ICML
International Conference
On Machine Learning

JOHNS HOPKINS
UNIVERSITY

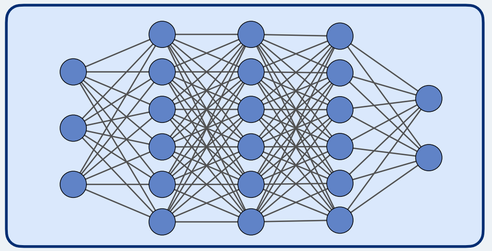
μpscaling Small models

Principled Warm Starts and Hyperparameter Transfer

Yuxin Ma, Nan Chen, Mateo Díaz, Soufiane Hayou, Dmitriy (Tim) Kunisky, Soledad Villar
Department of Applied Mathematics and Statistics, Johns Hopkins University

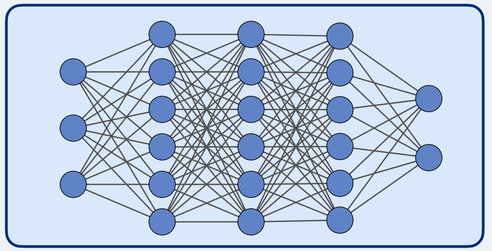
RANDOM INITIALIZATION

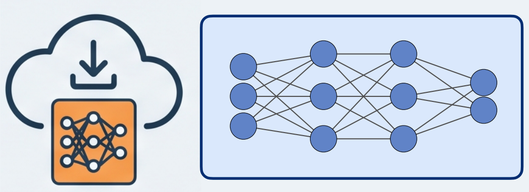
FULL TRAINING FROM SCRATCH



RANDOM
INITIALIZATION

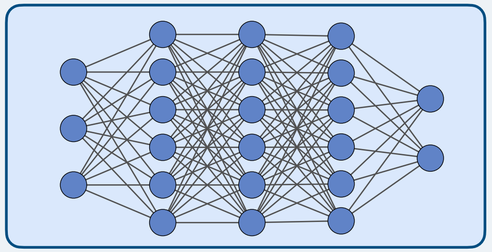
FULL TRAINING FROM SCRATCH



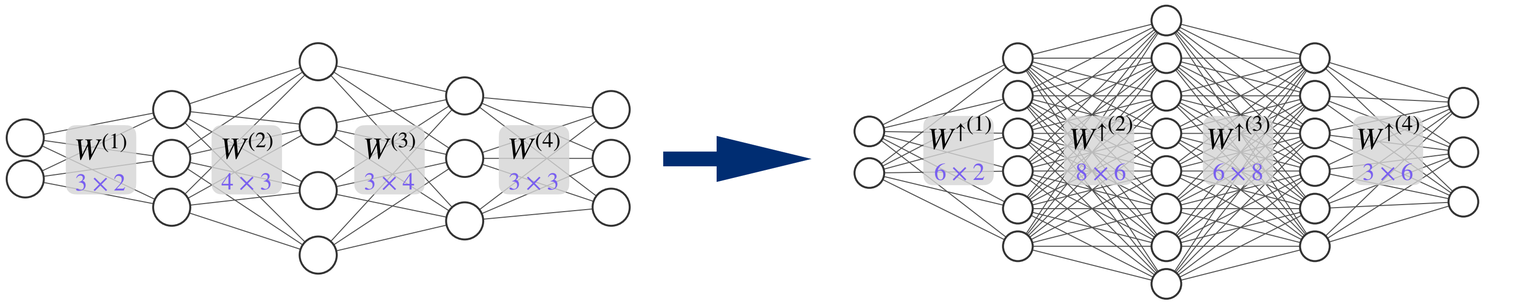


ACCESSIBLE
PRE-TRAINED
BASE MODEL

WARM-STARTED TRAINING
VIA MODEL UPSCALING



- **Key idea:** function preserving weight transformation.



Widened weight $W^{\uparrow(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$

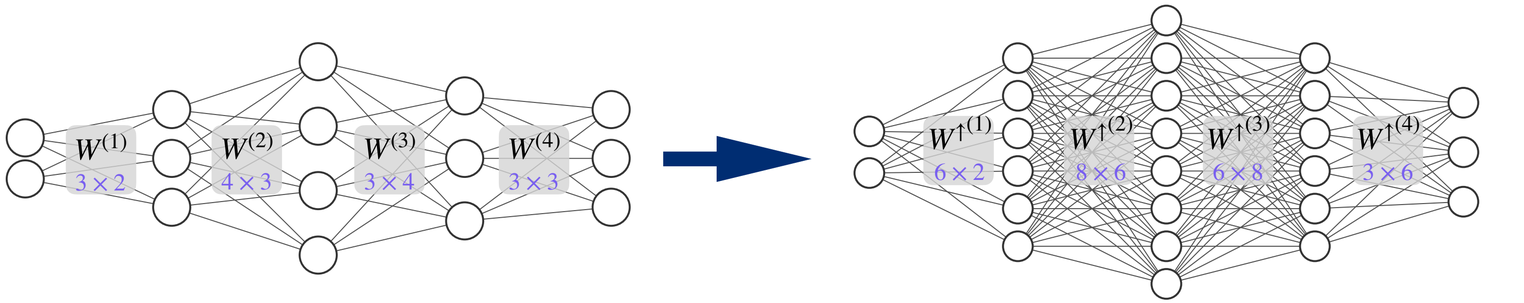
Base weight $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$

$$\begin{bmatrix} W^{(\ell)}_{1,1} & \cdots & W^{(\ell)}_{1,n_{\ell-1}} \\ \vdots & \ddots & \vdots \\ W^{(\ell)}_{n_\ell,1} & \cdots & W^{(\ell)}_{n_\ell,n_{\ell-1}} \end{bmatrix} \longrightarrow k_{\ell-1}^{-1} \cdot \begin{bmatrix} W^{(\ell)}_{1,1}\ \text{block} & \cdots & W^{(\ell)}_{1,n_{\ell-1}}\ \text{block} \\ \vdots & \ddots & \vdots \\ W^{(\ell)}_{n_\ell,1}\ \text{block} & \cdots & W^{(\ell)}_{n_\ell,n_{\ell-1}}\ \text{block} \end{bmatrix}$$

$k_{\ell-1}$ cols

k_ℓ rows

- **Key idea:** function preserving weight transformation.



Widened weight $W^{\uparrow(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$

Base weight $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$

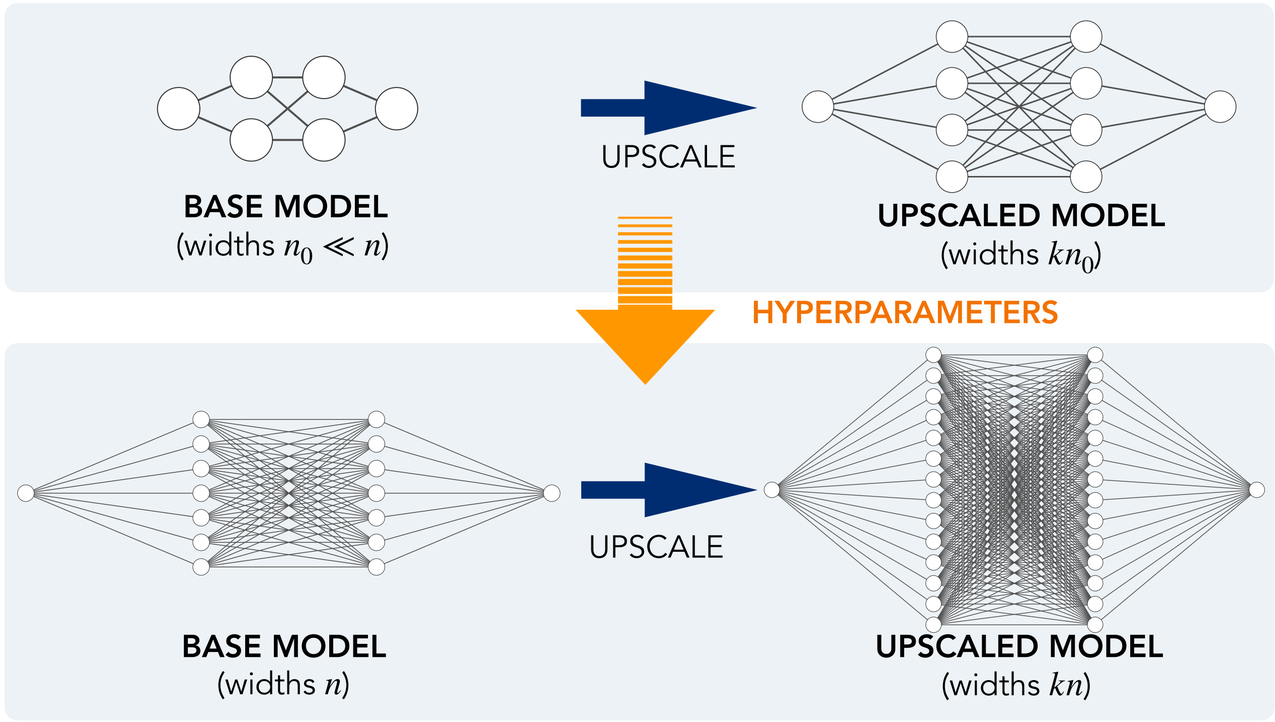
$$\begin{bmatrix} W^{(\ell)}_{1,1} & \cdots & W^{(\ell)}_{1,n_{\ell-1}} \\ \vdots & \ddots & \vdots \\ W^{(\ell)}_{n_\ell,1} & \cdots & W^{(\ell)}_{n_\ell,n_{\ell-1}} \end{bmatrix} \longrightarrow k_{\ell-1}^{-1} \cdot \begin{bmatrix} W^{(\ell)}_{1,1}\ \text{block} & \cdots & W^{(\ell)}_{1,n_{\ell-1}}\ \text{block} \\ \vdots & \ddots & \vdots \\ W^{(\ell)}_{n_\ell,1}\ \text{block} & \cdots & W^{(\ell)}_{n_\ell,n_{\ell-1}}\ \text{block} \end{bmatrix}$$

$k_{\ell-1}$ cols, k_ℓ rows

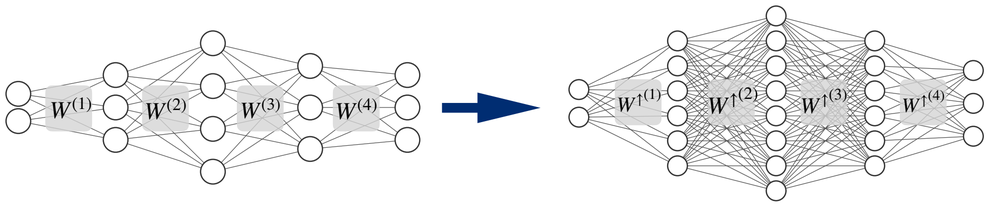
- **Critical gap:** Tuning hyperparameters is expensive.

Our contribution:

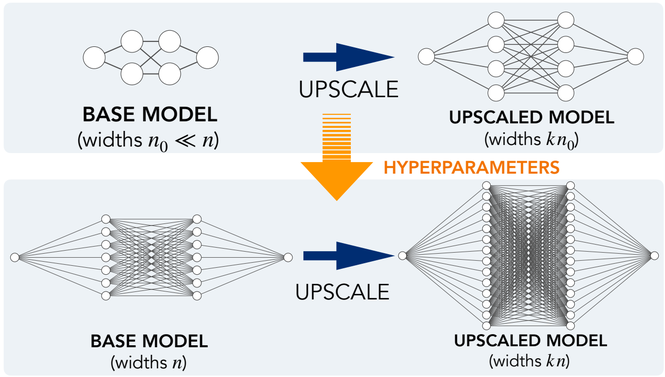
a **theoretically-grounded** upscaling method (with respect to widths)
that allows zero-shot **hyperparameter transfer** based on μP theory.

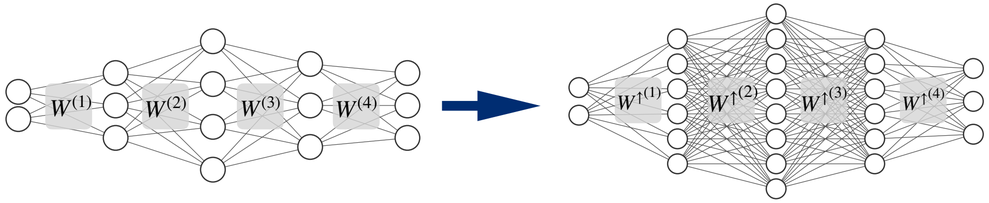


Our proposal: μpscaling

Step 1: Widen	Duplicate & rescale weights → a functionally equivalent wider model 
Step 2: Add noise	Symmetry breaking to exploit additional model capacity. Noise follows the the same scaling that μP prescribes for **random init**, controlled by $\overline{\sigma_\Delta}$.
Step 3: Train	Train the upscaled model with μP prescribed hyperparameters, with LR controlled by $\overline{\gamma}$.

Our proposal: μpscaling



Step 0: **Hyperparameter tuning**	Run Steps 1-3 on **a small upscaling system** with different choices of $\overline{\sigma_\Delta}, \bar{\gamma}$. Choose the values that minimizes training loss.
Step 1: Widen	Duplicate & rescale weights → a functionally equivalent wider model 
Step 2: Add noise	Symmetry breaking to exploit additional model capacity. Noise follows the the same scaling that μP prescribes for **random init**, controlled by $\overline{\sigma_\Delta}$.
Step 3: Train	Train the upscaled model with μP prescribed hyperparameters, with LR controlled by $\bar{\gamma}$.

What is μP (Maximal Update Parameterization)?

- Choice of scaling of initialization and hyperparameters wrt **width** n.

Weight type	**Init Var.** $\mathcal{N}(0, B \cdot \overline{\sigma}^2)$	**LR** $\gamma = C \cdot \overline{\gamma}$	**WD** (vanilla) $\lambda = D \cdot \overline{\lambda}$	**WD** (decoupled) $\lambda = \tilde{D} \cdot \overline{\lambda}$	**Extra HPs** $\varepsilon = E \cdot \overline{\varepsilon}$	**Output multiplier**
Vector-like	1	n^{m}	n^{-1}	n^{-m}	n^{-1}	n^{-1}
Matrix-like	n_{in}^{-1}	$n_{\text{out}}^{m} n_{\text{in}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-m}$	n_{out}^{-1}	$--$

What is μP (Maximal Update Parameterization)?

- Choice of scaling of initialization and hyperparameters wrt **width** n.

Weight type	**Init Var.** $\mathcal{N}(0, B \cdot \overline{\sigma}^2)$	**LR** $\gamma = C \cdot \overline{\gamma}$	**WD** (vanilla) $\lambda = D \cdot \overline{\lambda}$	**WD** (decoupled) $\lambda = \tilde{D} \cdot \overline{\lambda}$	**Extra HPs** $\varepsilon = E \cdot \overline{\varepsilon}$	**Output multiplier**
Vector-like	1	n^m	n^{-1}	n^{-m}	n^{-1}	n^{-1}
Matrix-like	n_{in}^{-1}	$n_{\text{out}}^{m} n_{\text{in}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-m}$	n_{out}^{-1}	$--$

- Consequences
  - Optimal feature learning in the **infinite-width limit** ($n \to \infty$).
  - Activations and their updates stay $\Theta(1)$ throughout the training.
  - Allows **hyperparameter transfer**.

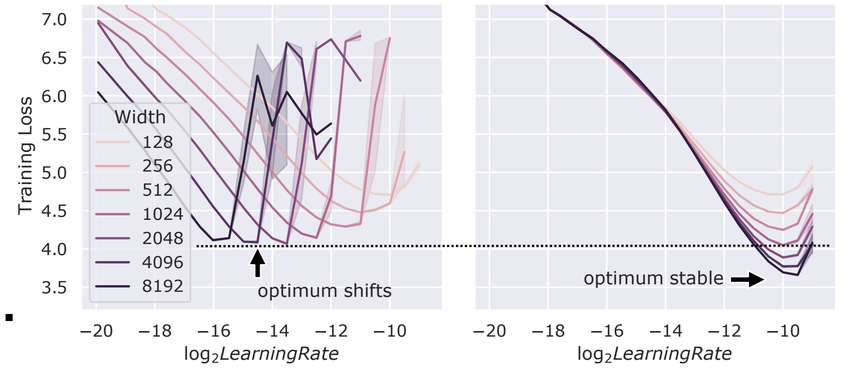
PyTorch default μP

What is μP (Maximal Update Parameterization)?

- Choice of scaling of initialization and hyperparameters wrt **width** n.

Weight type	**Init Var.** $\mathcal{N}(0, B \cdot \overline{\sigma}^2)$	**LR** $\gamma = C \cdot \overline{\gamma}$	**WD** (vanilla) $\lambda = D \cdot \overline{\lambda}$	**WD** (decoupled) $\lambda = \tilde{D} \cdot \overline{\lambda}$	**Extra HPs** $\varepsilon = E \cdot \overline{\varepsilon}$	**Output multiplier**
Vector-like	1	n^m	n^{-1}	n^{-m}	n^{-1}	n^{-1}
Matrix-like	n_{in}^{-1}	$n_{\text{out}}^{m} n_{\text{in}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-1}$	$n_{\text{in}} n_{\text{out}}^{-m}$	n_{out}^{-1}	$--$

- Consequences
  - Optimal feature learning in the **infinite-width limit** ($n \to \infty$).
  - Activations and their updates stay $\Theta(1)$ throughout the training.
  - Allows **hyperparameter transfer**.
- *A priori* does not apply to the context of model upscaling.



PyTorch default μP

We extend μP to the context of model upscaling

Key result: Our algorithm puts the entire training trajectory (base model training + upscaling + upscaled model training) in the optimal feature learning regime.

We extend μP to the context of model upscaling

Key result: Our algorithm puts the entire training trajectory (base model training + upscaling + upscaled model training) in the optimal feature learning regime.

Property 1.

Noise injection is provably “safe” and “useful” in the infinite-width limit.

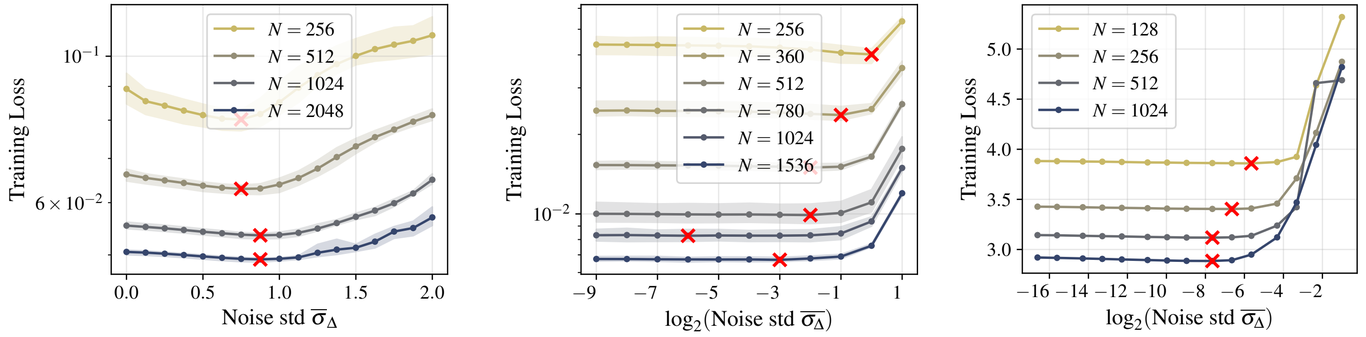
We extend μP to the context of model upscaling

Key result: Our algorithm puts the entire training trajectory (base model training + upscaling + upscaled model training) in the optimal feature learning regime.

Property 2.

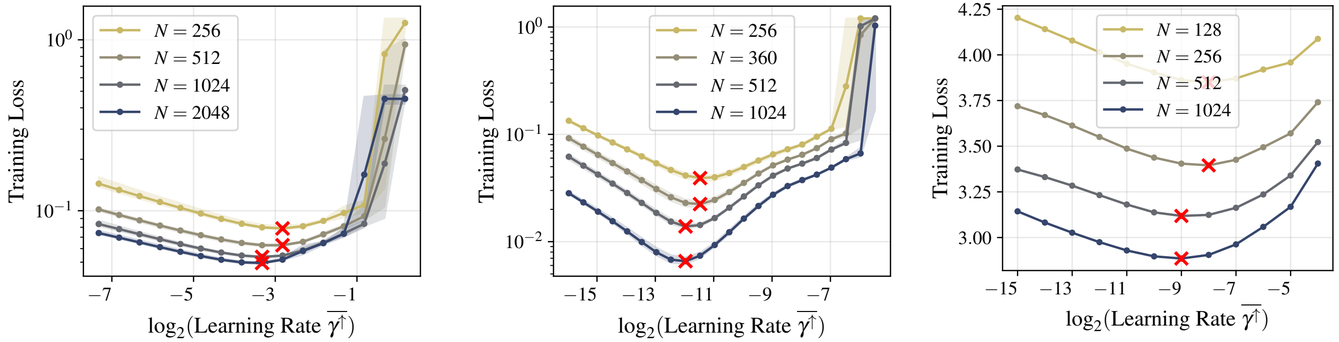
Hyperparameter transfer applies to upscaling.

(a) Noise sweep with fixed LR



MLP (SGD) | MLP (AdamW) | GPT-2 (AdamW)

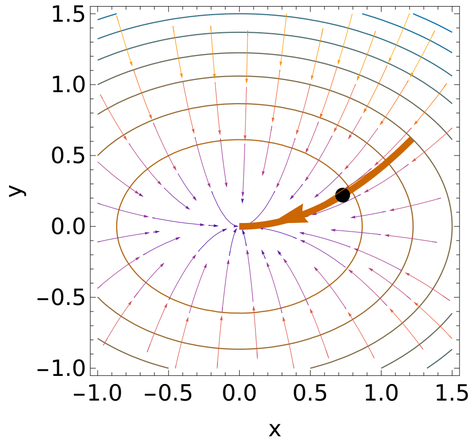
(b) LR sweep with fixed noise



MLP (SGD) | MLP (AdamW) | GPT-2 (AdamW)

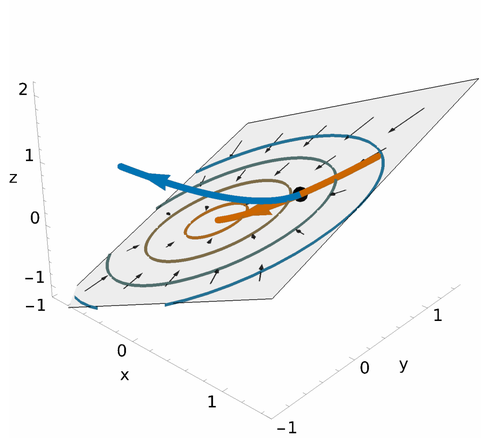
Property 3.

At zero noise, the upscaled model is exactly equivalent to the base model.



→ training trajectory of the base model

UPSCALE



→ equivalent widened model

→ upscaled model with added noise

Experiments

MLP (AdamW) on Forest Cover Type	ResNet (SGD) on CIFAR-100	GPT-2 (AdamW) on FineWeb
width $500 \to 2000$ tuning on width $100 \to 400$ tuning speedup $23.6\times$ FLOPs	$2\times \to 4\times$ model tuning on $0.5\times \to 1\times$ model tuning speedup $15.8\times$ FLOPs	width $1920 \to 3840$ tuning on width $192 \to 384$ tuning speedup $48.2\times$ FLOPs