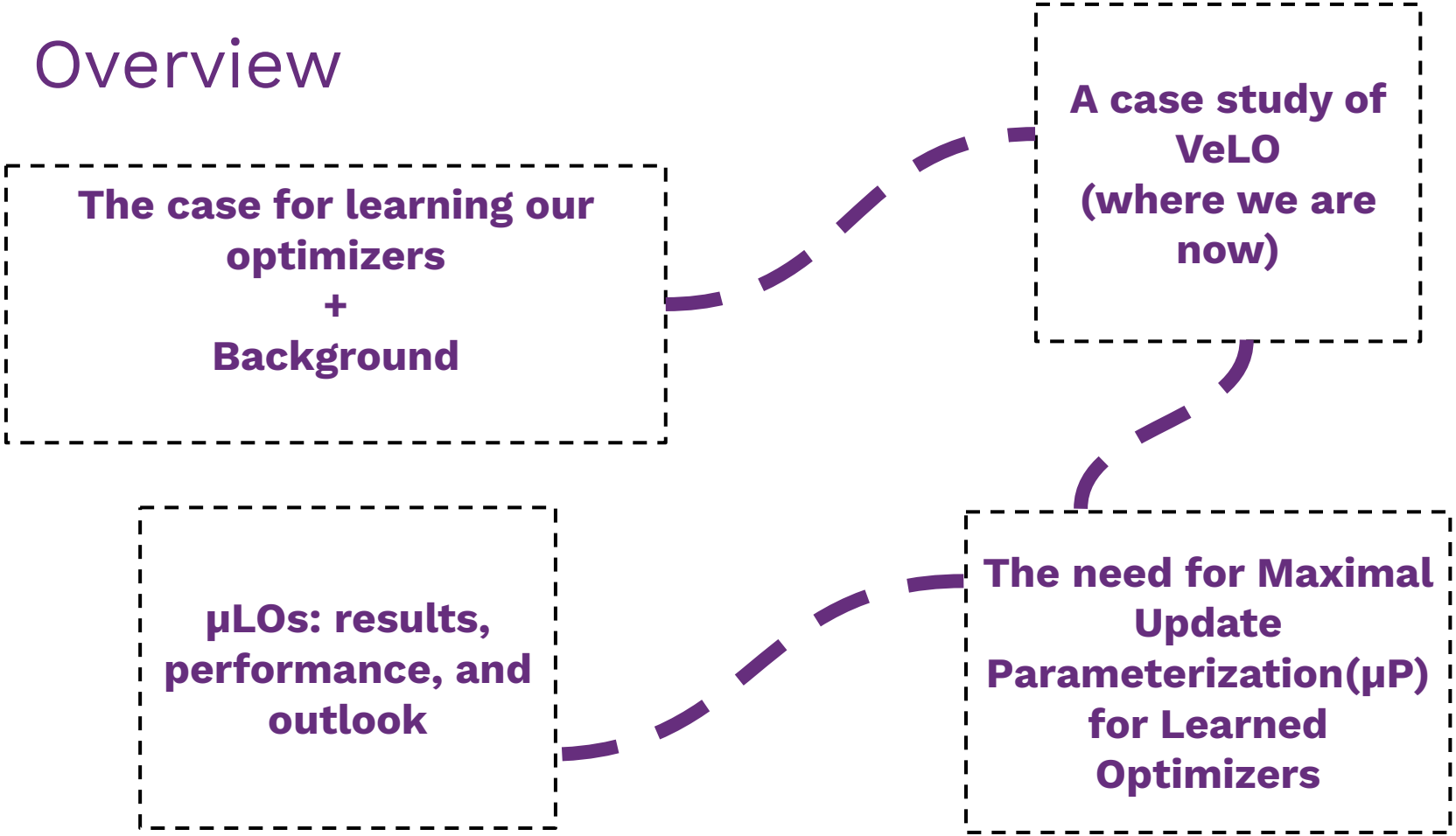
Mila

ChatGPT
"Here is an image of a learned optimizer training a gigantic neural network in a futuristic setting."

μLO: Compute-Efficient Meta-Generalization of Learned Optimizers

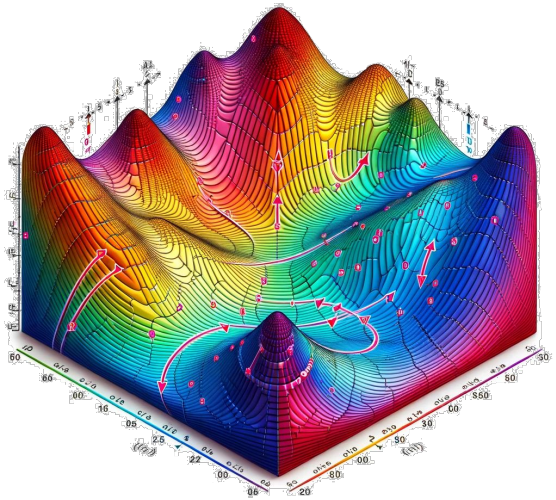
by **Benjamin Thérien, Charles-Étienne Joseph, Boris Knyazev, Edouard Oyallon, Irina Rish, Eugene Belilovsky**

Overview


The case for learning our optimizers + Background
A case study of VeLO (where we are now)
The need for Maximal Update Parameterization(μP) for Learned Optimizers
μLOs: results, performance, and outlook

Why should we learn optimizers?

- Gradient-based optimizers such as Adam, SGD, etc... are prevalent in DL.
- But, DL optimization problems are highly non-convex and theory is hard.
- No reason to expect our optimizers will converge to the global optimum at the optimal rate.



Can we meta-learn better optimization algorithms?

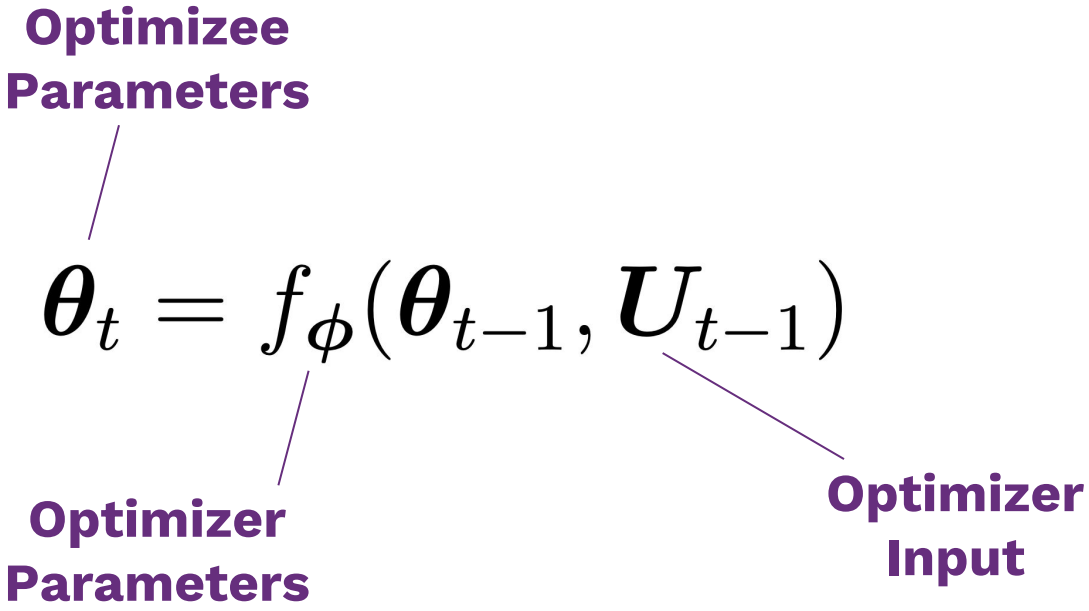
The Bitter Lesson:

“General methods that leverage computation are ultimately the most effective, and by a large margin.”

- Rich Sutton, 2019

Learned Optimizer Background

General Optimizer Update



Learned Optimizer Update

Optimizee Parameters

$$\boldsymbol{\theta}_t = f_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{t-1}, \boldsymbol{U}_{t-1}) = \boldsymbol{\theta}_{t-1} - \underbrace{\lambda_1 \boldsymbol{d}_{\boldsymbol{\phi}} e^{(\lambda_2 \boldsymbol{m}_{\boldsymbol{\phi}})}}_{\textbf{LO Update}}$$

$$g_{\boldsymbol{\phi}}(\boldsymbol{\theta}_{t-1}, \boldsymbol{U}_{t-1}) = [\boldsymbol{d}_{\boldsymbol{\phi}}, \boldsymbol{m}_{\boldsymbol{\phi}}]$$

Optimizer Neural Network

Learned Optimizer Input Features

Type	#	Description	Equation
Accumulators	3	Momentum accumulators with coefficients $\beta_i, i \in \{1, 2, 3\}$.	$m_{t,i} = \beta_i m_{t-1,i} + (1 - \beta_i)\nabla_t$
	1	Second moment accumulator with coefficients β_4.	$v_t = \beta_4 v_{t-1} + (1 - \beta_4)\nabla_t^2$
	3	Adafactor row accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$r_{t,i} = \beta_i r_{t-1,i} + (1 - \beta_i)\ \texttt{row\_mean}(\nabla_t^2)$
	3	Adafactor accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$c_{t,i} = \beta_i c_{t-1,i} + (1 - \beta_i)\ \texttt{col\_mean}(\nabla_t^2)$

Learned Optimizer Input Features

Type	#	Description	Equation
Accumulators	3	Momentum accumulators with coefficients $\beta_i, i \in \{1, 2, 3\}$.	$m_{t,i} = \beta_i m_{t-1,i} + (1 - \beta_i)\nabla_t$
	1	Second moment accumulator with coefficients β_4.	$v_t = \beta_4 v_{t-1} + (1 - \beta_4)\nabla_t^2$
	3	Adafactor row accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$r_{t,i} = \beta_i r_{t-1,i} + (1 - \beta_i)\ \texttt{row\_mean}(\nabla_t^2)$
	3	Adafactor accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$c_{t,i} = \beta_i c_{t-1,i} + (1 - \beta_i)\ \texttt{col\_mean}(\nabla_t^2)$

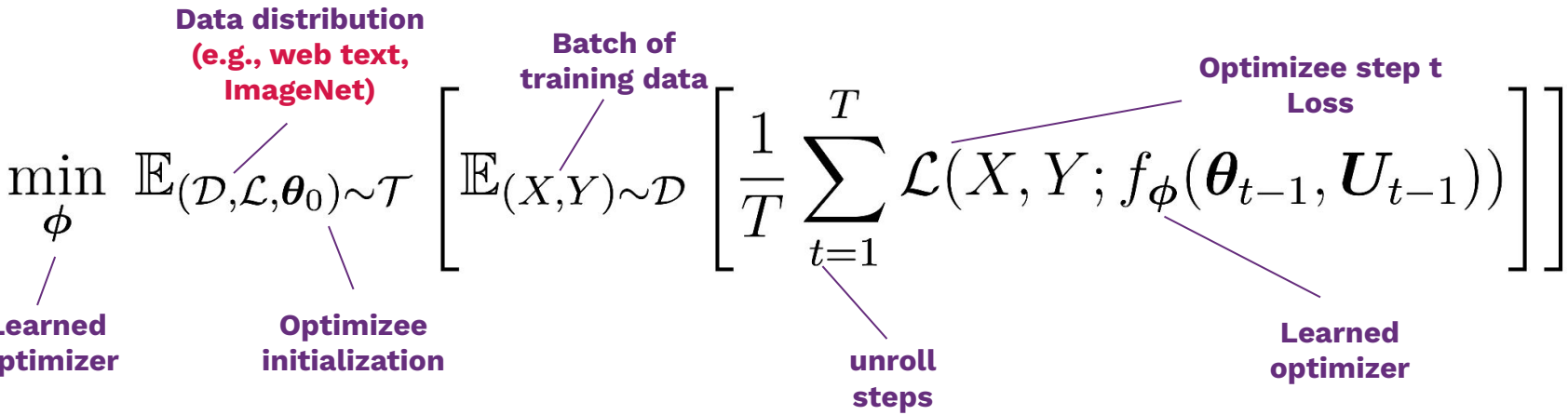
Learned Optimizer Input Features

Type	#	Description	Equation
Accumulators	3	Momentum accumulators with coefficients $\beta_i, i \in \{1, 2, 3\}$.	$m_{t,i} = \beta_i m_{t-1,i} + (1 - \beta_i)\nabla_t$
	1	Second moment accumulator with coefficients β_4.	$v_t = \beta_4 v_{t-1} + (1 - \beta_4)\nabla_t^2$
	3	Adafactor row accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$r_{t,i} = \beta_i r_{t-1,i} + (1 - \beta_i)\ \texttt{row\_mean}(\nabla_t^2)$
	3	Adafactor accumulator with coefficients $\beta_i, i \in \{5, 6, 7\}$.	$c_{t,i} = \beta_i c_{t-1,i} + (1 - \beta_i)\ \texttt{col\_mean}(\nabla_t^2)$

Learned Optimizer Input Features

Type	#	Description	Equation
Accumulator Features	3	Momentum values normalized by the square root of the second moment for $i \in \{5, 6, 7\}$.	$\frac{m_{t,i}}{\sqrt{v}}$
	1	The reciprocal square root of the second moment value.	$\frac{1}{\sqrt{v}}$
	6	The reciprocal square root of the Adafactor accumulators.	$\frac{1}{\sqrt{r_{t,i} \text{ OR } c_{t,i}}}$
	3	Adafactor gradient features for $i \in \{5, 6, 7\}$.	$\nabla_t \times r_{t,i} \times c_{t,i}$
	3	Adafactor momentum features for $i, j \in \{(5, 1), (6, 2), (7, 3)\}$.	$m_{t,j} \times r_{t,i} \times c_{t,i}$
Time Features	11	Time Features for $x \in \{1, 3, 10, 30, 100, 300, 1000, 3000, 10^4, 3 \cdot 10^4, 10^5\}$.	$\tanh\left(\frac{t}{x}\right)$
Parameters	1	Parameter value.	w_t

Learned Optimizer's Objective



1) Sample a data distribution **D**, Loss **L,** and weight **θ.**
2) Sample **X**,**Y** from **D**.
3) Compute the loss, **L**, for the current optimizee weights **θ**.
4) Use **ϕ** to update **w**.
5) Repeat for **T** steps.

Noisy Meta-loss Landscapes

Low Meta-Loss

High Meta-Loss

(Metz et al., 2019)

- Unrolled optimization problems have many steps
- Longer unrolls = Noisy gradients
- We should smooth the meta loss

Zeroth Order Meta-gradient Estimation

$$\nabla_{\phi}^{\text{ES-A}} = \frac{1}{N\sigma^2} \sum_{i=1}^{N/2} \boldsymbol{\epsilon}^{(i)} \left(L(\boldsymbol{\phi} + \boldsymbol{\epsilon}^{(i)}) - L(\boldsymbol{\phi} - \boldsymbol{\epsilon}^{(i)}) \right)$$

- By sampling epsilon from a gaussian we smooth the loss
- Zeroth order methods are used instead of backprop
- Antithetic sampling is used to reduce variance

Mila

VeLO Optimizer: A Case Study (where we are now)

Scaling learned optimization leads to faster per-step optimization

(Metz et al., 2022b)

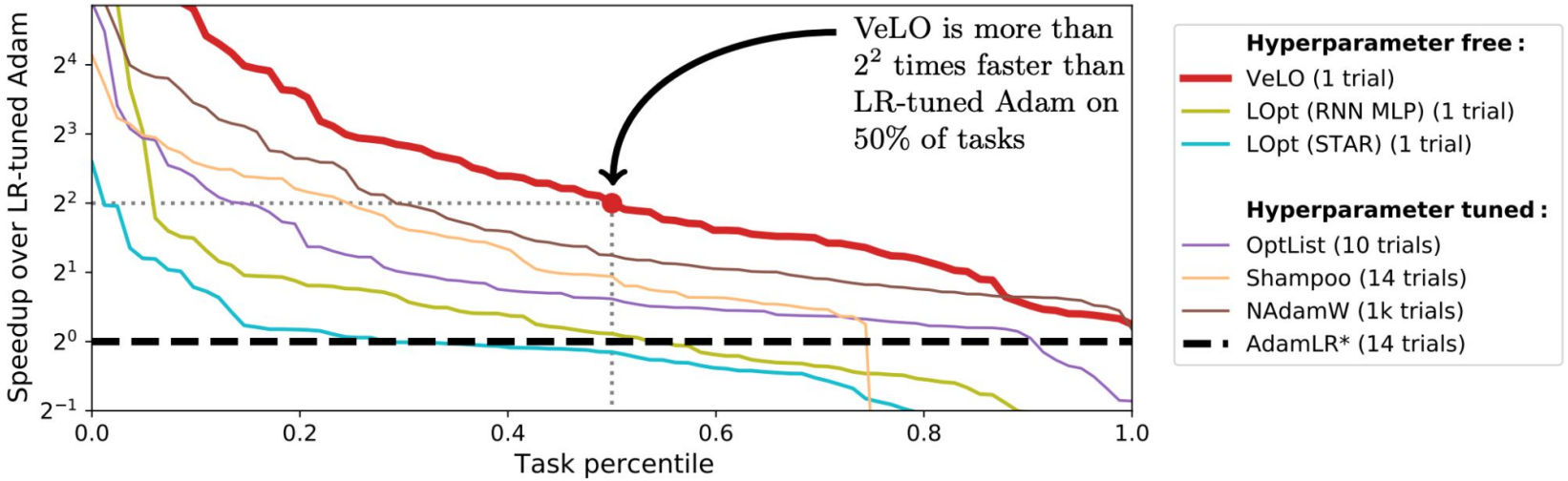
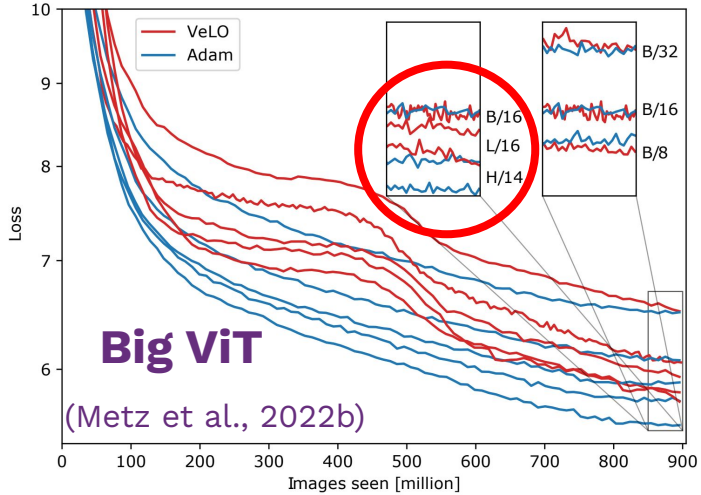
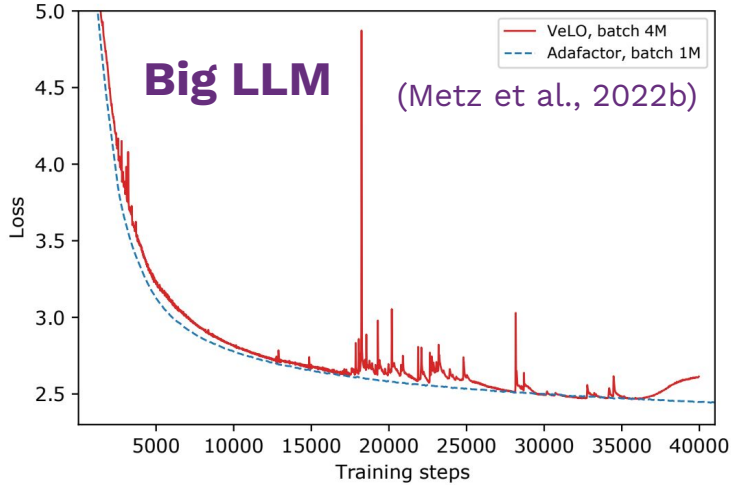


Figure 1: **Optimizer performance on the 83 canonical tasks in the VeLOdrome benchmark.**

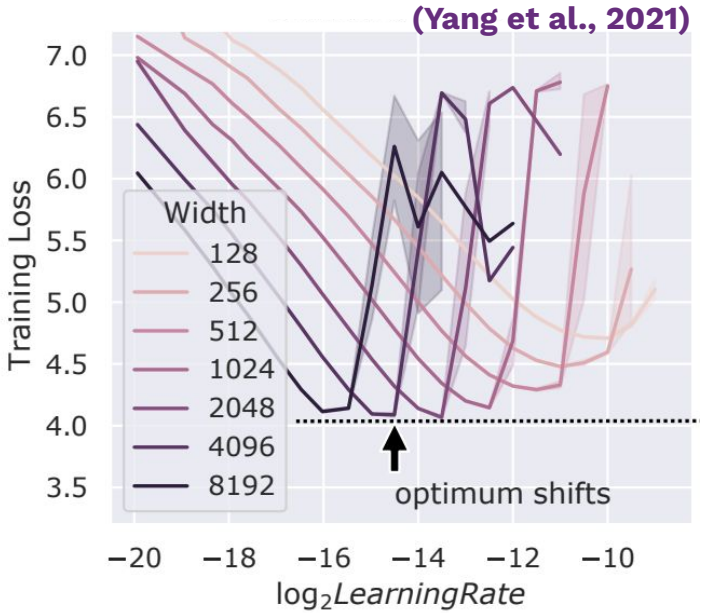
VeLO is outperformed by hand-designed optimizers for the largest models





- VeLO is tested on big models
- VeLO fails
- Why? VeLO is not trained on big models

The optimal learning rate depends on width



- Optimal LR depends on width in Standard Parameterization

μP Background; Parameterization

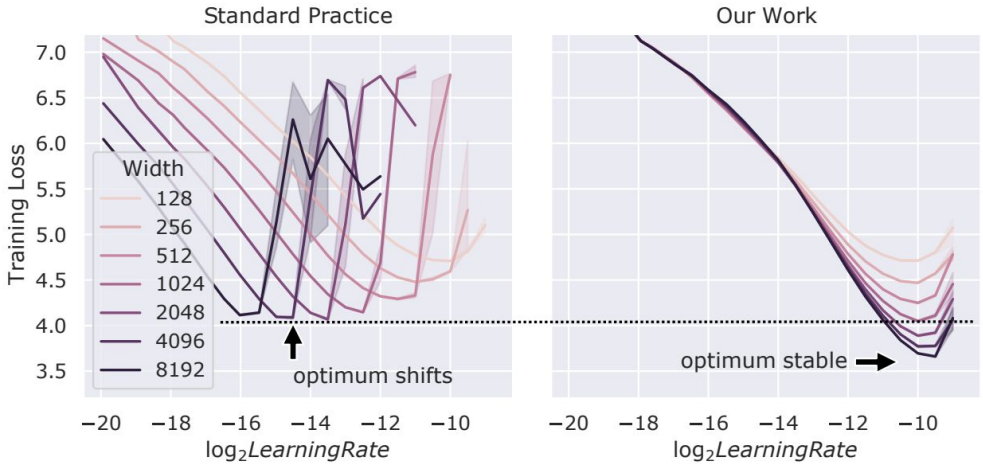
(Yang et al., 2021)

	Input weights & all biases	Output weights	Hidden weights
Init. Var.	$1/\text{fan\_in}$	1 $(1/\text{fan\_in})$	$1/\text{fan\_in}$
Multiplier	1	$1/\text{fan\_in}$ (1)	1
SGD LR	fan_out (1)	fan_in (1)	1
Adam LR	1	1	$1/\text{fan\_in}$ (1)

- A parameterization specifies:
  - initialization variance
  - pre-activation multipliers
  - LR multipliers

μP enables HP transfer from small width to large width models

(Yang et al., 2021)



- (**LEFT**) Optimal LR depends on width in SP
- (**RIGHT**) Optimal LR is the same for all widths in μP
- μP stabilizes training for infinite width

μP for learned optimizers

How μP Impacts the LO Update

Updated Parameter Value

μP LR Multiplier

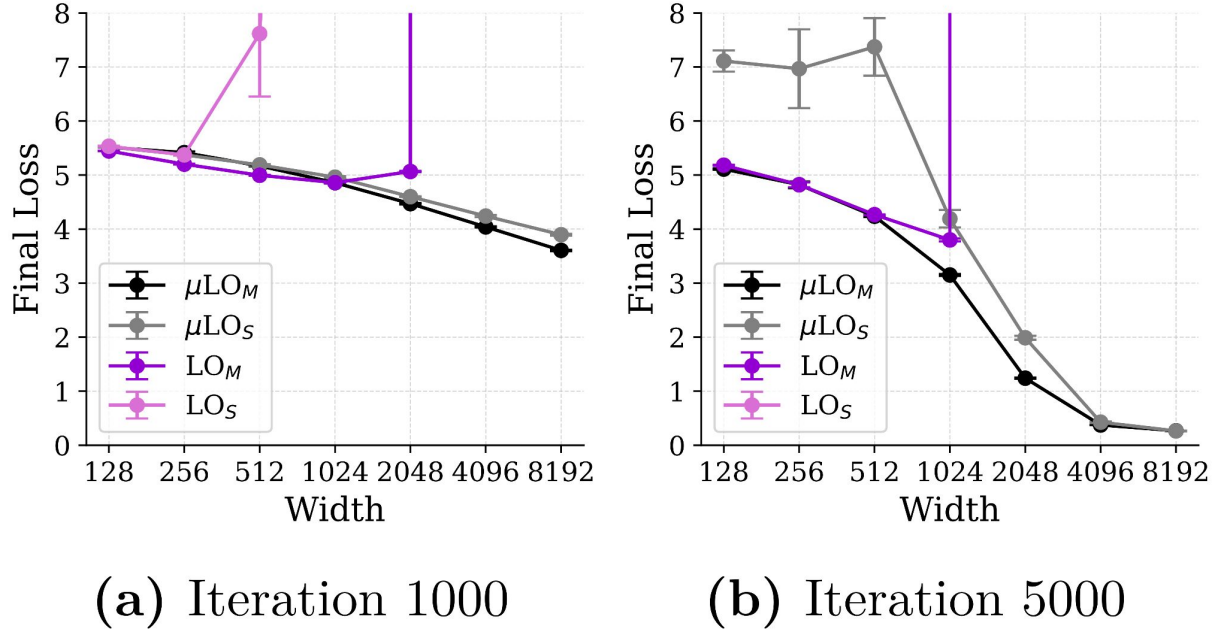
Direction

$$w_t = \begin{cases} w_{t-1}^i - \frac{1}{\text{FAN\_IN}} \cdot \left(\lambda_1 d_\phi^i \exp\left(\lambda_2 m_\phi^i\right)\right) & \text{if } w^i \text{ is part of a hidden layer} \\ w_{t-1}^i - \lambda_1 d_\phi^i \exp\left(\lambda_2 m_\phi^i\right) & \text{otherwise.} \end{cases}$$

Magnitude

- LO μP requires scaling the LR for hidden weights

μLO Meta-training Recipe



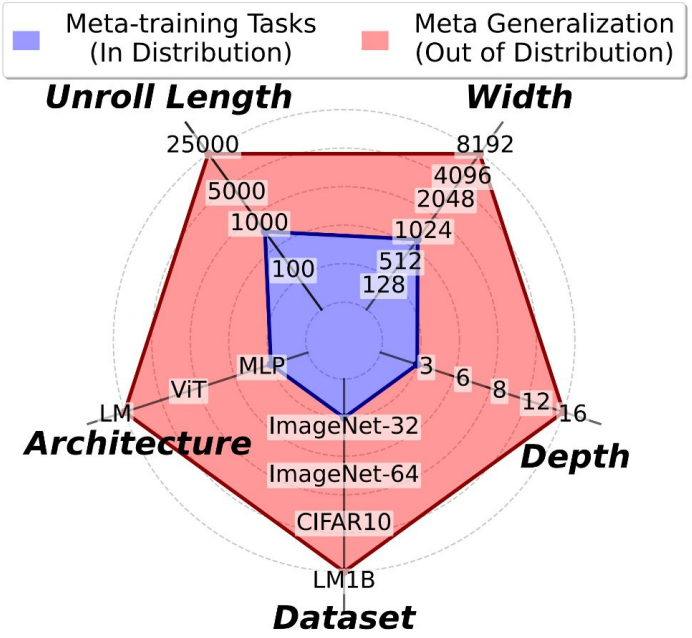
(a) Iteration 1000 (b) Iteration 5000

- **3-task Meta-training** v.s. **1-task Meta-training**
- **3-task Meta-training** consistently performs better
- We use **3-task Meta-training** in our experiments

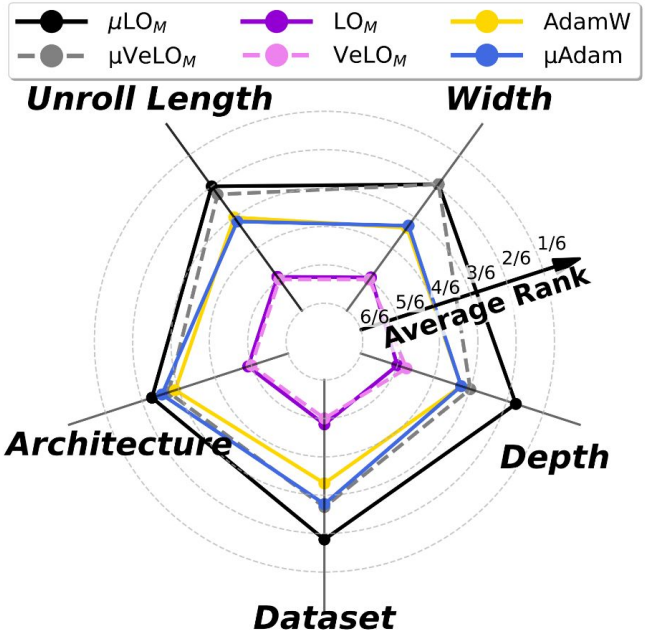
Mila

Results and Outlook of µLOs

Results summary



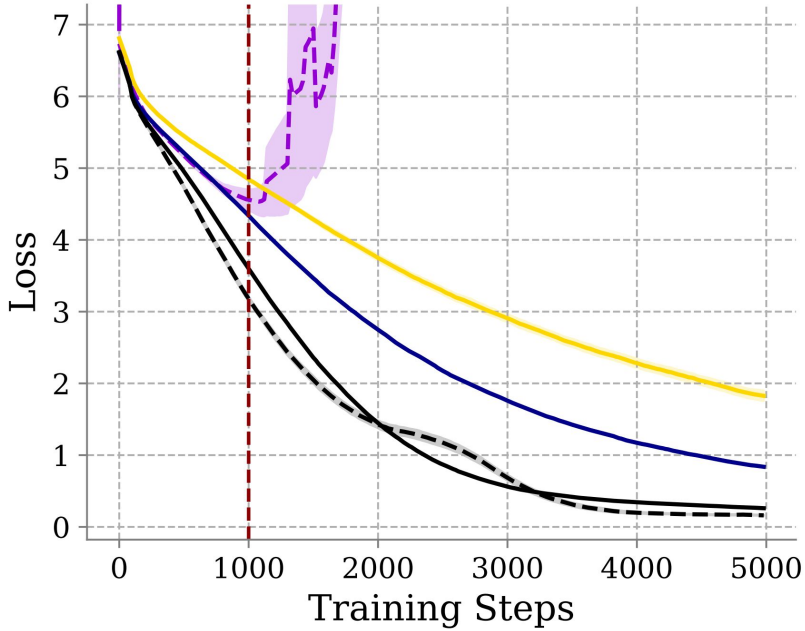
(a) Axes of Meta-Generalization



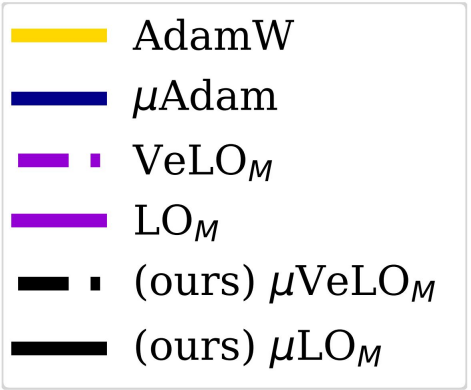
(b) Performance by Average Rank

- μLOs reach the highest average rank of all tasks

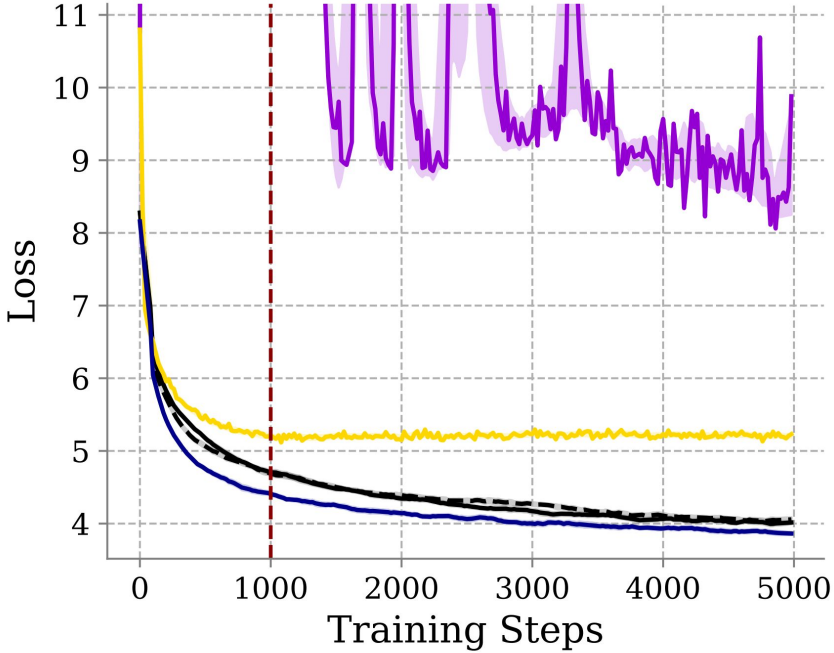
Width Generalization - Imagenet 32x32x3



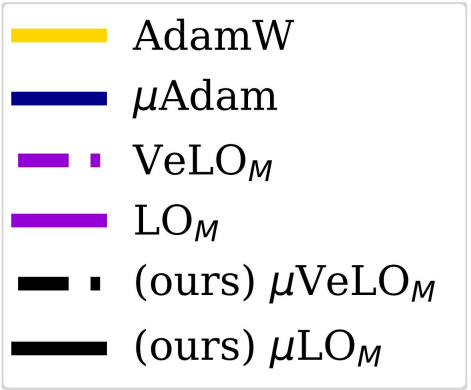
(a) MLP IN32 W=8192



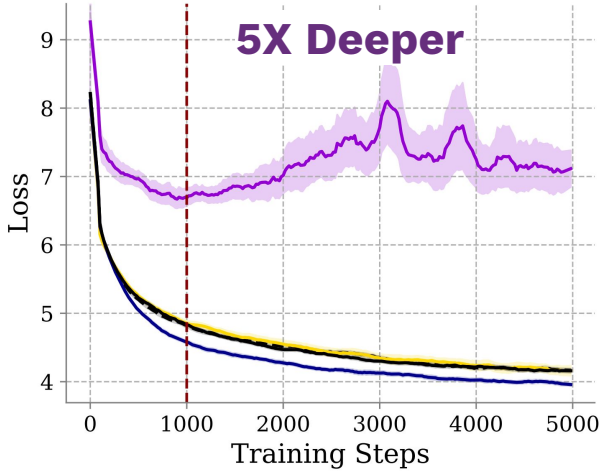
Width Generalization - Transformer LM



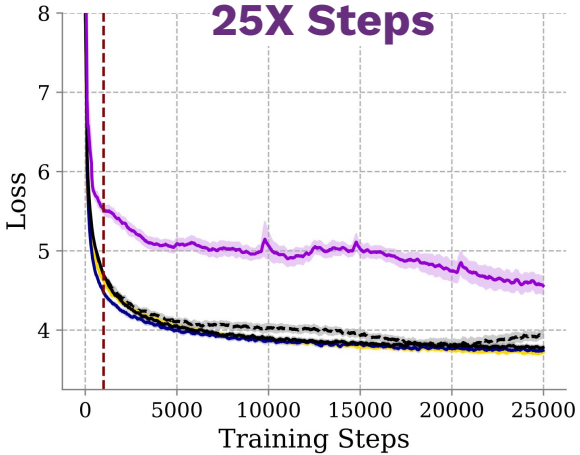
(d) LM W=4096



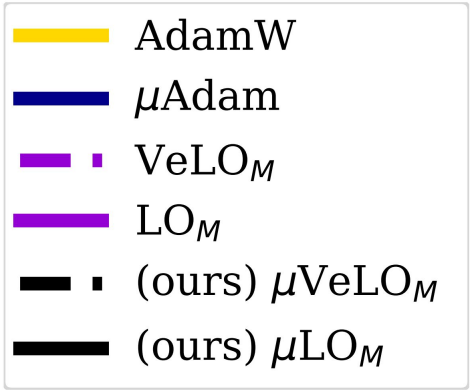
Deeper networks & Longer training



(b) LM W=1024 D=16

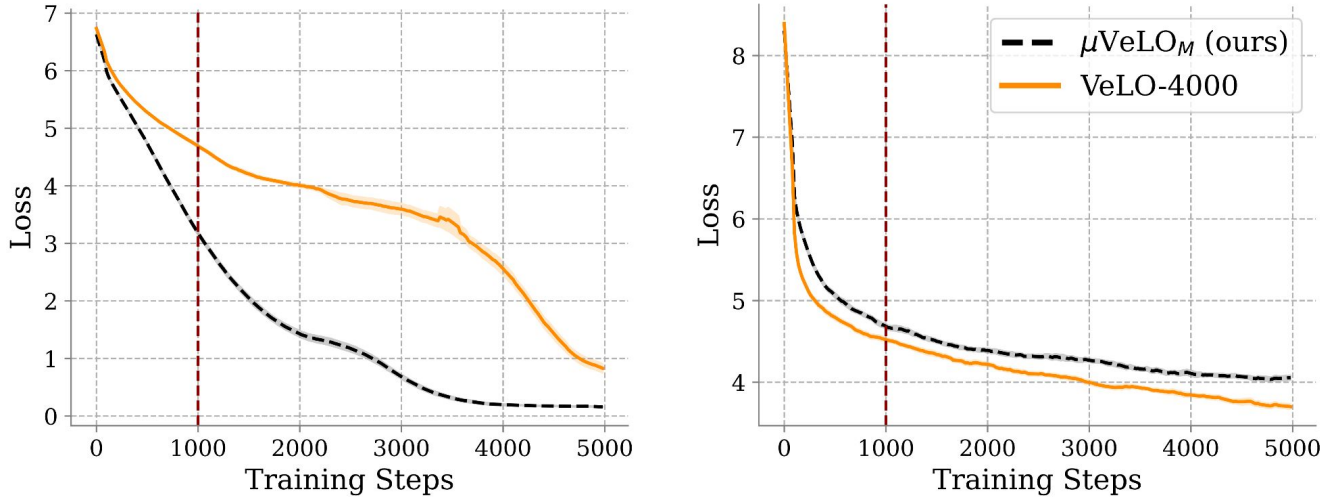


(b) LM W=1024



- We test on deeper networks and for longer training.
- We observe empirically, that μLOs improve generalization in both these cases.

Comparison with VeLO



(a) MLP IN32 W=8192

(b) LM W=4096

- We outperform VeLO when testing on the meta-training distribution.
- VeLO outperforms on OOD tasks.
- Will expanding µLOs meta-training distributions bridge the gap?

Conclusion: Learned optimizers should be meta-trained in μ-parameterization.

- We derived μ-parameterization for popular learned optimizers (VeLO and small_fc_lopt) and proposed a meta-training recipe for μLOs.
- μLOs generalize significantly better to wider networks than SP LOs and outperform VeLO on wider in-distribution tasks!
- μLOs also improve generalization to deeper networks (5X meta-training) and longer training horizons (25X meta-training).