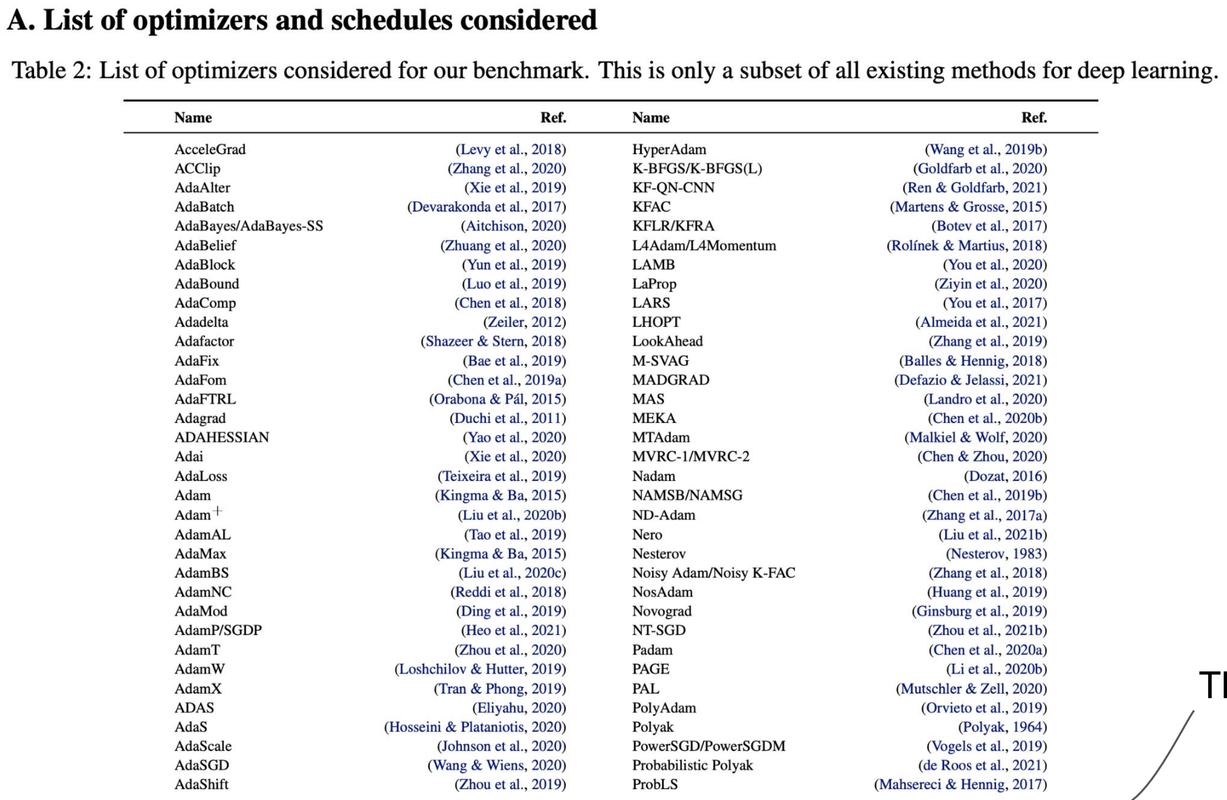
VeLO
Training Versatile Learned Optimizers by Scaling Up

Motivation



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.

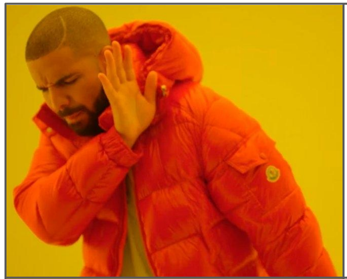
Name	Ref.	Name	Ref.
AcceleGrad	(Levy et al., 2018)	HyperAdam	(Wang et al., 2019b)
ACClip	(Zhang et al., 2020)	K-BFGS/K-BFGS(L)	(Goldfarb et al., 2020)
AdaAlter	(Xie et al., 2019)	KF-QN-CNN	(Ren & Goldfarb, 2021)
AdaBatch	(Devarakonda et al., 2017)	KFAC	(Martens & Grosse, 2015)
AdaBayes/AdaBayes-SS	(Aitchison, 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)

There's more…

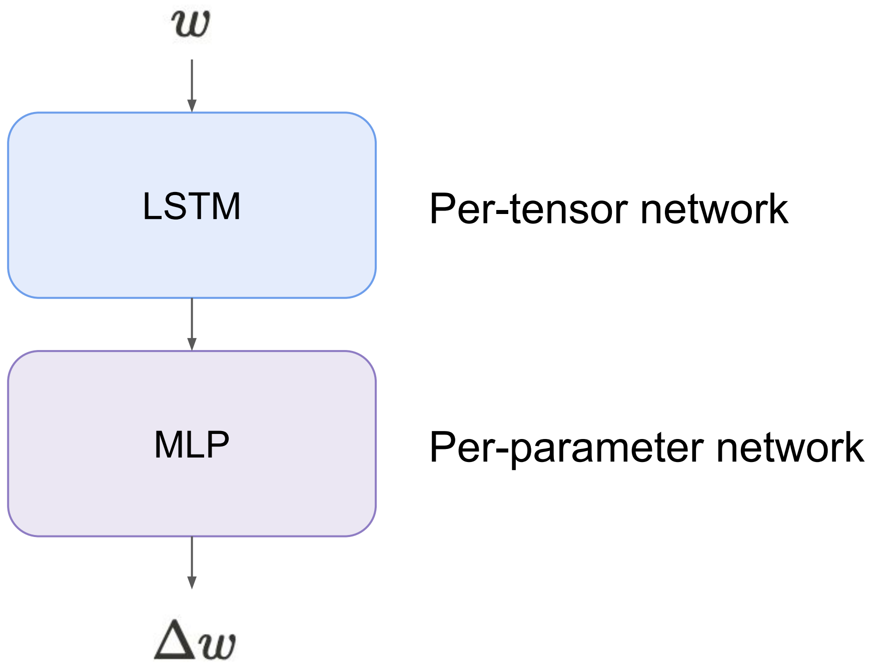
Schmidt, Schneider, Hennig (2021)

Please welcome, VeLO

VeLO - a neural network, that acts like optimizer and requires no hyperparameter tuning!

	$w_{t+1} = w_t - \lambda \nabla_{w_t} \mathcal{L}$
	$w_{t+1} = w_t - \mathrm{VeLO}(w_t, \ldots)$

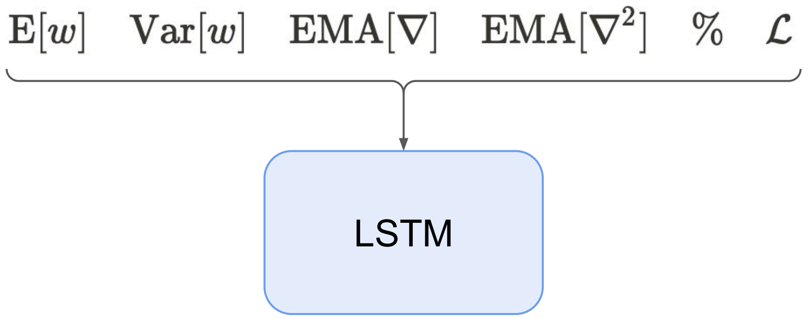
Architecture



Architecture

- Per-tensor network

  Input:
  - mean, variance of weights
  - EMA of gradient and squared gradient
  - fraction of training completed
  - training loss

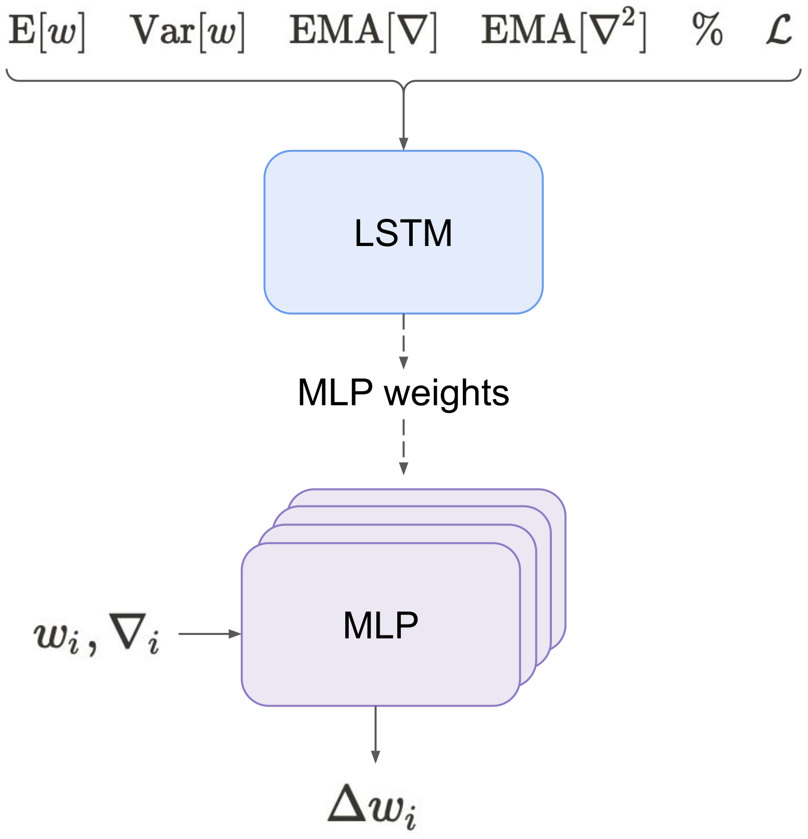


Architecture

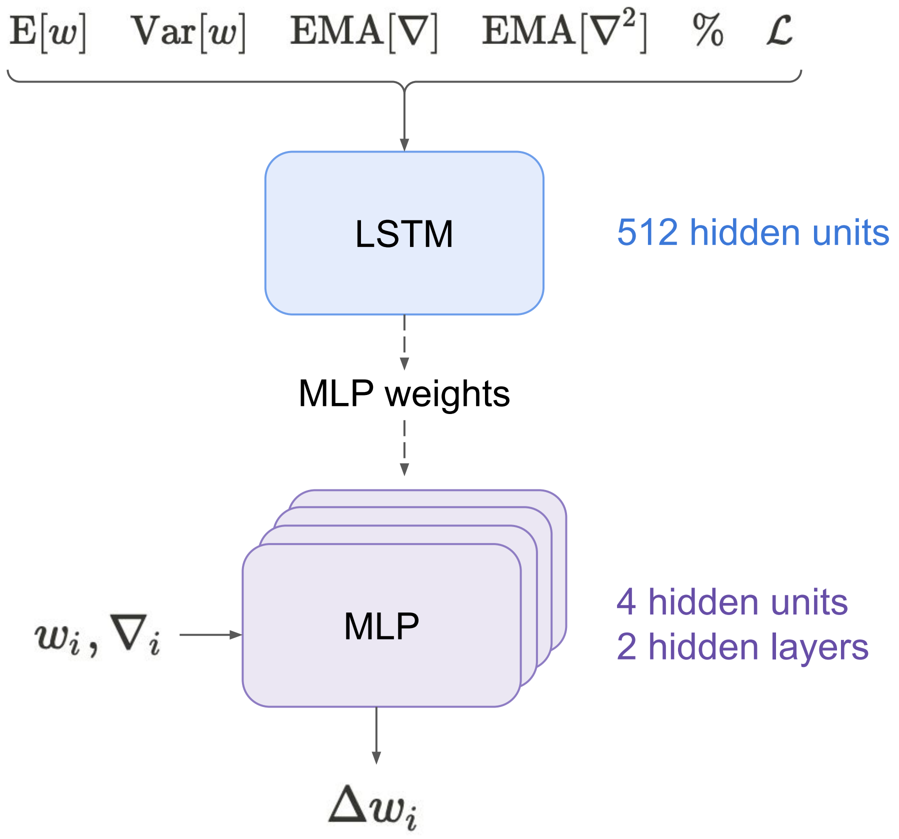
- Per-tensor network
- Per-parameter network

  Input:

  - value of the weight
  - gradients



Architecture


$\mathrm{E}[w]$ $\mathrm{Var}[w]$ $\mathrm{EMA}[\nabla]$ $\mathrm{EMA}[\nabla^2]$ $\%$ $\mathcal{L}$
LSTM
512 hidden units
MLP weights
MLP
4 hidden units
2 hidden layers
w_i, ∇_i
Δw_i

Learning an Optimizer

We have:

- Target model
- Its training data
- Its loss function

Learning an Optimizer

We have:

- Target model
- Its training data
- Its loss function

ϕ - target model weights

∇- target model gradients

$U(...;\theta)$- VeLO

$L(\theta)$- meta-loss

Meta-learning: learning to learn

1 training run = 1 data point for VeLO

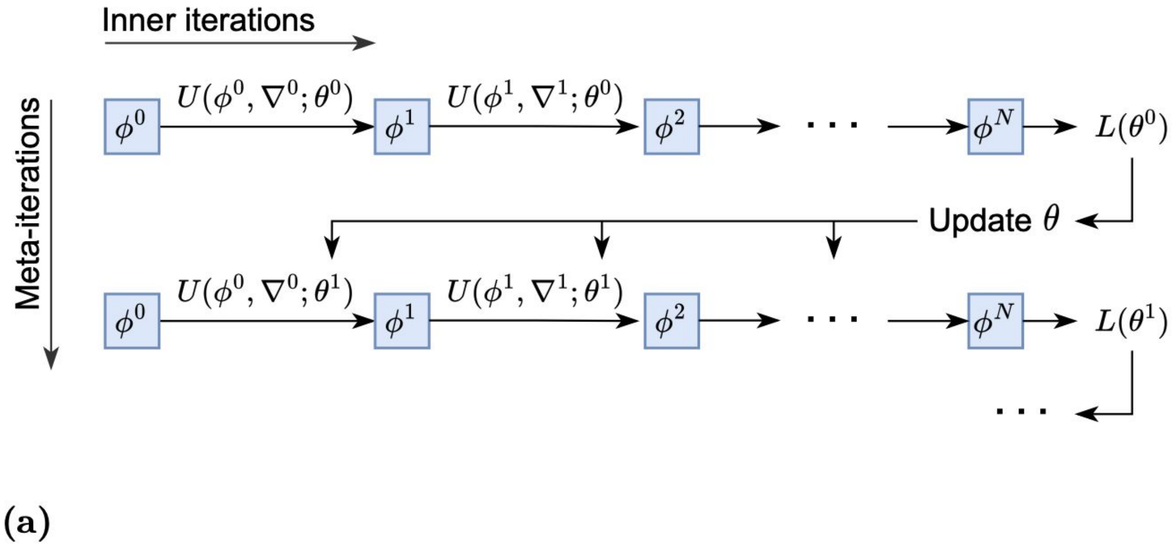

Inner iterations
Meta-iterations
ϕ^0
$U(\phi^0, \nabla^0; \theta^0)$
ϕ^1
$U(\phi^1, \nabla^1; \theta^0)$
ϕ^2
ϕ^N
$L(\theta^0)$
Update θ
$U(\phi^0, \nabla^0; \theta^1)$
$U(\phi^1, \nabla^1; \theta^1)$
$L(\theta^1)$
(a)

Figure 2: **(a) Training and meta-training.**

Learning an Optimizer

We have:

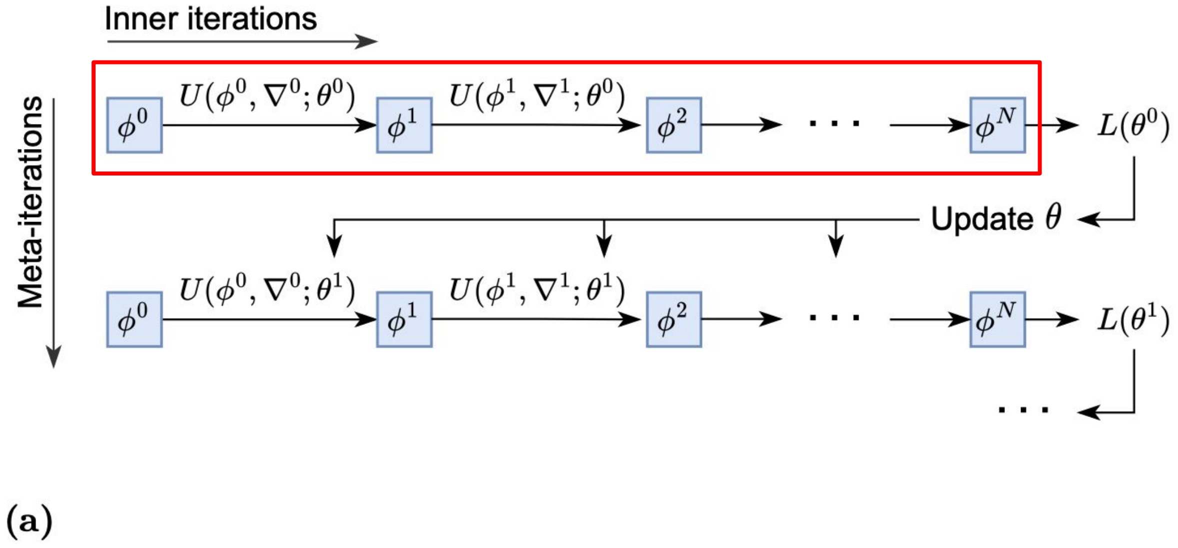
- Target model
- Its training data
- Its loss function

ϕ - target model weights
∇ - target model gradients

$U(...;\theta)$ - VeLO
$L(\theta)$ - meta-loss

Meta-learning: learning to learn

1 training run = 1 data point for VeLO



Inner iterations
Meta-iterations
ϕ^0
$U(\phi^0, \nabla^0; \theta^0)$
ϕ^1
$U(\phi^1, \nabla^1; \theta^0)$
ϕ^2
ϕ^N
$L(\theta^0)$
Update θ
$U(\phi^0, \nabla^0; \theta^1)$
$U(\phi^1, \nabla^1; \theta^1)$
$L(\theta^1)$
(a)

Figure 2: **(a) Training and meta-training.**

Learning an Optimizer

We have:

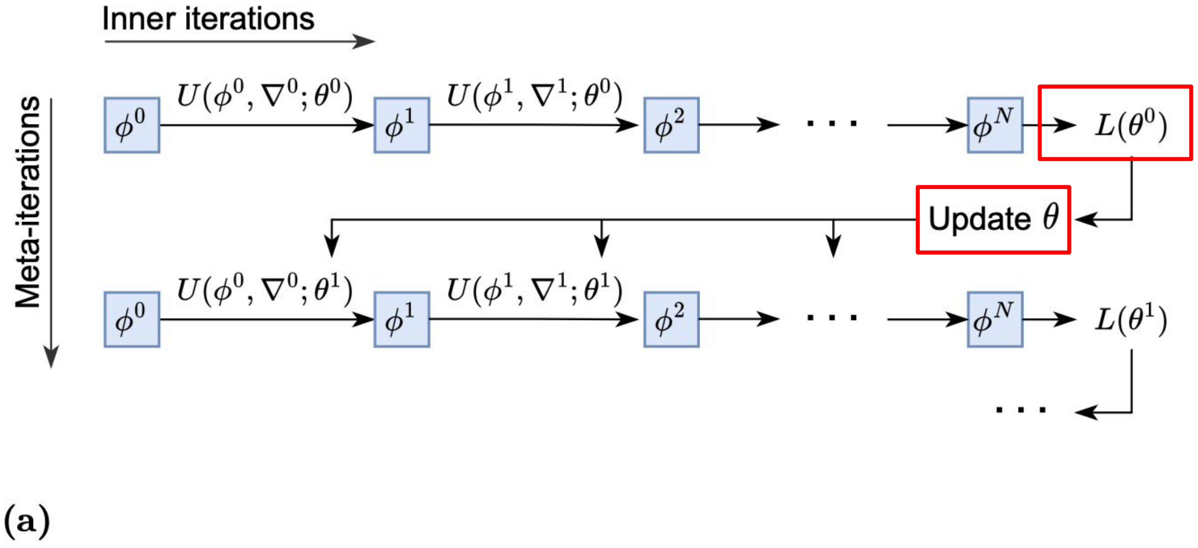
- Target model
- Its training data
- Its loss function

ϕ - target model weights
∇ - target model gradients

$U(\dots;\theta)$ - VeLO
$L(\theta)$ - meta-loss

Meta-learning: learning to learn

1 training run = 1 data point for VeLO



Inner iterations
Meta-iterations
ϕ^0
$U(\phi^0, \nabla^0; \theta^0)$
ϕ^1
$U(\phi^1, \nabla^1; \theta^0)$
ϕ^2
ϕ^N
$L(\theta^0)$
Update θ
$U(\phi^0, \nabla^0; \theta^1)$
$U(\phi^1, \nabla^1; \theta^1)$
$L(\theta^1)$
(a)

Figure 2: **(a) Training and meta-training.**

Learning an Optimizer

We have:

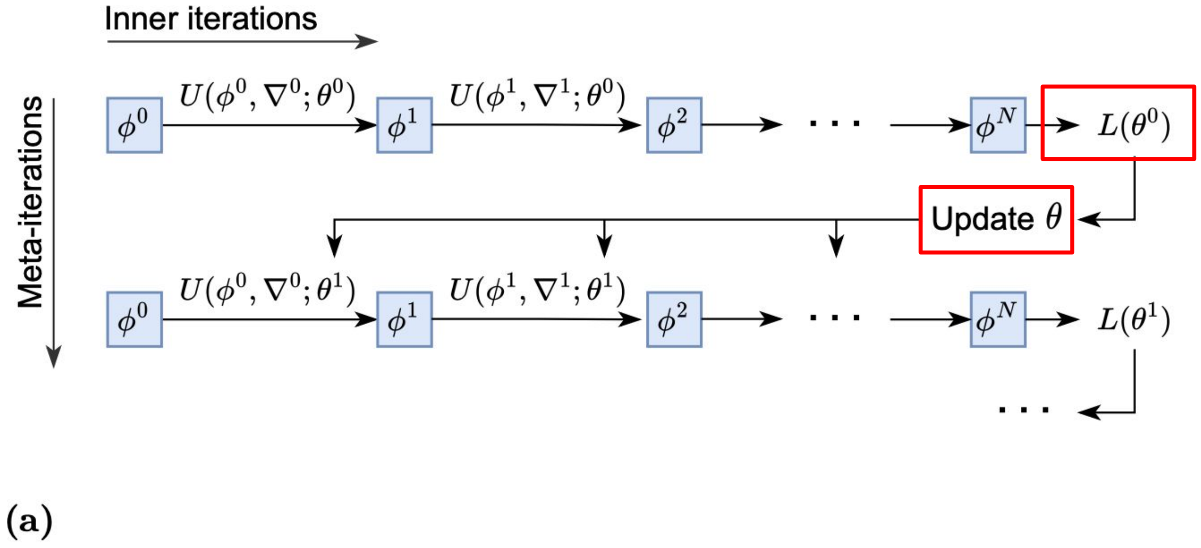
- Target model
- Its training data
- Its loss function

$$L(\theta) = \ell_N(\phi^N)$$

$\ell_N(\phi^N)$ - loss at the end of target model training

Meta-learning: learning to learn

1 training run = 1 data point for VeLO



Inner iterations
Meta-iterations
ϕ^0
$U(\phi^0, \nabla^0; \theta^0)$
ϕ^1
$U(\phi^1, \nabla^1; \theta^0)$
ϕ^2
ϕ^N
$L(\theta^0)$
Update θ
$U(\phi^0, \nabla^0; \theta^1)$
$U(\phi^1, \nabla^1; \theta^1)$
$L(\theta^1)$
(a)

Figure 2: **(a) Training and meta-training.**

Training data

Model families:

- MLPs
- CNNs
- ResNets
- Transformers
- RNNs
- (V)AEs

Training data

Model families:

- MLPs
- CNNs
- ResNets
- Transformers
- RNNs
- (V)AEs

1. Sample model family & data type & loss
   (image-MLP, image-CNN, LM-Transformer, …)
2. Sample model architecture
   (depth, width, activation)
3. Sample dataset

VeLOdrome

- 83 tasks
- wide range of models
- training time on 1 GPU < 1h

Results

VeLOdrome

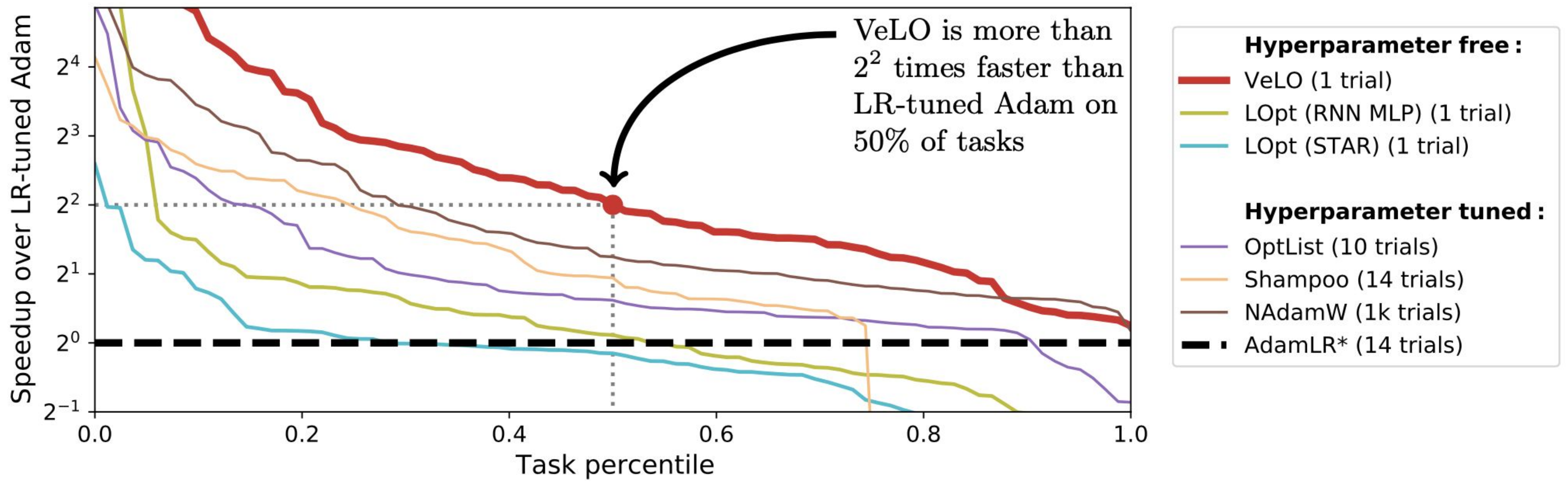
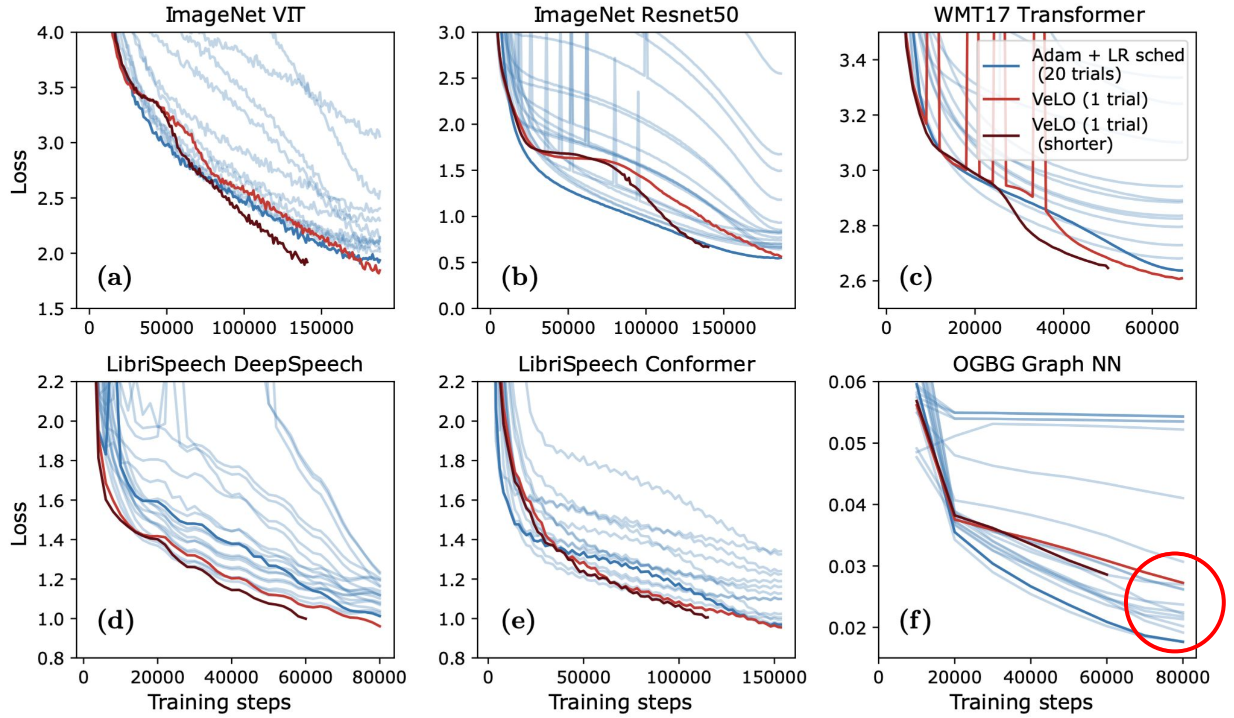


Figure 1: **Optimizer performance on the 83 canonical tasks in the VeLOdrome benchmark.**

Results

MLCommon Tasks (out-of-distribution)



Limitations

- > 200K training steps
- > 500M model parameters

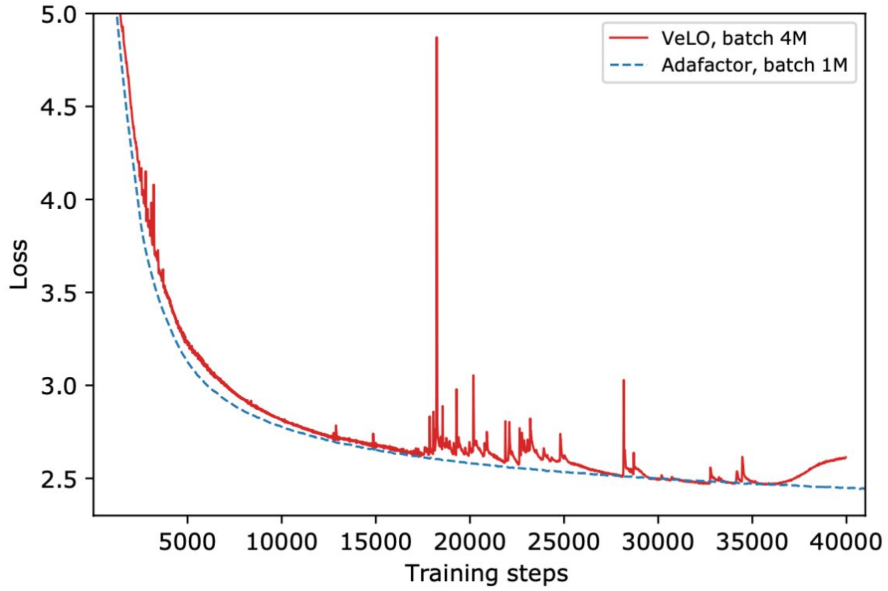


Figure 9: **VeLO struggles to train models which are much larger than those used for meta-training** Plot shows training of an 8B parameter Transformer language model, trained to 160B tokens. VeLO exhibits instability even with weight decay, and underperforms an untuned Adafactor baseline with exponential learning rate decay on a step-for-step basis despite 4x larger batch.

References

- [Current paper] VeLO: Training Versatile Learned Optimizers by Scaling Up
- [Schmidt, Schneider, Hennig (2021)] Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers