020.

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Nguyen, H., and Cohen, J. M. Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks. *arXiv preprint:* 1905.11286, 2019.

Goldfarb, D., Ren, Y., and Bahamou, A. Practical Quasi-Newton Methods for Training Deep Neural Networks. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv preprint:* 1706.02677, 2017.

Grankin, M. RangerLars. https://github.com/mgrankin/over9000, 2020.

Granziol, D., Wan, X., and Roberts, S. Gadam: Combining Adaptivity with Iterate Averaging Gives Greater Generalisation. *arXiv preprint:* 2003.01247, 2020.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned Stochastic Tensor Optimization. In *35th International Conference on Machine Learning, ICML*, 2018.

Hayashi, H., Koushik, J., and Neubig, G. Eve: A Gradient Based Optimization Method with Locally and Globally Adaptive Learning Rates. *arXiv preprint:* 1611.01505, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

Henriques, J. F., Ehrhardt, S., Albanie, S., and Vedaldi, A. Small Steps and Giant Leaps: Minimal Newton Solvers for Deep Learning. In *IEEE/CVF International Conference on Computer Vision, ICCV*, 2019.

Heo, B., Chun, S., Oh, S. J., Han, D., Yun, S., Uh, Y., and Ha, J.-W. AdamP: Slowing Down the Weight Norm Increase in Momentum-based Optimizers. In *7th International Conference on Learning Representations, ICLR*, 2021.

Hosseini, M. S. and Plataniotis, K. N. AdaS: Adaptive Scheduling of Stochastic Gradients. *arXiv preprint:* 2006.06587, 2020.

Howard, J. and Ruder, S. Universal Language Model Fine-tuning for Text Classification. In *56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Hu, Y., Lin, L., and Tang, S. Second-order Information in First-order Optimization Methods. *arXiv preprint:* 1912.09926, 2019.

Hu, Y., Zhang, S., Chen, X., and He, N. Biased Stochastic First-Order Methods for Conditional Stochastic Optimization and Applications in Meta Learning. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Huang, H., Wang, C., and Dong, B. Nostalgic Adam: Weighting More of the Past Gradients When Designing the Adaptive Learning Rate. In *28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

Huang, X., Zhou, H., Xu, R., Wang, Z., and Li, L. Adaptive Gradient Methods Can Be Provably Faster than SGD after Finite Epochs. *arXiv preprint:* 2006.07037, 2020.

Ida, Y., Fujiwara, Y., and Iwamura, S. Adaptive Learning Rate via Covariance Matrix Based Preconditioning for Deep Neural Networks. In *26th International Joint Conference on Artificial Intelligence, IJCAI*, 2017.

Ilboudo, W. E. L., Kobayashi, T., and Sugimoto, K. TAdam: A Robust Stochastic Gradient Optimizer. *arXiv preprint:* 2003.00179, 2020.

Jiang, Z., Balu, A., Tan, S. Y., Lee, Y. M., Hegde, C., and Sarkar, S. On Higher-order Moments in Adam. *arXiv preprint:* 1910.06878, 2019.

Jin, Y., Zhou, T., Zhao, L., Zhu, Y., Guo, C., Canini, M., and Krishnamurthy, A. AutoLRS: Automatic Learning-Rate Schedule by Bayesian Optimization on the Fly. In *9th International Conference on Learning Representations, ICLR*, 2021. URL https://openreview.net/forum?id=SlrqM9_lyju.

Johnson, T. B., Agrawal, P., Gu, H., and Guestrin, C. AdaScale SGD: A User-Friendly Algorithm for Distributed Training. In *37th International Conference on Machine Learning, ICML*, 2020.

Kafka, D. and Wilke, D. Gradient-only line searches: An Alternative to Probabilistic Line Searches. *arXiv preprint:* `1903.09383`, 2019.

Kelterborn, C., Mazur, M., and Petrenko, B. V. Gravilon: Applications of a New Gradient Descent Method to Machine Learning. *arXiv preprint:* `2008.11370`, 2020.

Keskar, N. S. and Socher, R. Improving Generalization Performance by Switching from Adam to SGD. *arXiv preprint:* `1712.07628`, 2017.

Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *35th International Conference on Machine Learning, ICML*, 2018.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Kwon, J., Kim, J., Park, H., and Choi, I. K. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. *arXiv preprint:* `2102.11600`, 2021.

Landro, N., Gallo, I., and Grassa, R. L. Mixing ADAM and SGD: a Combined Optimization Method. *arXiv preprint:* `2011.08042`, 2020.

Levy, K. Y., Yurtsever, A., and Cevher, V. Online Adaptive Methods, Universality and Acceleration. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Li, W., Zhang, Z., Wang, X., and Luo, P. AdaX: Adaptive Gradient Descent with Exponential Long Term Memory. *arXiv preprint:* `2004.09740`, 2020a.

Li, Z., Bao, H., Zhang, X., and Richtárik, P. PAGE: A Simple and Optimal Probabilistic Gradient Estimator for Nonconvex Optimization. *arXiv preprint:* `2008.10898`, 2020b.

Liu, G.-H., Chen, T., and Theodorou, E. A. Dynamic game theoretic neural optimizer. *arXiv preprint:* `2105.03788`, 2021a.

Liu, H. and Tian, X. AEGD: Adaptive Gradient Decent with Energy. *arXiv preprint:* `2010.05109`, 2020.

Liu, L. and Luo, X. A New Accelerated Stochastic Gradient Method with Momentum. *arXiv preprint:* `2006.00423`, 2020.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the Variance of the Adaptive Learning Rate and Beyond. In *8th International Conference on Learning Representations, ICLR*, 2020a.

Liu, M., Zhang, W., Orabona, F., and Yang, T. Adam$^+$: A Stochastic Method with Adaptive Variance Reduction. *arXiv preprint:* `2011.11985`, 2020b.

Liu, R., Wu, T., and Mozafari, B. Adam with Bandit Sampling for Deep Learning. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020c.

Liu, Y., Bernstein, J., Meister, M., and Yue, Y. Learning by Turning: Neural Architecture Aware Optimisation, `arXiv:2102.07227`, 2021b.

Loshchilov, I. and Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations, ICLR*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR*, 2019.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. In *7th International Conference on Learning Representations, ICLR*, 2019.

Ma, J. and Yarats, D. Quasi-hyperbolic momentum and Adam for deep learning. In *7th International Conference on Learning Representations, ICLR*, 2019.

Mahsereci, M. and Hennig, P. Probabilistic Line Searches for Stochastic Optimization. In *Journal of Machine Learning Research, JMLR*, volume 18, 2017.

Malkiel, I. and Wolf, L. MTAdam: Automatic Balancing of Multiple Training Loss Terms. *arXiv preprint:* `2006.14683`, 2020.

Martens, J. and Grosse, R. Optimizing Neural Networks with Kronecker-Factored Approximate Curvature. In *32nd International Conference on Machine Learning, ICML*, 2015.

Mukkamala, M. C. and Hein, M. Variants of RMSProp and Adagrad with Logarithmic Regret Bounds. In *34th International Conference on Machine Learning, ICML*, 2017.

Mutschler, M. and Zell, A. Parabolic Approximation Line Search for DNNs. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Nazari, P., Tarzanagh, D. A., and Michailidis, G. DADAM: A Consensus-based Distributed Adaptive Gradient Method for Online Optimization. *arXiv preprint:* `1901.09109`, 2019.

Nesterov, Y. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27, 1983.

Orabona, F. and Pál, D. Scale-Free Algorithms for Online Linear Optimization. In *Algorithmic Learning Theory - 26th International Conference, ALT*, 2015.

Orvieto, A., Köhler, J., and Lucchi, A. The Role of Memory in Stochastic Optimization. In *35th Conference on Uncertainty in Artificial Intelligence, UAI*, 2019.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5), 1964.

Preechakul, K. and Kijsirikul, B. CProp: Adaptive Learning Rate Scaling from Past Gradient Conformity. *arXiv preprint:* `1912.11493`, 2019.

Purkayastha, S. and Purkayastha, S. A Variant of Gradient Descent Algorithm Based on Gradient Averaging. *arXiv preprint:* `2012.02387`, 2020.

Ramezani-Kebrya, A., Khisti, A., and Liang, B. On the Generalization of Stochastic Gradient Descent with Momentum. *arXiv preprint:* `2102.13653`, 2021.

Reddi, S. J., Kale, S., and Kumar, S. On the Convergence of Adam and Beyond. In *6th International Conference on Learning Representations, ICLR*, 2018.

Ren, Y. and Goldfarb, D. Kronecker-factored Quasi-Newton Methods for Convolutional Neural Networks. *arXiv preprint:* `2102.06737`, 2021.

Robbins, H. and Monro, S. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3), 1951.

Rolínek, M. and Martius, G. L4: Practical loss-based stepsize adaptation for deep learning. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Roy, S. K., Paoletti, M. E., Haut, J. M., Dubey, S. R., Kar, P., Plaza, A., and Chaudhuri, B. B. AngularGrad: A New Optimization Technique for Angular Convergence of Convolutional Neural Networks. *arXiv preprint:* `2105.10190`, 2021.

Salas, A., Kessler, S., Zohren, S., and Roberts, S. Practical Bayesian Learning of Neural Networks via Adaptive Subgradient Methods. *arXiv preprint:* `1811.03679`, 2018.

Savarese, P., McAllester, D., Babu, S., and Maire, M. Domain-independent Dominance of Adaptive Methods. *arXiv preprint:* `1912.01823`, 2019.

Schaul, T. and LeCun, Y. Adaptive learning rates and parallelization for stochastic, sparse, non-smooth gradients. In *1st International Conference on Learning Representations, ICLR*, 2013.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates. In *30th International Conference on Machine Learning, ICML*, 2013.

Shang, F., Zhou, K., Liu, H., Cheng, J., Tsang, I. W., Zhang, L., Tao, D., and Jiao, L. VR-SGD: A Simple Stochastic Variance Reduction Method for Machine Learning. *IEEE Trans. Knowl. Data Eng.*, 32(1), 2020.

Shazeer, N. and Stern, M. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *35th International Conference on Machine Learning, ICML*, 2018.

Smith, L. N. Cyclical Learning Rates for Training Neural Networks. In *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2017.

Smith, L. N. and Topin, N. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. *arXiv preprint: 1708.07120*, 2017.

Sun, H., Gu, L., and Sun, B. Adathm: Adaptive Gradient Method Based on Estimates of Third-Order Moments. In *4th IEEE International Conference on Data Science in Cyberspace, DSC*, 2019.

Sung, W., Choi, I., Park, J., Choi, S., and Shin, S. S-SGD: Symmetrical Stochastic Gradient Descent with Weight Noise Injection for Reaching Flat Minima. *arXiv preprint: 2009.02479*, 2020.

Tan, C., Ma, S., Dai, Y., and Qian, Y. Barzilai-Borwein Step Size for Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 29, NIPS*, 2016.

Tao, Z., Xia, Q., and Li, Q. A new perspective in understanding of Adam-Type algorithms and beyond. https://openreview.net/forum?id=SyxM51BYPB, 2019.

Teixeira, B., Tamersoy, B., Singh, V., and Kapoor, A. Adaloss: Adaptive Loss Function for Landmark Localization. *arXiv preprint: 1908.01070*, 2019.

Tieleman, T. and Hinton, G. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude, 2012.

Tong, Q., Liang, G., and Bi, J. Calibrating the Adaptive Learning Rate to Improve Convergence of ADAM. *arXiv preprint: 1908.00700*, 2019.

Tran, H. and Cutkosky, A. Correcting Momentum with Second-order Information. *arXiv preprint: 2103.03265*, 2021.

Tran, P. T. and Phong, L. T. On the Convergence Proof of AMSGrad and a New Version. *IEEE Access*, 7, 2019.

Tran, T. H., Nguyen, L. M., and Tran-Dinh, Q. Shuffling Gradient-Based Methods with Momentum. *arXiv preprint: 2011.11884*, 2020.

Tutunov, R., Li, M., Cowen-Rivers, A. I., Wang, J., and Bou-Ammar, H. Compositional ADAM: An Adaptive Compositional Solver. *arXiv preprint: 2002.03755*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30, NIPS*, 2017.

Vaswani, S., Mishkin, A., Laradji, I. H., Schmidt, M., Gidel, G., and Lacoste-Julien, S. Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates. In *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.

Vogels, T., Karimireddy, S. P., and Jaggi, M. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.

Wang, B. and Ye, Q. Stochastic Gradient Descent with Nonlinear Conjugate Gradient-Style Adaptive Momentum. *arXiv preprint: 2012.02188*, 2020.

Wang, B., Nguyen, T. M., Bertozzi, A. L., Baraniuk, R. G., and Osher, S. J. Scheduled Restart Momentum for Accelerated Stochastic Gradient Descent. *arXiv preprint:* 2002.10583, 2020a.

Wang, D., Liu, Y., Tang, W., Shang, F., Liu, H., Sun, Q., and Jiao, L. signADAM++: Learning Confidences for Deep Neural Networks. In *International Conference on Data Mining Workshops, ICDM*, 2019a.

Wang, G., Lu, S., Cheng, Q., Tu, W., and Zhang, L. SAdam: A Variant of Adam for Strongly Convex Functions. In *8th International Conference on Learning Representations, ICLR*, 2020b.

Wang, J. and Wiens, J. AdaSGD: Bridging the gap between SGD and Adam. *arXiv preprint:* 2006.16541, 2020.

Wang, S., Sun, J., and Xu, Z. HyperAdam: A Learnable Task-Adaptive Adam for Network Training. In *33rd AAAI Conference on Artificial Intelligence, AAAI*, 2019b.

Wright, L. Deep Memory. https://github.com/lessw2020/Best-Deep-Learning-Optimizers/tree/master/DeepMemory, 2020a.

Wright, L. Ranger. https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2020b.

Wu, X., Ward, R., and Bottou, L. WNGrad: Learn the Learning Rate in Gradient Descent. *arXiv preprint:* 1803.02865, 2018.

Xie, C., Koyejo, O., Gupta, I., and Lin, H. Local AdaAlter: Communication-Efficient Stochastic Gradient Descent with Adaptive Learning Rates. *arXiv preprint:* 1911.09030, 2019.

Xie, Z., Wang, X., Zhang, H., Sato, I., and Sugiyama, M. Adai: Separating the Effects of Adaptive Learning Rate and Momentum Inertia. *arXiv preprint:* 2006.15815, 2020.

Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. A Walk with SGD. *arXiv preprint:* 1802.08770, 2018.

Xu, Y. Momentum-based variance-reduced proximal stochastic gradient method for composite nonconvex stochastic optimization. *arXiv preprint:* 2006.00425, 2020.

Yang, M., Xu, D., Li, Y., Wen, Z., and Chen, M. Structured Stochastic Quasi-Newton Methods for Large-Scale Optimization Problems. *arXiv preprint:* 2006.09606, 2020.

Yao, Z., Gholami, A., Shen, S., Keutzer, K., and Mahoney, M. W. ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning. *arXiv preprint:* 2006.00719, 2020.

You, Y., Gitman, I., and Ginsburg, B. Large Batch Training of Convolutional Networks. *arXiv preprint:* 1708.03888, 2017.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR*, 2020.

Yuan, W. and Gao, K.-X. EAdam Optimizer: How ϵ Impact Adam. *arXiv preprint:* 2011.02150, 2020.

Yue, X., Nouiehed, M., and Kontar, R. A. SALR: Sharpness-aware Learning Rates for Improved Generalization. *arXiv preprint:* 2011.05348, 2020.

Yun, J., Lozano, A. C., and Yang, E. Stochastic Gradient Methods with Block Diagonal Matrix Adaptation. *arXiv preprint:* 1905.10757, 2019.

Zaheer, M., Reddi, S. J., Sachan, D. S., Kale, S., and Kumar, S. Adaptive Methods for Nonconvex Optimization. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint:* 1212.5701, 2012.

Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. Noisy Natural Gradient as Variational Inference. In *35th International Conference on Machine Learning, ICML*, 2018.

Zhang, J. and Gouza, F. B. GADAM: Genetic-Evolutionary ADAM for Deep Neural Network Optimization. *arXiv preprint:* `1805.07500`, 2018.

Zhang, J. and Mitliagkas, I. YellowFin and the Art of Momentum Tuning. In *Machine Learning and Systems, MLSys*, 2019.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Zhang, M. R., Lucas, J., Hinton, G., and Ba, J. Lookahead Optimizer: k steps forward, 1 step back. *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.

Zhang, Z., Ma, L., Li, Z., and Wu, C. Normalized Direction-preserving Adam. *arXiv preprint:* `1709.04546`, 2017a.

Zhang, Z., Wu, Y., and Wang, G. BPGrad: Towards Global Optimality in Deep Learning via Branch and Pruning. *arXiv preprint:* `1711.06959`, 2017b.

Zhao, S.-Y., Xie, Y.-P., and Li, W.-J. Stochastic Normalized Gradient Descent with Momentum for Large Batch Training. *arXiv preprint:* `2007.13985`, 2020.

Zhou, B., Zheng, X., and Gao, J. On the Trend-corrected Variant of Adaptive Stochastic Optimization Methods. In *International Joint Conference on Neural Networks, IJCNN*, 2020.

Zhou, Y., Huang, K., Cheng, C., Wang, X., and Liu, X. FastAdaBelief: Improving Convergence Rate for Belief-based Adaptive Optimizer by Strong Convexity. *arXiv preprint:* `2104.13790`, 2021a.

Zhou, Y., Li, X., and Banerjee, A. Noisy Truncated SGD: Optimization and Generalization. *arXiv preprint:* `2103.00075`, 2021b.

Zhou, Z., Zhang, Q., Lu, G., Wang, H., Zhang, W., and Yu, Y. AdaShift: Decorrelation and Convergence of Adaptive Learning Rate Methods. In *7th International Conference on Learning Representations, ICLR*, 2019.

Zhuang, J., Tang, T., Ding, Y., Tatikonda, S., Dvornek, N., Papademetris, X., and Duncan, J. S. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. In *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Ziyin, L., Wang, Z. T., and Ueda, M. LaProp: a Better Way to Combine Momentum with Adaptive Gradient. *arXiv preprint:* `2002.04839`, 2020.