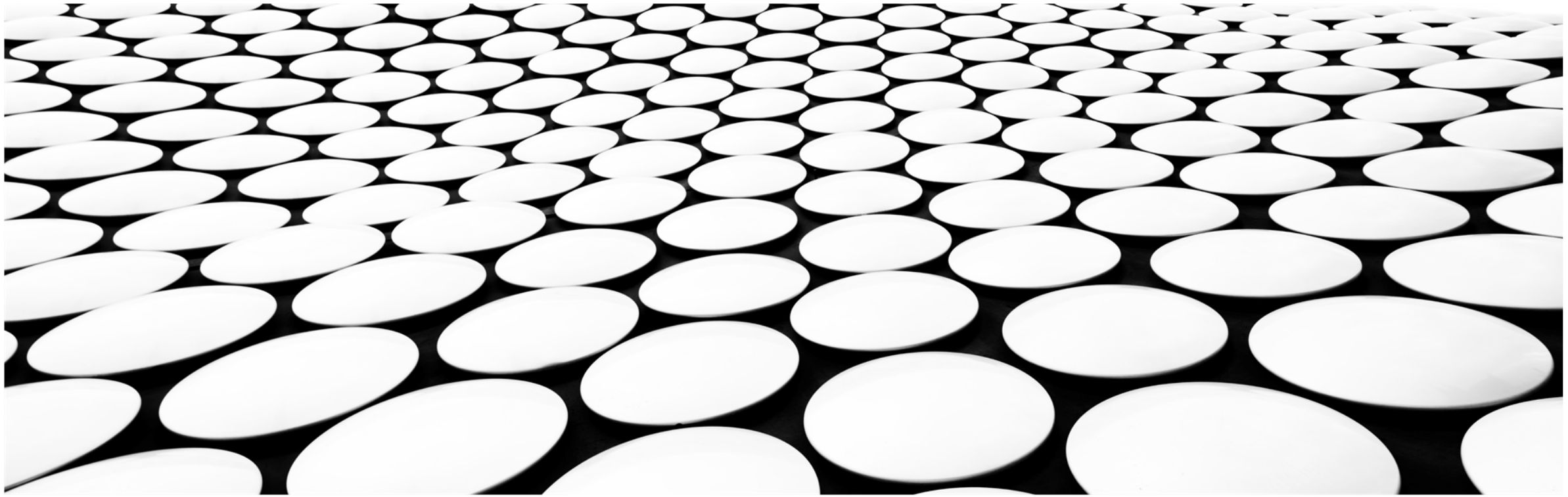


Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer

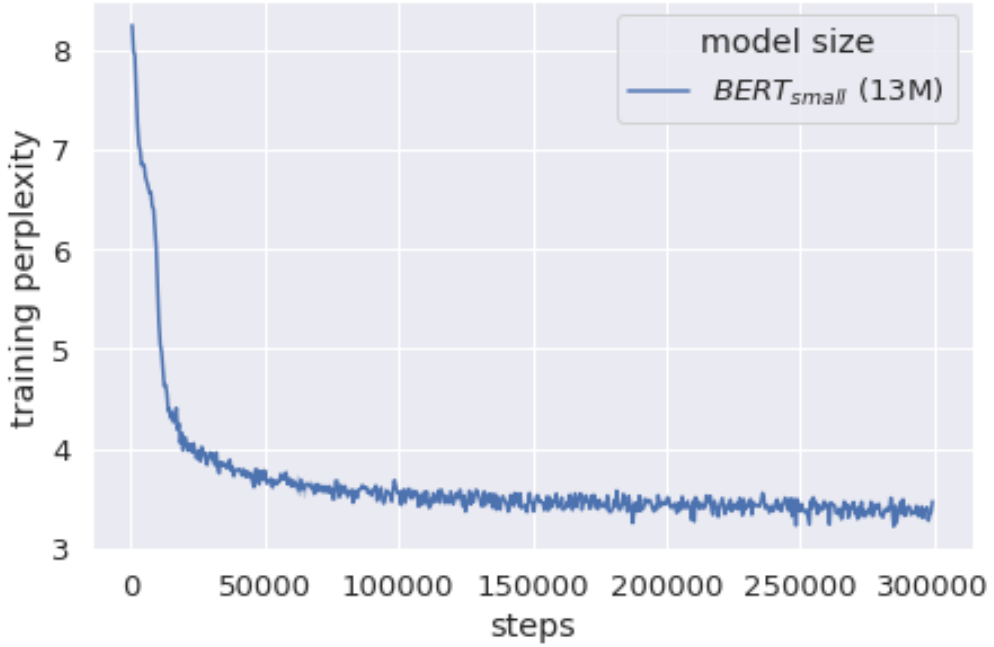
Greg Yang

in Collaboration with Edward Hu, Igor Babuschkin, Szymon Sidor, David Farhi, Nick Ryder,

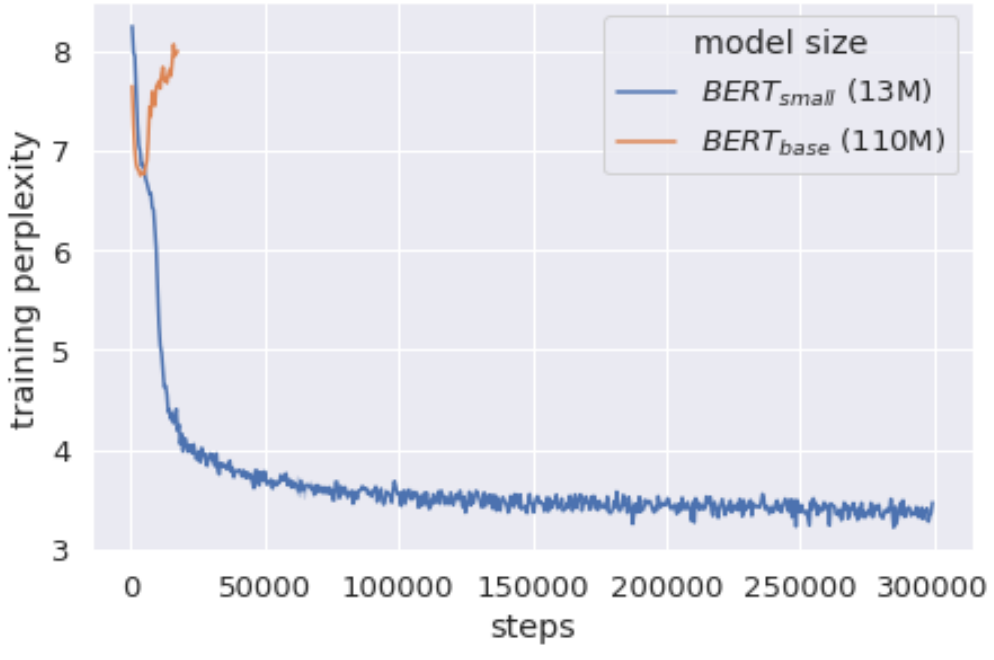
Jakub Pachocki, Xiaodong Liu, Weizhu Chen, Jianfeng Gao

Microsoft OpenAI

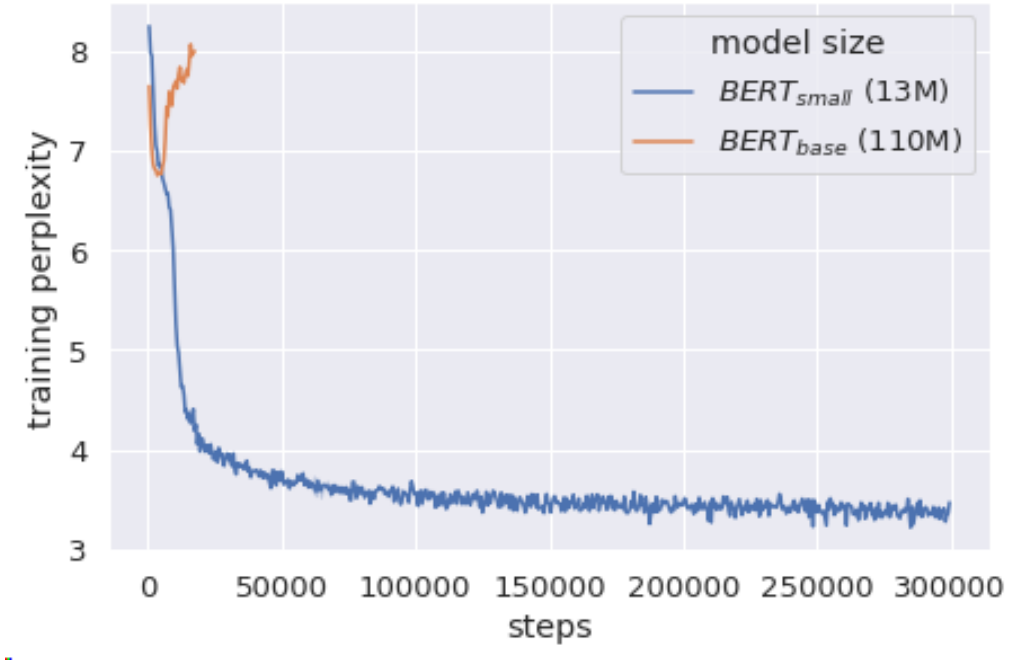
Explore with a small model



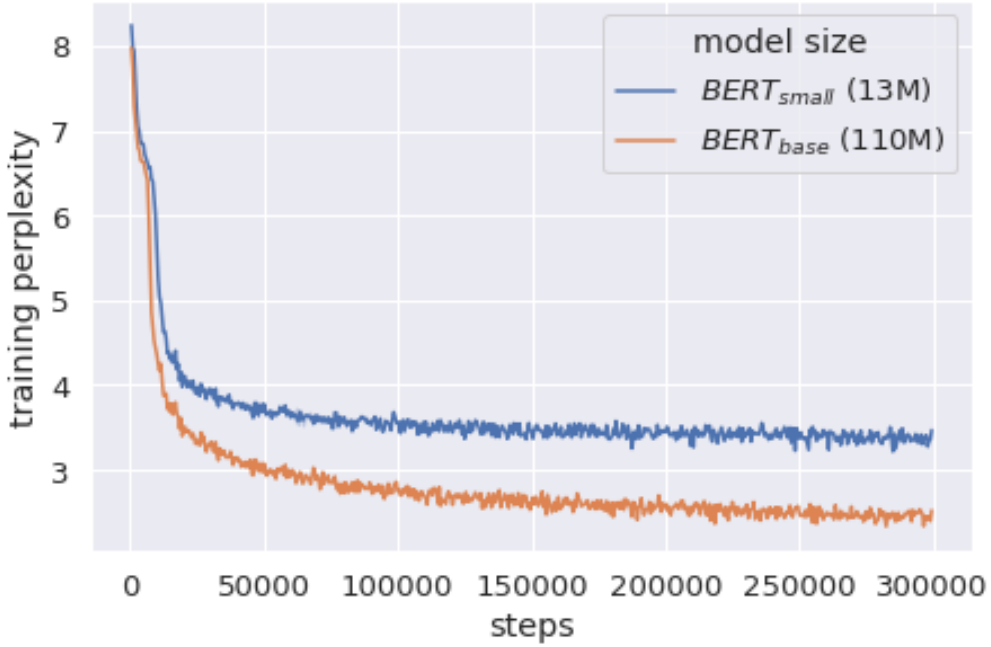
Model fails to train when scaled up with the same hyperparameters



An Example of
What *This Work* Allows You To Do



Before



After

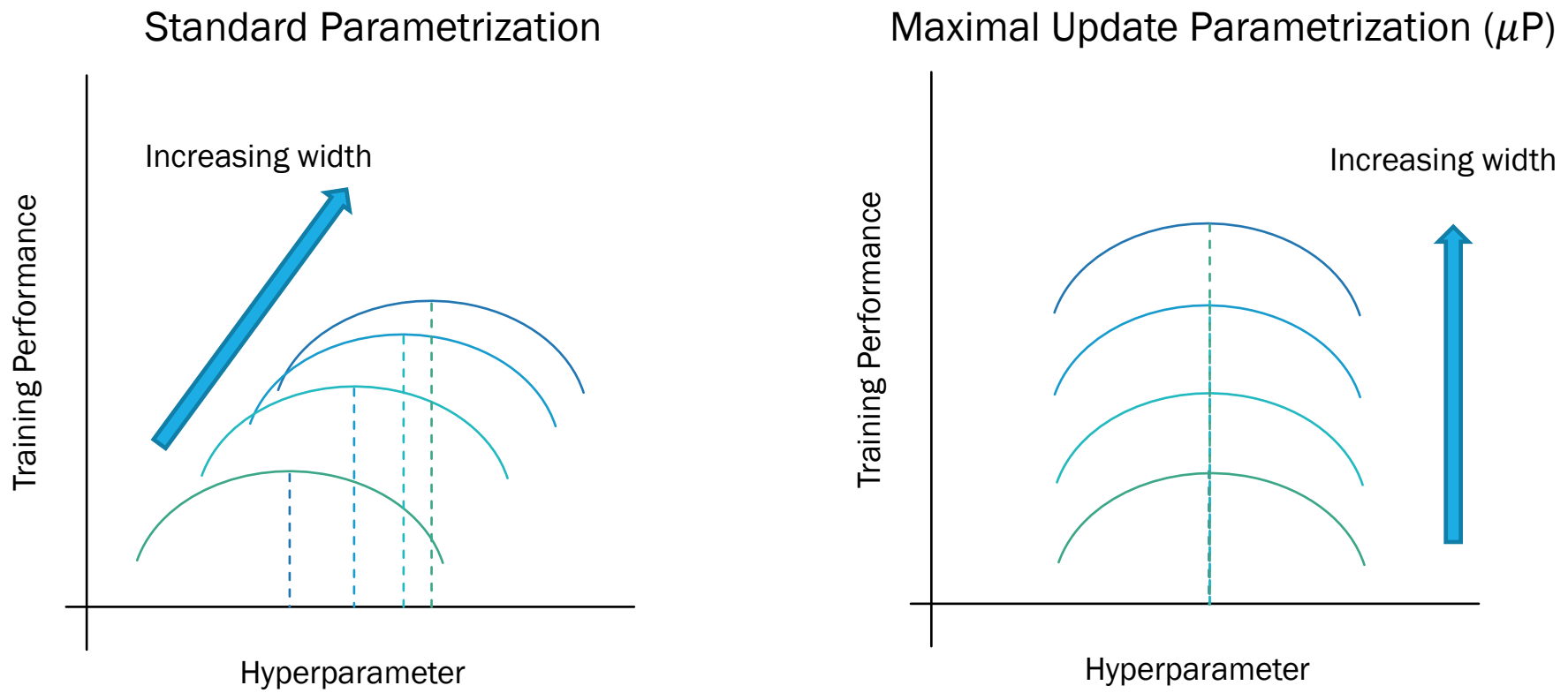
Enter *Maximal Update Parametrization*, abbreviated μP

A principled way of scaling initialization and learning rate with width

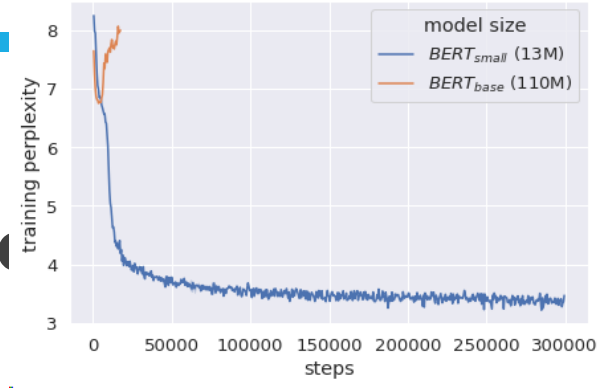
	Input weights & all biases	Output weights	Hidden weights
Init. Var.	$1/\text{fan\_in}$	$1/\text{fan\_in}^2$ $(1/\text{fan\_in})$	$1/\text{fan\_in}$
SGD LR	fan_out (1)	$1/\text{fan\_in}$ (1)	1
Adam LR	1	$1/\text{fan\_in}$ (1)	$1/\text{fan\_in}$ (1)

Note: focus on scaling with fan_in or fan_out; everything else is a tunable constant

Key Property of μP: Hyperparameter Stability

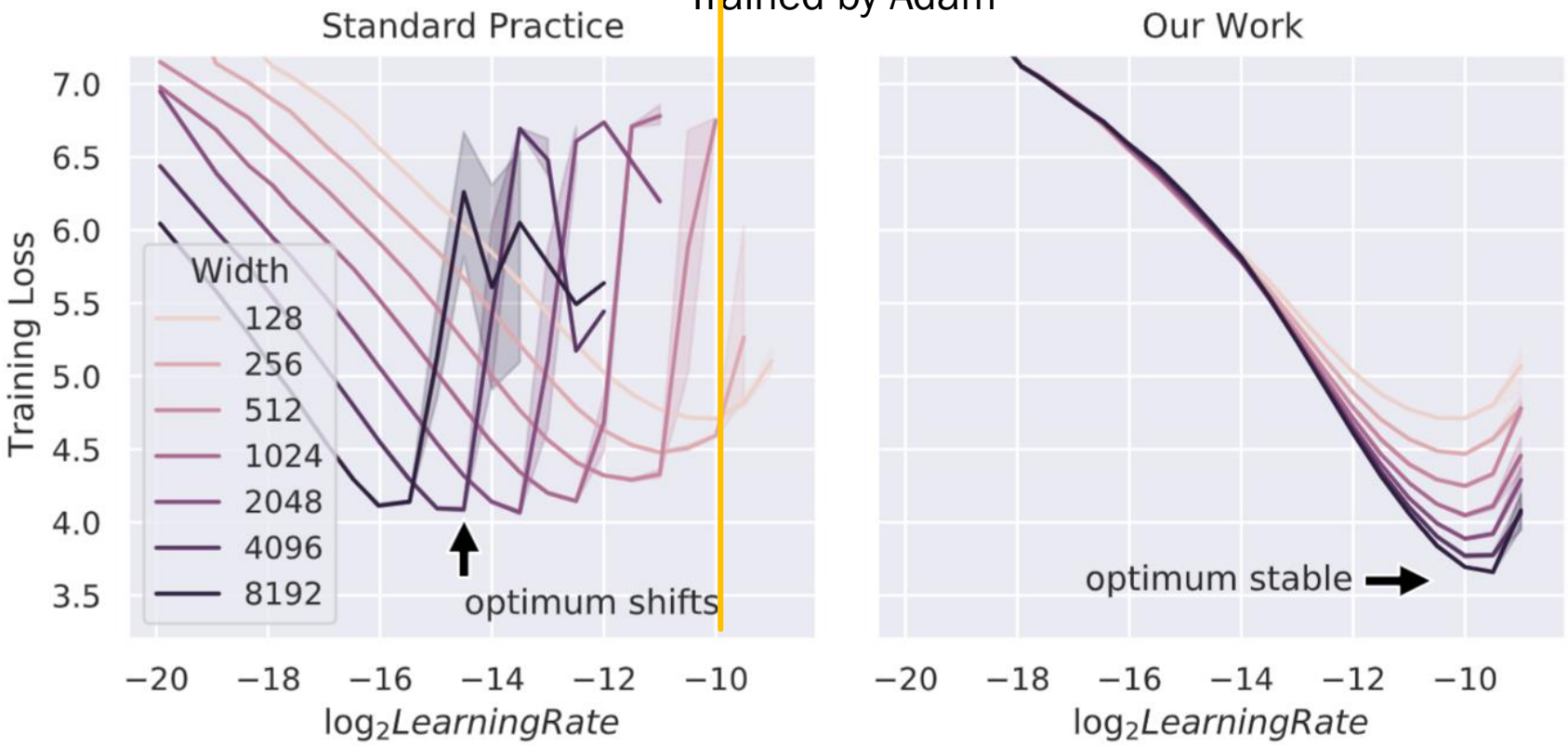


Key Property of μP: Hype ity

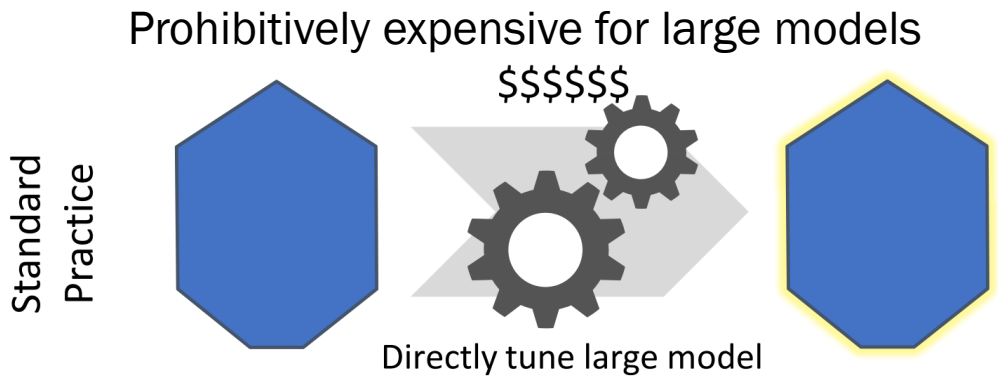


ext-2

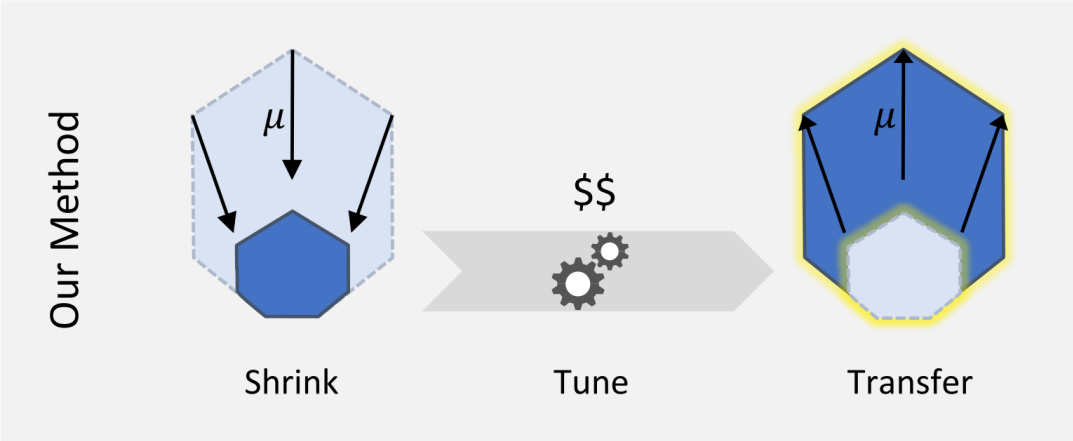
Trained by Adam



μTransfer: Zero-Shot Hyperparameter Transfer



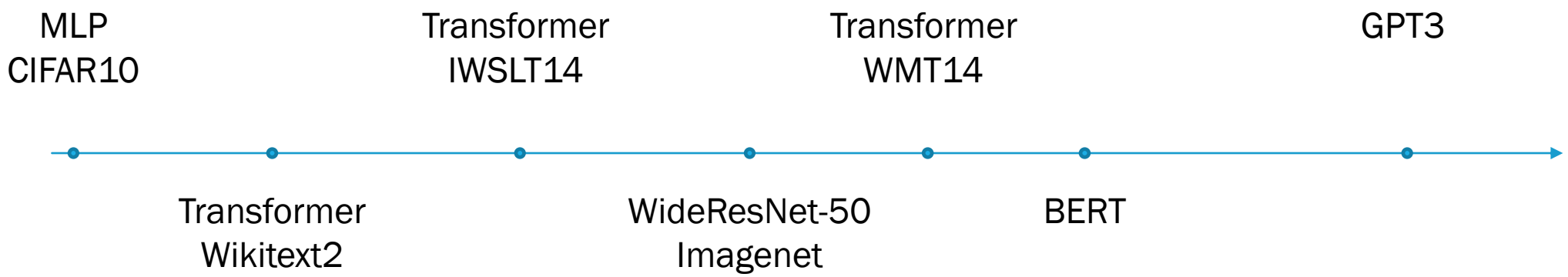
μTransfer



You don't need to tune the small model if you don't want to!
--- You can μTransfer any existing set of hyperparameters you like from a small model to a large model!

"Transfer" = optimal hyperparameter remains stable with model size

CAREFUL VALIDATION ON MODELS OF INCREASING SCALE



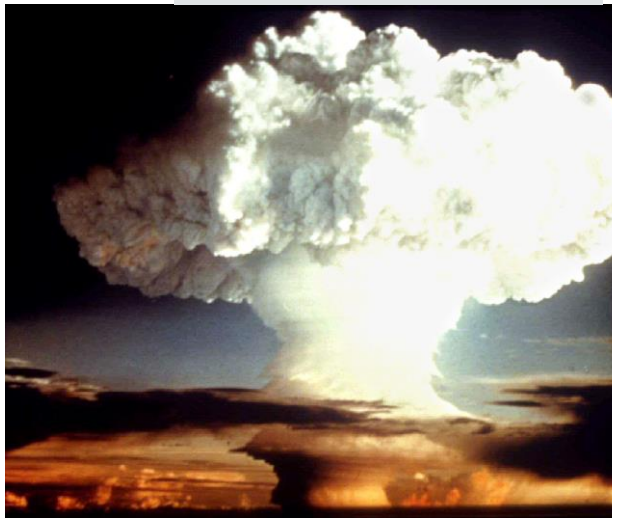
PRACTICAL BENEFITS OF μP

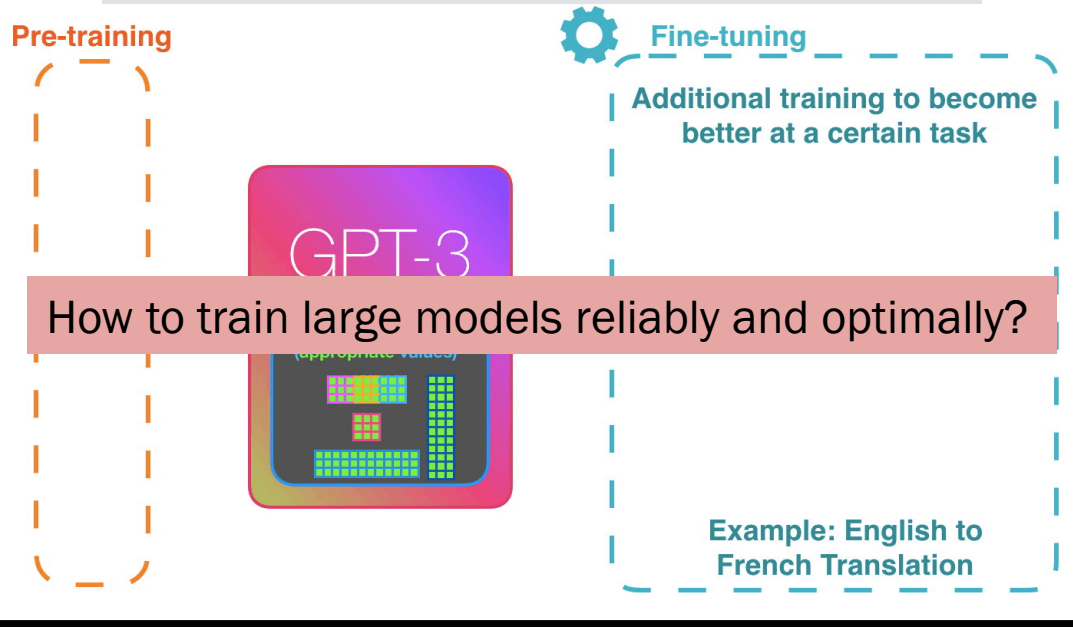
- Preserves hyperparameter optimum across width
- Allows zero-shot hyperparameter transfer
  - Efficient tuning
    - Can tune enormous models only on a single GPU
    - Very fast
  - The only method for tuning large models
- Altogether enable *more reliable scaling-up* of neural networks

WHAT DO THESE HAVE IN COMMON?

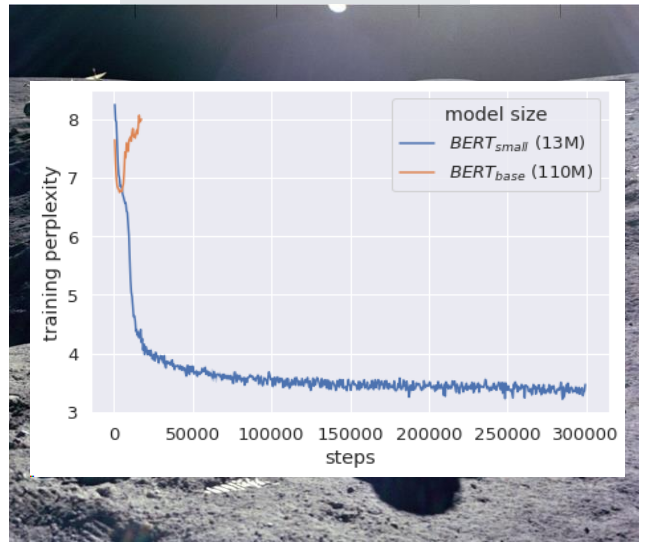
Large pretrained language/vision models

Manhattan Project



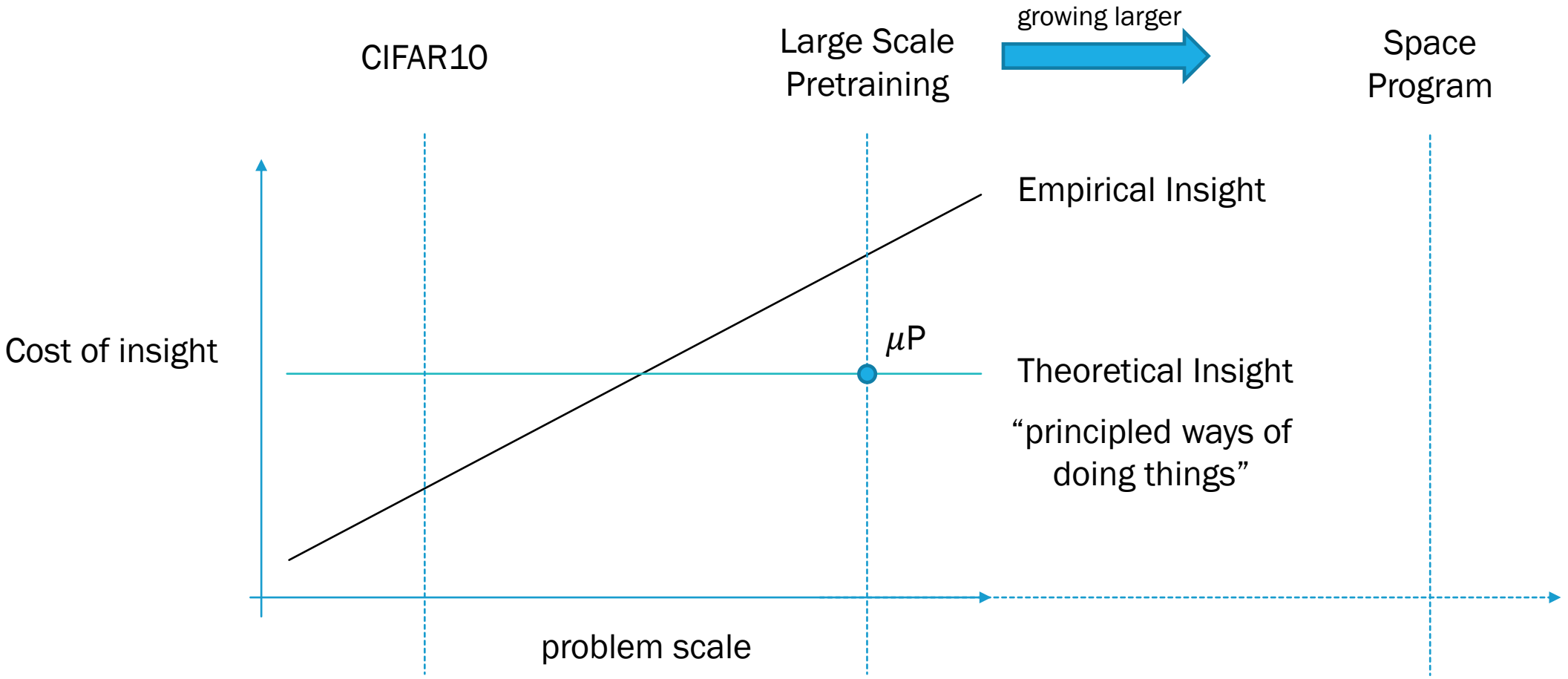


Space Program



- Impressive achievements of humankind
- Each empirical test is very expensive
- Require extensive theoretical calculation first before launching any empirical test

ECONOMY OF INSIGHTS AT SCALE



What is Maximal Update Parametrization?

Which
Hyperparameters
Can Be μTransferred?

Empirical Efficacy of μTransfer

Theoretical Foundation

CONCLUSION

What is Maximal Update Parametrization?

WARMUP: MLP

$$n = width,\ d_{in} = data\ dimension$$

Standard Parametrization (SP)

$$f(\xi) = W^{3\top}\phi(W^{2\top}\phi(W^{1\top}\xi + b^1) + b^2)$$

$$\text{with init.} \quad W^1 \sim \mathcal{N}(0, 1/d_{in}),\ W^{\{2,3\}} \sim \mathcal{N}(0, 1/n),\ b^{\{1,2\}} = 0, \quad \text{LR } \eta$$

μ-Parametrization (μP)

$$\text{initialize} \quad W^1 \sim \mathcal{N}(0, 1/d_{in}),\ W^2 \sim \mathcal{N}(0, 1/n),\ W^3 \sim \mathcal{N}(0, 1/n^2),\ b^{\{1,2\}} = 0$$

$$\text{with SGD learning rates} \quad \eta_{W^1} = \eta_{b^1} = \eta_{b^2} = \eta n,\ \eta_{W^2} = \eta,\ \eta_{W^3} = \eta n^{-1}.$$

WARMUP: MLP

$n = width,\ d_{in} = data\ dimension$

μ-Parametrization (μP)

$$f(\xi) = W^{3\top}\phi(W^{2\top}\phi(W^{1\top}\xi + b^1) + b^2)$$

$$\text{initialize} \quad W^1 \sim \mathcal{N}(0, 1/d_{in}),\ W^2 \sim \mathcal{N}(0, 1/n),\ W^3 \sim \mathcal{N}(0, 1/n^2),\ b^{\{1,2\}} = 0$$

$$\text{with SGD learning rates} \quad \eta_{W^1} = \eta_{b^1} = \eta_{b^2} = \eta n,\ \eta_{W^2} = \eta,\ \eta_{W^3} = \eta n^{-1}.$$

Correlated during training
→ Law of Large Numbers!
→ Not Central Limit!

Why?

Standard Param (SP)
- didn't think about correlation during training!

$$f(\xi) = \sum_{\alpha=1}^{n} W_\alpha^3 \phi(\cdots)_\alpha \qquad = \Theta(n)$$

$\Theta(1)$

$\Theta(1)$

$$f(\xi) = \sum_{\alpha=1}^{n} W_\alpha^3 \phi(\cdots)_\alpha \qquad = \Theta(1)$$

$\Theta(1)$

$\Theta(1/n)$

ensures

μ-PARAMETRIZATION IN GENERAL

	Input weights & all biases	Output weights	Hidden weights
Init. Var.	$1/\text{fan\_in}$	$1/\text{fan\_in}^2$ $(1/\text{fan\_in})$	$1/\text{fan\_in}$
SGD LR	fan_out (1)	$1/\text{fan\_in}$ (1)	1
Adam LR	1	$1/\text{fan\_in}$ (1)	$1/\text{fan\_in}$ (1)

Note: focus on scaling with fan_in or fan_out; everything else is a tunable constant

For Transformers, also use $1/d$ instead of $1/\sqrt{d}$ attention, i.e. the attention logit is calculated as $k^\top q/d$ instead of $k^\top q/\sqrt{d}$ where query q and key k have dimension d.

Which Hyperparameters Can Be μTransferred?

Table 1: **Hyperparameters That Can Be μTransferred, Not μTransferred, or μTransferred Across,** with a few caveats discussed in Section 5.1. * means *empirically validated only* on Transformers, while all others additionally have theoretical justification.

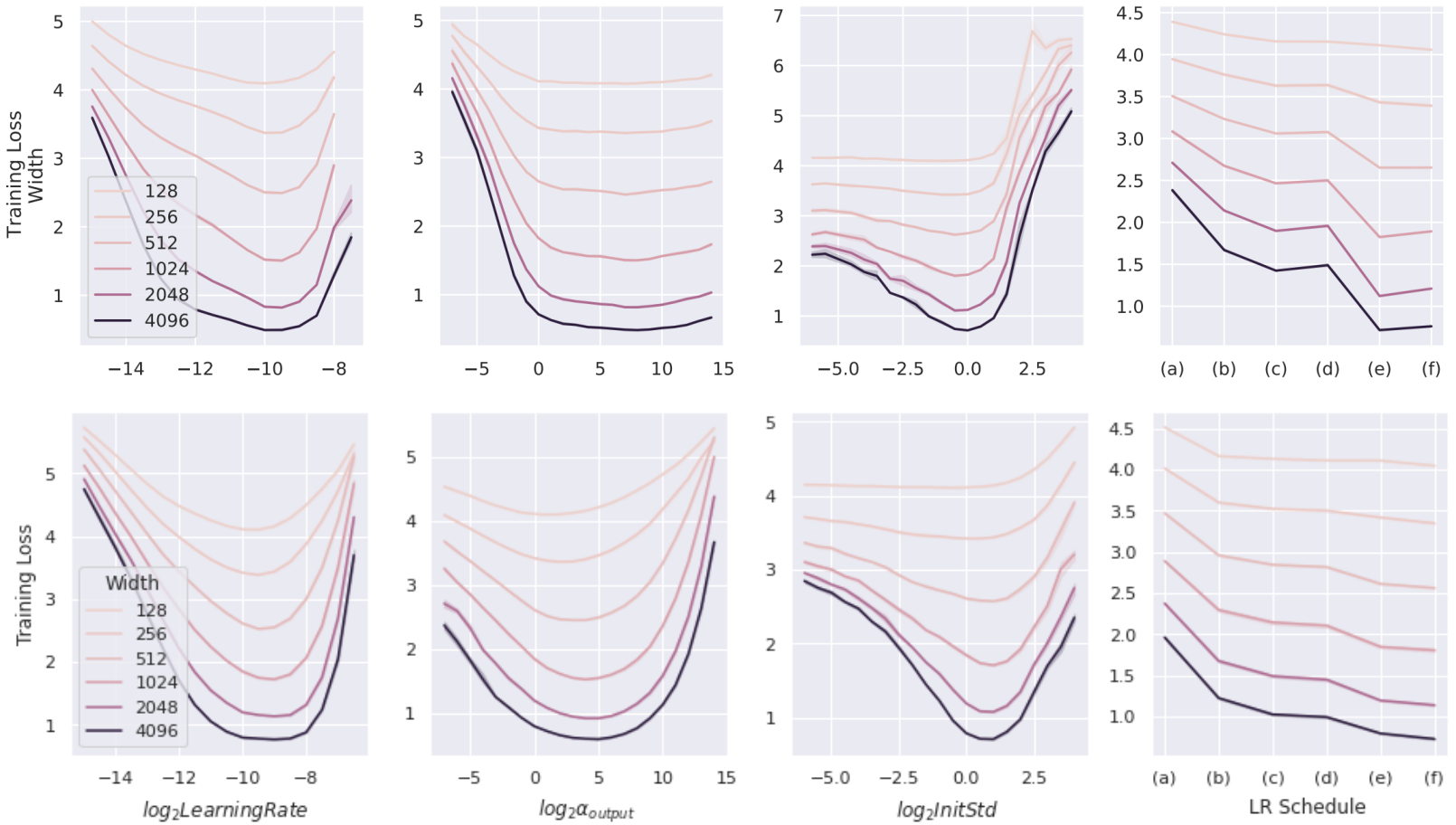
μTransferable	Not μTransferable	μTransferred *Across*
optimization related, init, parameter multipliers, etc	regularization (dropout, weight decay, etc)	width, depth*, batch size*, training time*, seq length*

- Why not regularization?
  - The amount of regularization (for the purpose of controlling overfitting) naturally depends on both the model size and data size
    - so we should not expect transfer to work if the parametrization only depends on model size (width)
  - Therefore, if regularization is the bottleneck of performance of a large model, μP may not help with that
  - Luckily, for large scale pretraining, we still have too much data so regularization is not a bottleneck
- While transfer across width is justified by theory, we empirically explore transferring across other scale dimensions like depth, batch size, training time, seq length.
  - But without theoretical understanding, we cannot be as confident that these results hold beyond the *reasonable scales* and the problem settings we did experiments in

Verifying Transfer Across Width

postLN Transformer

preLN Transformer



What if We Change Width Ratio?

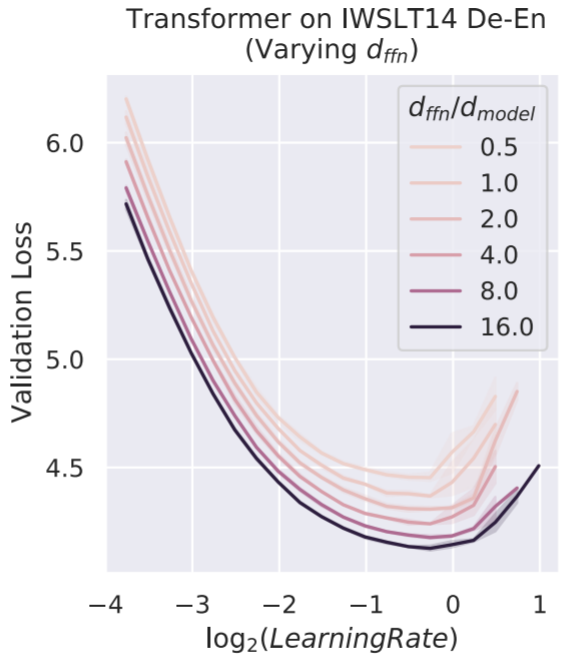


Figure 12: Learning rate landscape in μP is stable even if we vary d_{ffn} by a factor of 32, fixing d_{model}.

You Can Vary Both d_{head} and n_{head}

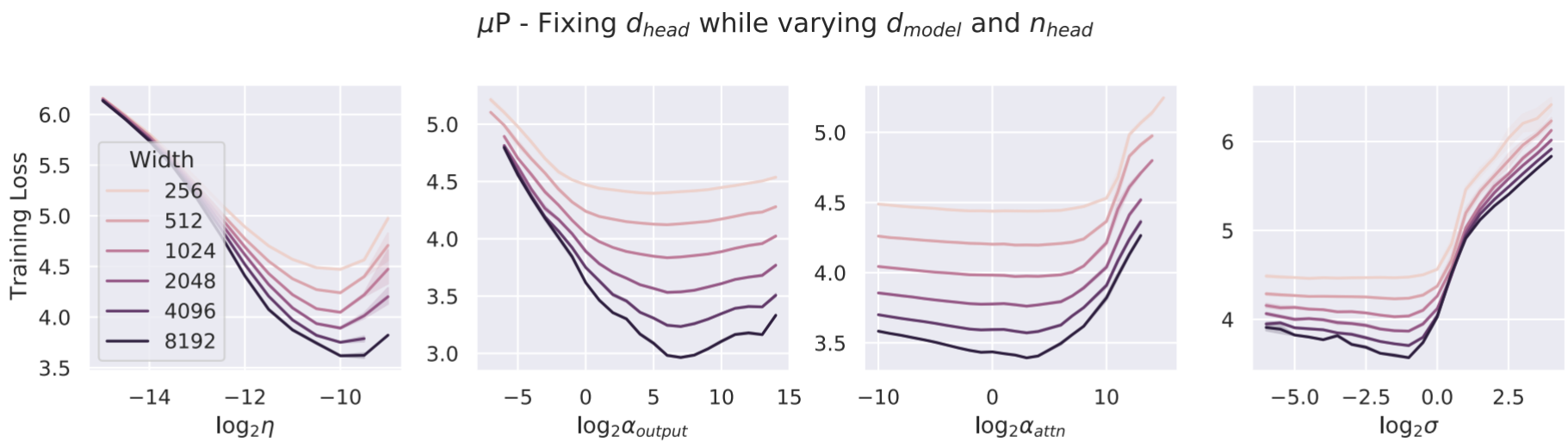


Figure 13: μTransfer across width when we fix d_{head} and vary d_{model} and n_{head}. $\alpha_{output}, \alpha_{attn}$ are multipliers for output and key weights, and σ is initialization standard deviation.

WIDER IS BETTER THROUGHOUT TRAINING

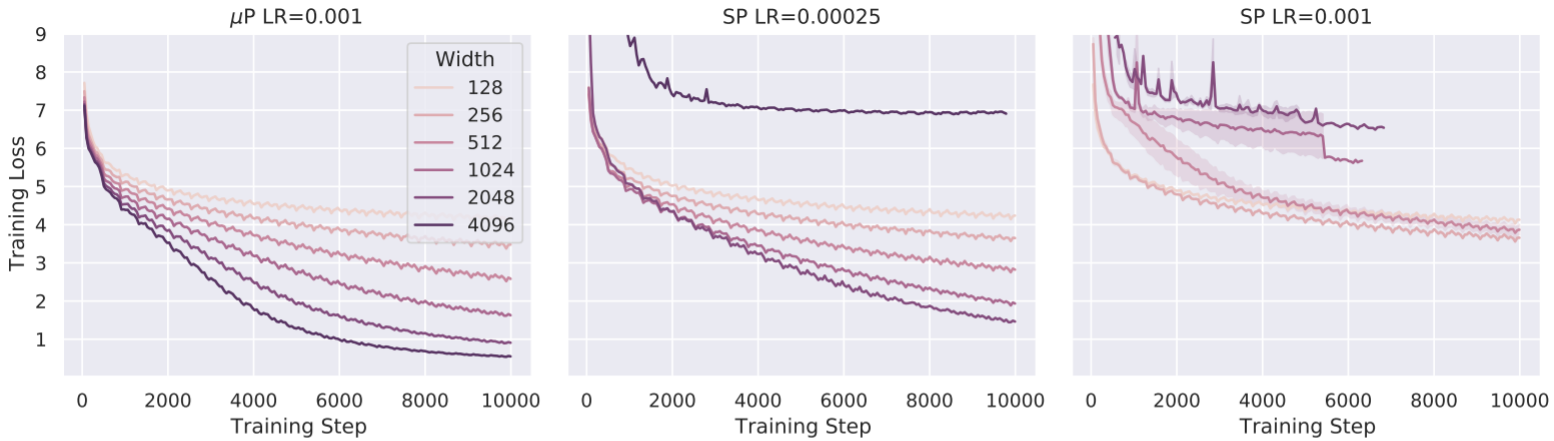


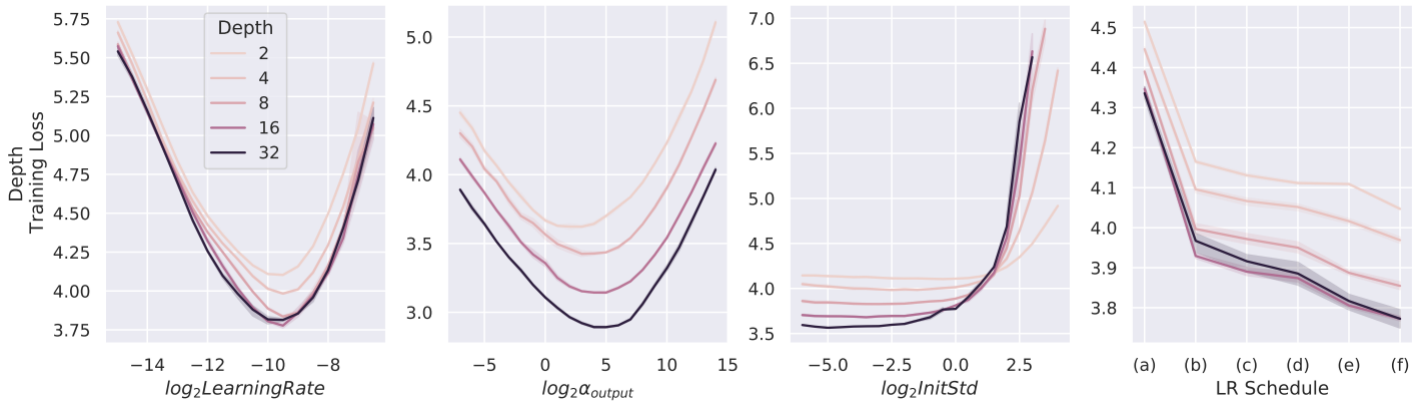
Figure 9: **Wider is always better in training loss under μP, but not in SP, given the same HP.** Learning curves for μP and SP with different learning rates, aggregated over 5 seeds. **(Left)** Wider μP models always achieve better training loss at any time in training. **(Middle)** If using a small learning rate, SP models can appear to do so up to some large width, at which point the pattern fails (at width 2048 in our plot). **(Right)** If using a large learning rate, SP model can do *worse* with width; here the SP model is identical to the μP model in (Left) at width 128.

EXPLORING EMPIRICAL TRANSFER ACROSS DEPTH

postLN
Transformer

😑 Doesn't work

preLN
Transformer

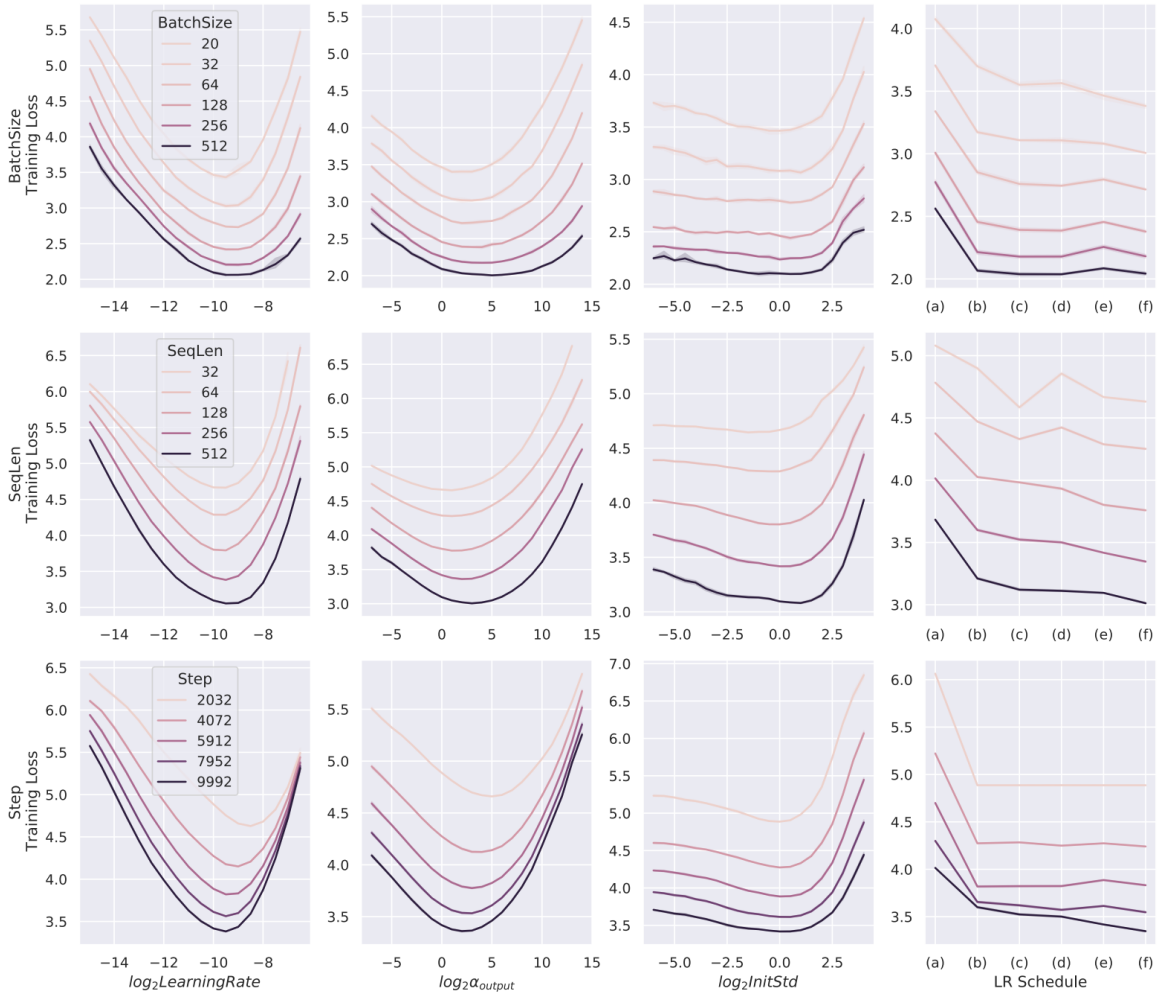


EXPLORING EMPIRICAL TRANSFER ACROSS OTHER DIMENSIONS

Batch Size

Seq Len

Training Time



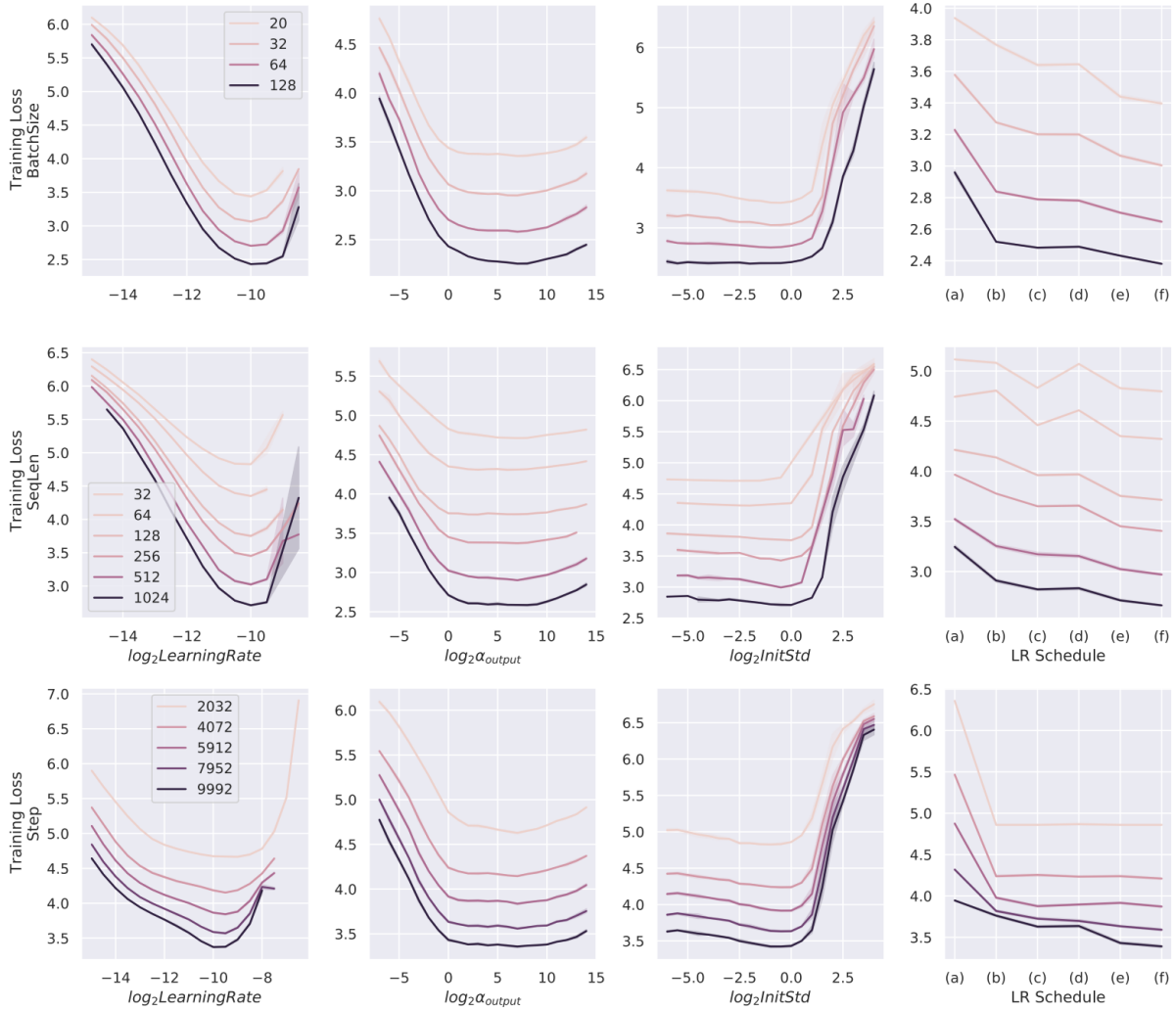
preLN Transformer

EXPLORING EMPIRICAL TRANSFER ACROSS OTHER DIMENSIONS

Batch Size

Seq Len

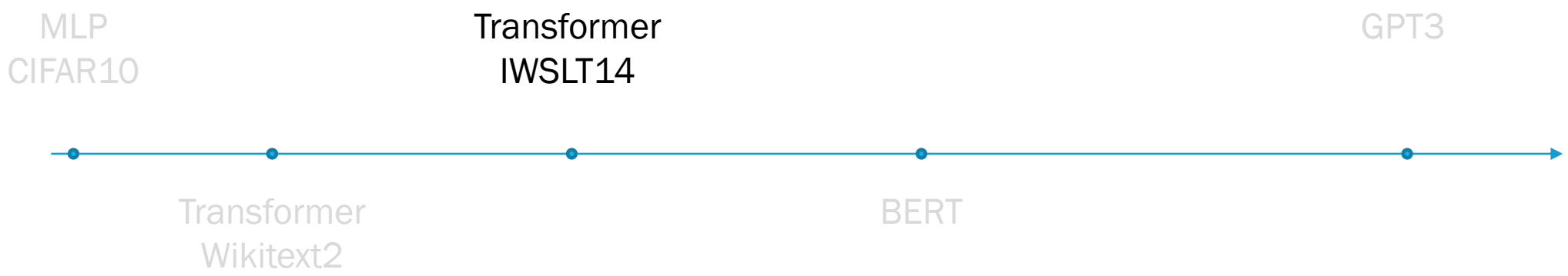
Training Time



postLN
Transformer

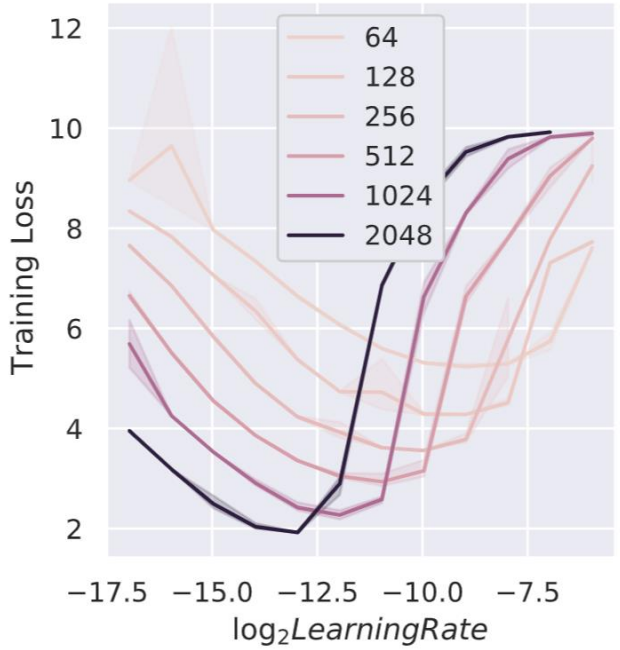
Empirical Efficacy of μTransfer

CAREFUL VALIDATION ON MODELS OF INCREASING SCALE



Transformer on IWSLT14 De-En

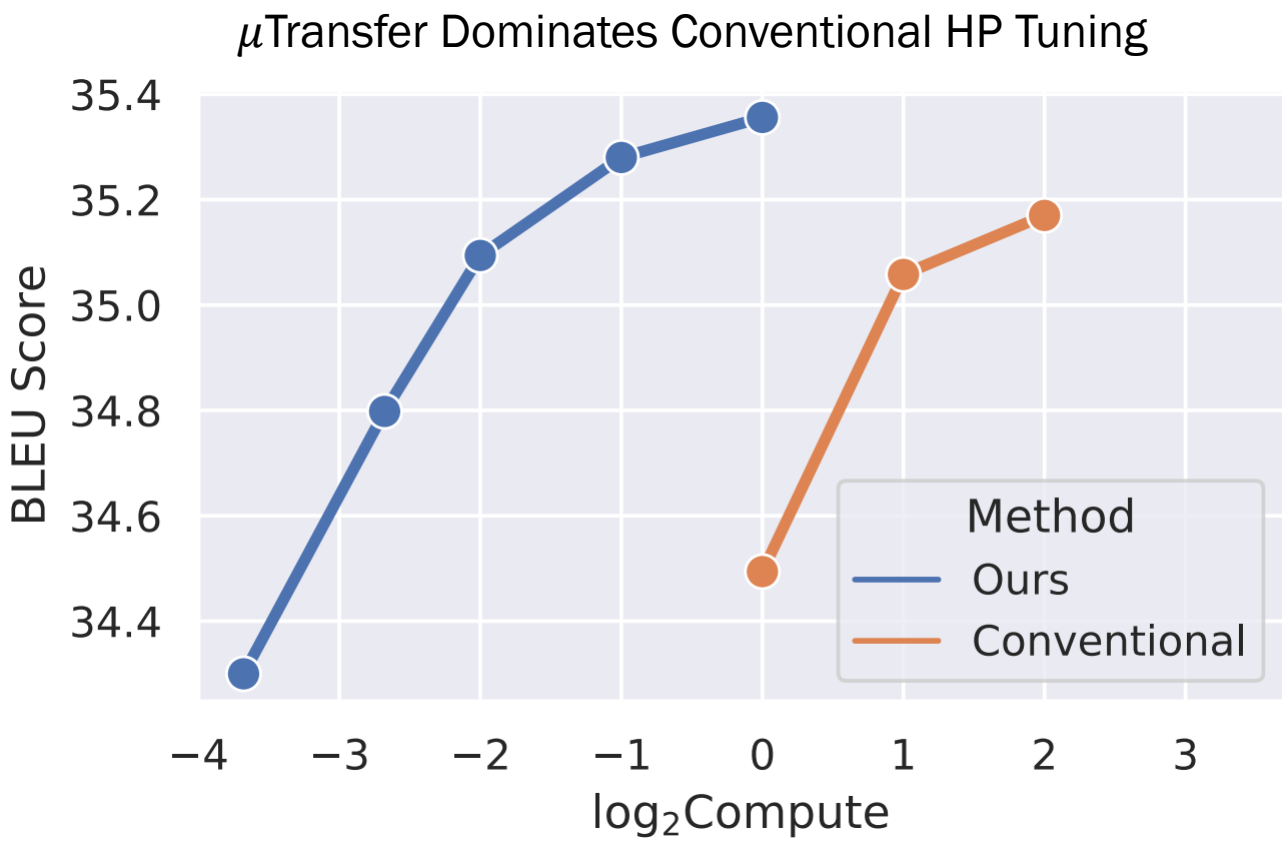
Standard Parametrization



μP + μTransfer

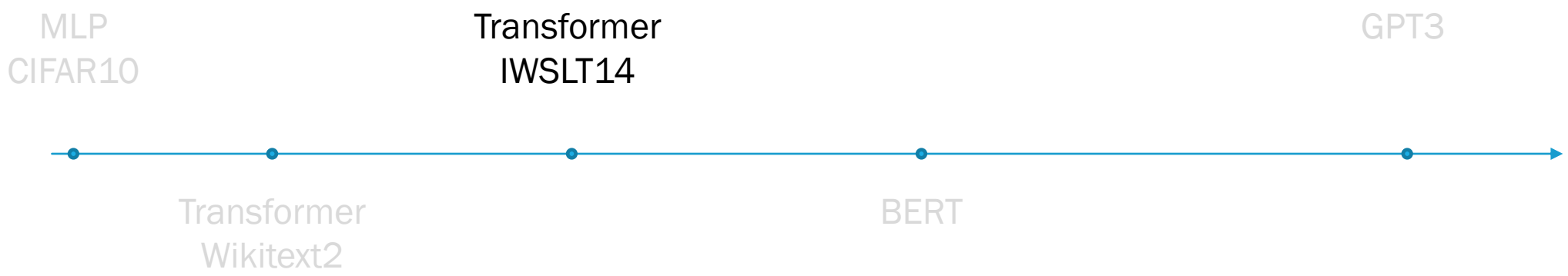
Setup	Total Compute	#Samples	Val. BLEU Percentiles 25	50	75	100
`fairseq` default	-	-	-	-	-	35.40
Tuning on 1x	1x	5	33.62	35.00	35.35	35.45
Naive transfer from 0.25x	1x	64	training diverged			
Transfer from 0.25x	1x	64	**35.27**	**35.33**	**35.45**	**35.53**

Performance-Compute Pareto Frontier



IWSLT Transformer is still fairly small --- efficiency only goes up with model scale!

CAREFUL VALIDATION ON MODELS OF INCREASING SCALE



CAREFUL VALIDATION ON MODELS OF INCREASING SCALE

Tuning BERT with μTransfer

Step 1:

Parameterize BERT in μP

Step 2:

Tune hyperparameters on $\text{BERT}_{\text{SMALL}}$ via random search (256 combinations), training only for 10% of tokens

Step 3:

Copy the best hyperparameter combination to $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$

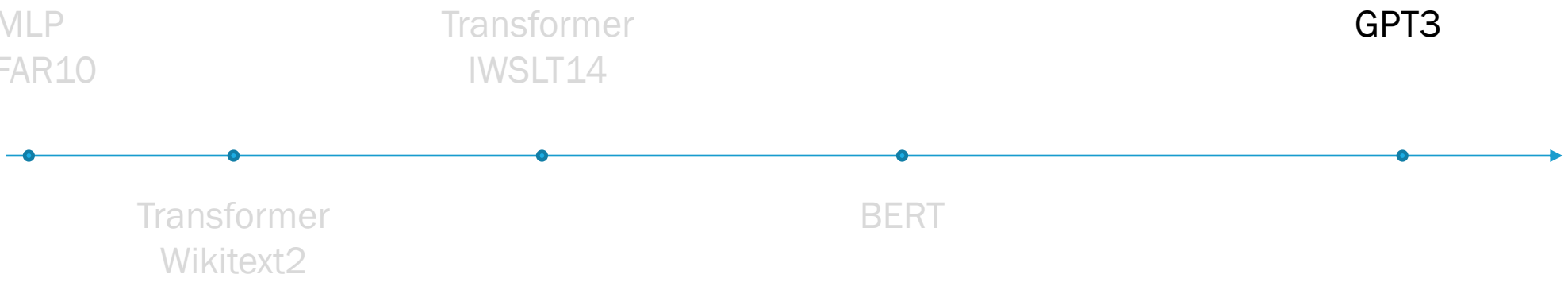
- ✓ Tune once, use for a family of models
- ✓ Only run the large models once

The total tuning cost is equivalent to 1 full training run of BERT-large

Model	Method	Model Speedup	Total Speedup	Test loss	MNLI (m/mm)	QQP
BERT_{base}	Megatron Default	1x	1x	1.995	84.2/84.2	90.6
BERT_{base}	Naive Transfer	4x	40x		training diverged	
BERT_{base}	μTransfer (Ours)	4x	40x	**1.970**	**84.3/84.8**	**90.8**
BERT_{large}	Megatron Default	1x	1x	1.731	86.3/86.2	90.9
BERT_{large}	Naive Transfer	22x	220x		training diverged	
BERT_{large}	μTransfer (Ours)	22x	220x	**1.683**	**87.0/86.5**	**91.4**

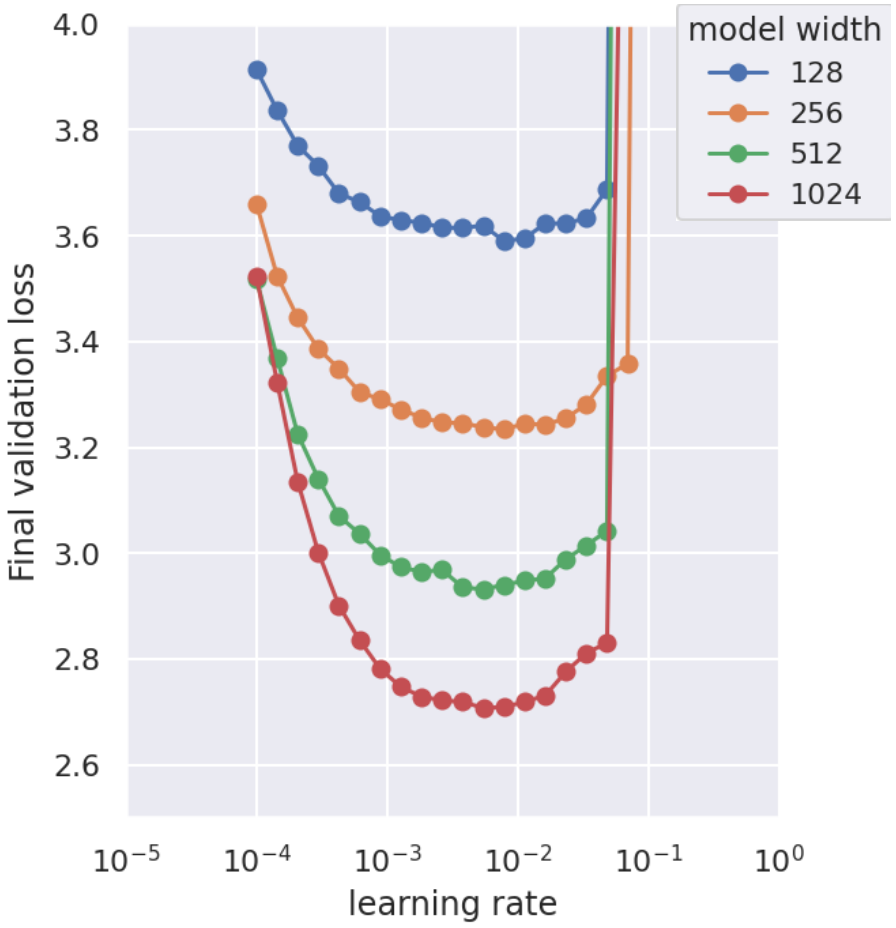
CAREFUL VALIDATION ON MODELS OF INCREASING SCALE

CAREFUL VALIDATION ON MODELS OF INCREASING SCALE

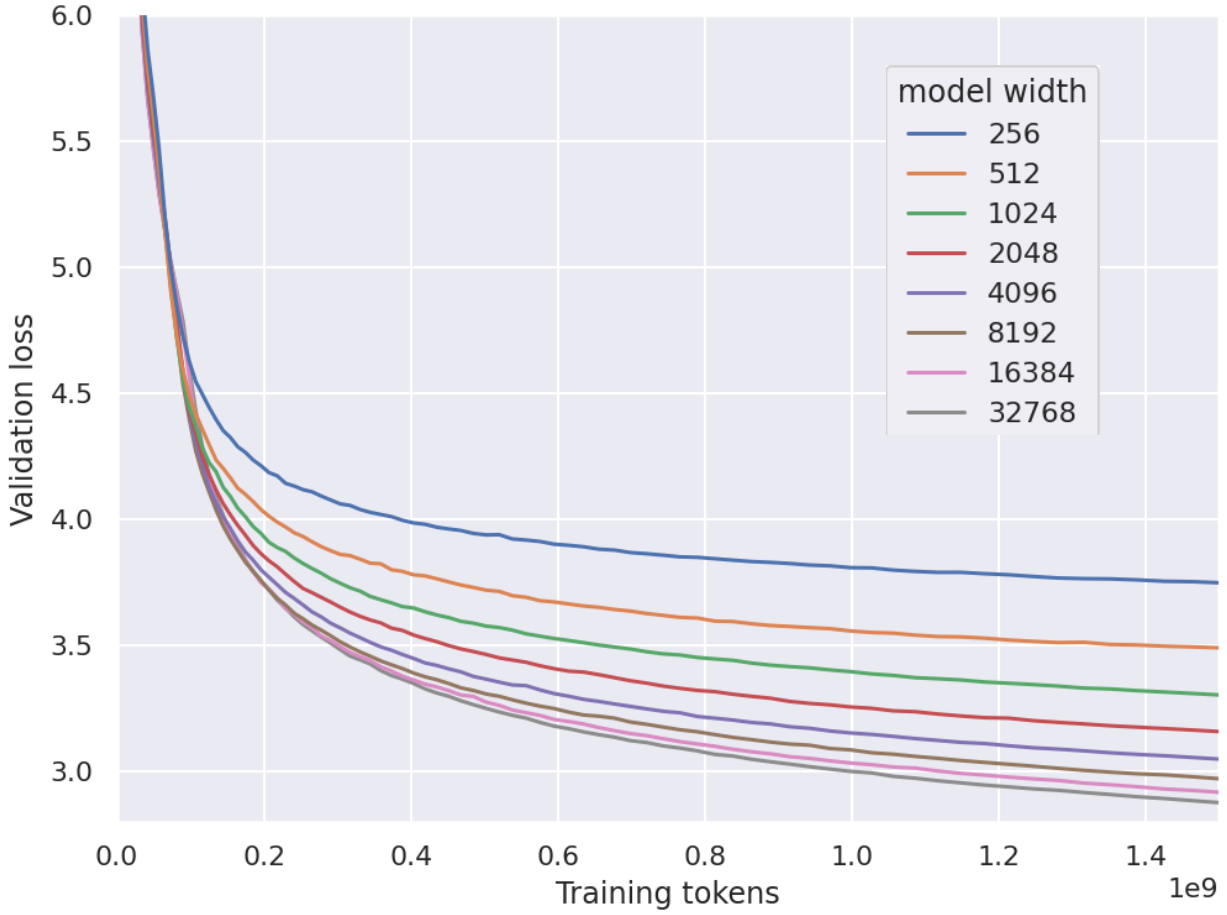


OpenAI GPT-3 Family + μP

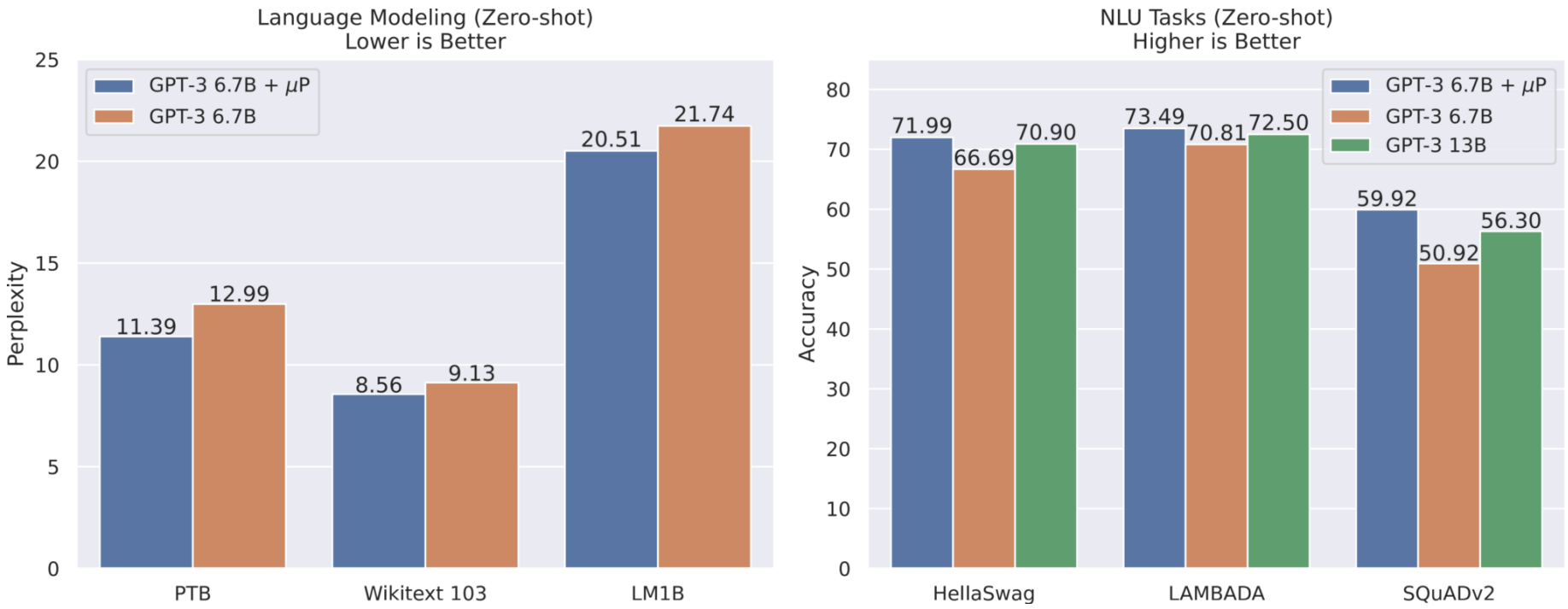
Hyperparameter Optimum is Stable



Wider is Always Better Given the Same HPs



OpenAI GPT-3 6.7B + μTransfer



Total tuning compute budget is only 7% of training budget!!!

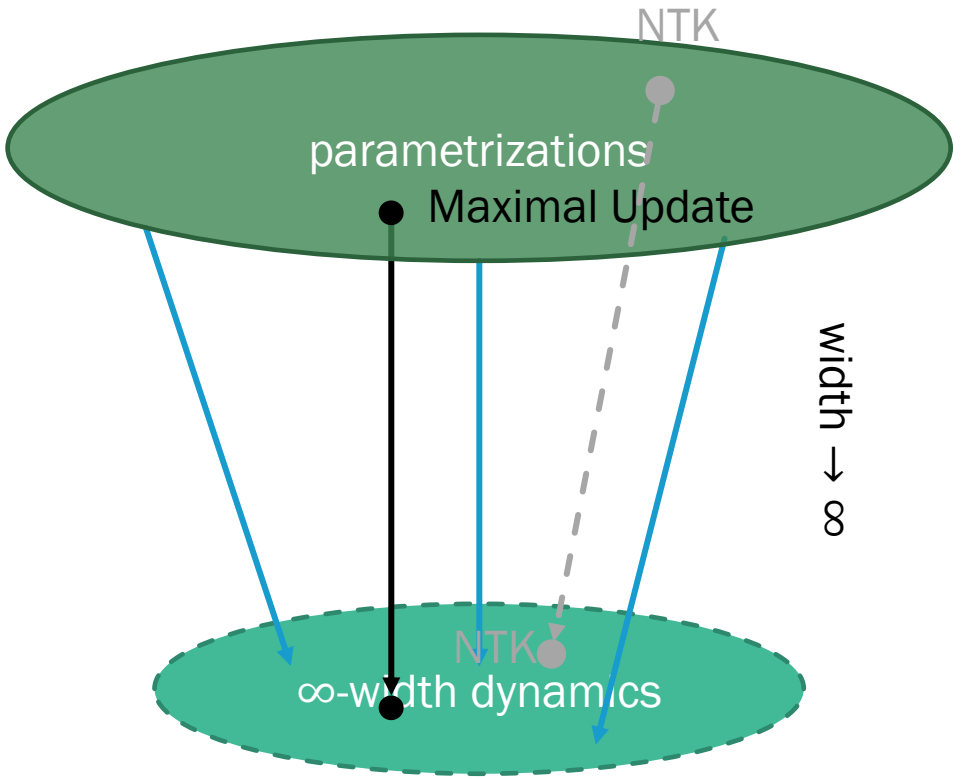
Theoretical Foundation

WHAT IS A PARAMETRIZATION?

	Input weights & all biases	Output weights	Hidden weights
Init. Var.	?	?	?
SGD LR	?	?	?
Adam LR	?	?	?

- Tells you how to scale HP when model size changes
  - "half learning rate when width doubles"
- *Doesn't* tell you how to set specific values of HP at particular model sizes
  - "use learning rate 1e-3 at width 1024"
- For any fixed model size, all parametrizations are *equivalent* up to tunable HP
  - Parametrization gives a way for expressing hyperparameters, just like a basis gives a way to express vectors in a vector space
  - A priori, there is no canonical parametrization, just like a priori there is no canonical basis in a vector space

EACH PARAMETRIZATION CORRESPONDS TO A ∞-WIDTH LIMIT



"The" Feature Learning Limit

OPTIMAL SCALING THESIS

Verify empirically

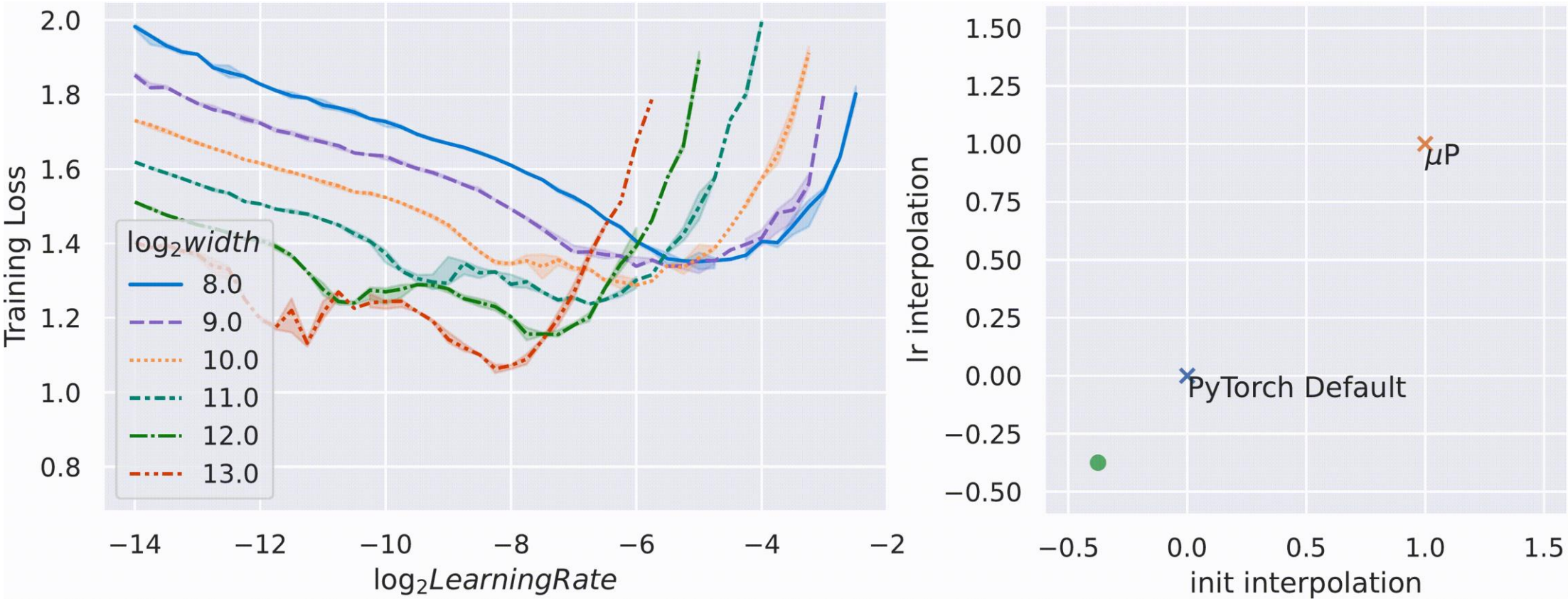
Verify theoretically

HP Transfer
A parametrization preserves HP optimum across model sizes

Maximal Limit
Its limit maximizes feature learning (without blowing up)

This is a thesis for any notion of model size: width, depth, #experts, etc

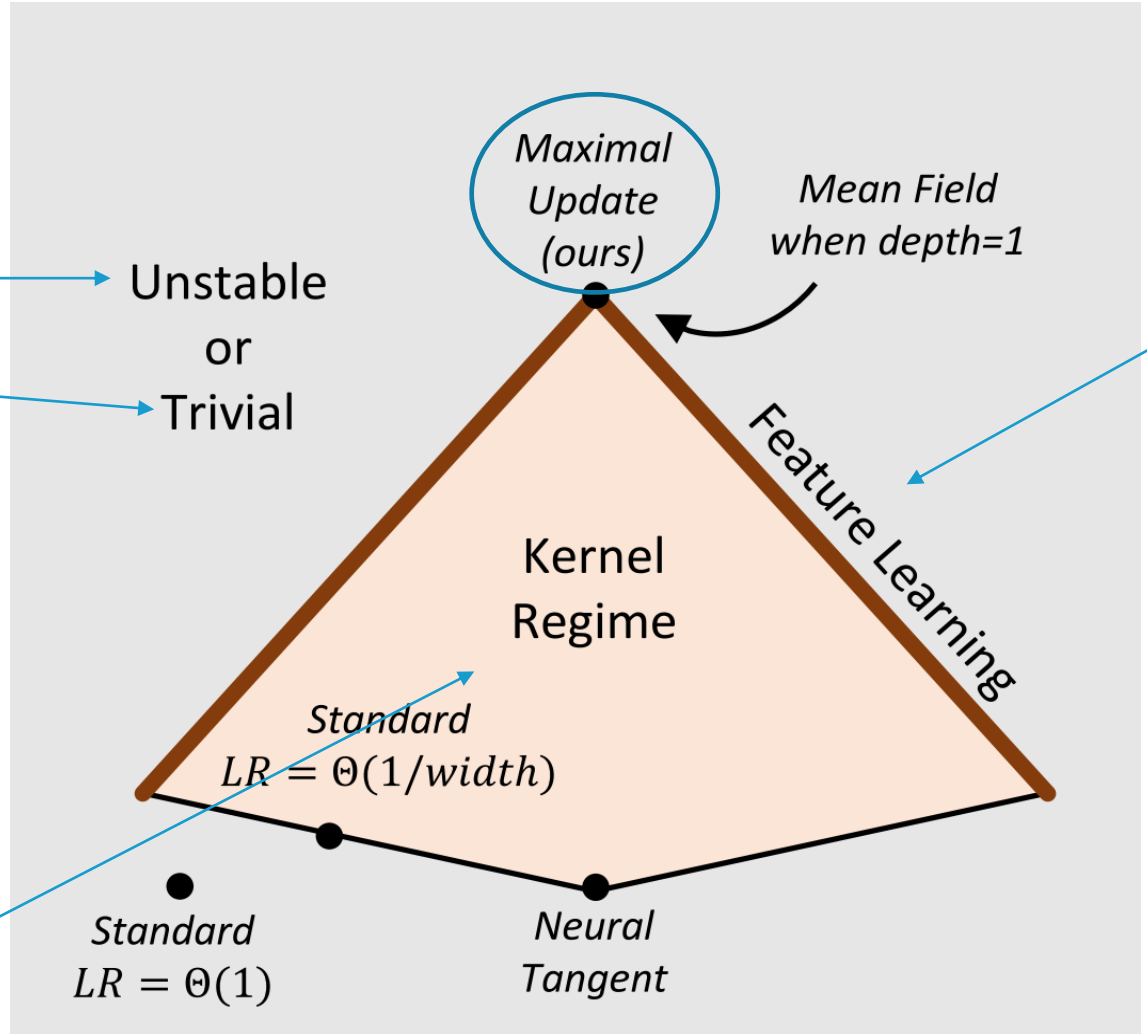
OPTIMAL SCALING THESIS



“PROVING” OPTIMAL SCALING THESIS FOR WIDTH

Dynamical Dichotomy Theorem

A Caricature of
Space of Parametrizations



- Feature learning
- Function evolution cannot be described purely in the function space

- Function evolution described by functional equation for some kernel K

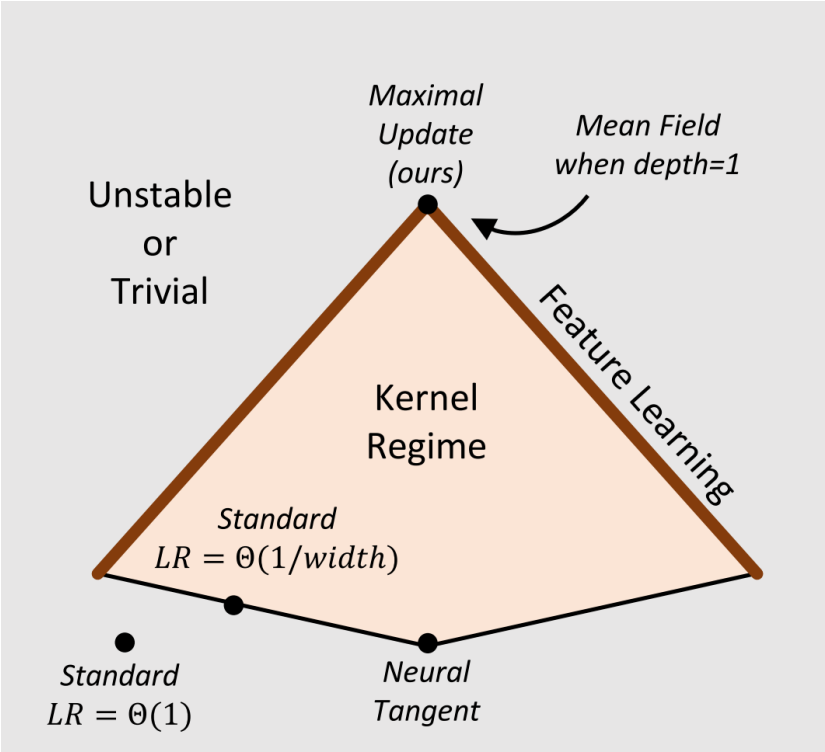
$$f_{t+1} - f_t = -\eta K \mathcal{L}'(f_t)$$

  - No feature learning

"PROOF" OF OPTIMAL SCALING THESIS FOR WIDTH

- Obviously unstable or trivial parametrizations won't work
  - E.g., optimal learning rate will converge to 0 or diverge.
- Kernel regime parametrizations have trivial performance on pretraining tasks
  - E.g., NTK performs trivially on Word2vec; see TP4
- Nonmaximal feature learning limits differ from μP's (maximal) limit only in that some parameters are stuck at init, ignoring the learning rate
  - But in finite networks, for such parameters, learning rate obviously has nontrivial effect --- so there's no way we can expect transfer
- The only parametrization left is μP

A Caricature of
Space of Parametrizations



TENSOR PROGRAMS

WHAT IS A TENSOR PROGRAM?

- A computation composed of 1) matrix multiplication and 2) coordinatewise nonlinearities

Key Insight 1

Church-Turing Thesis for Deep Learning

Any “useful” deep learning computation can be expressed as a Tensor Program.

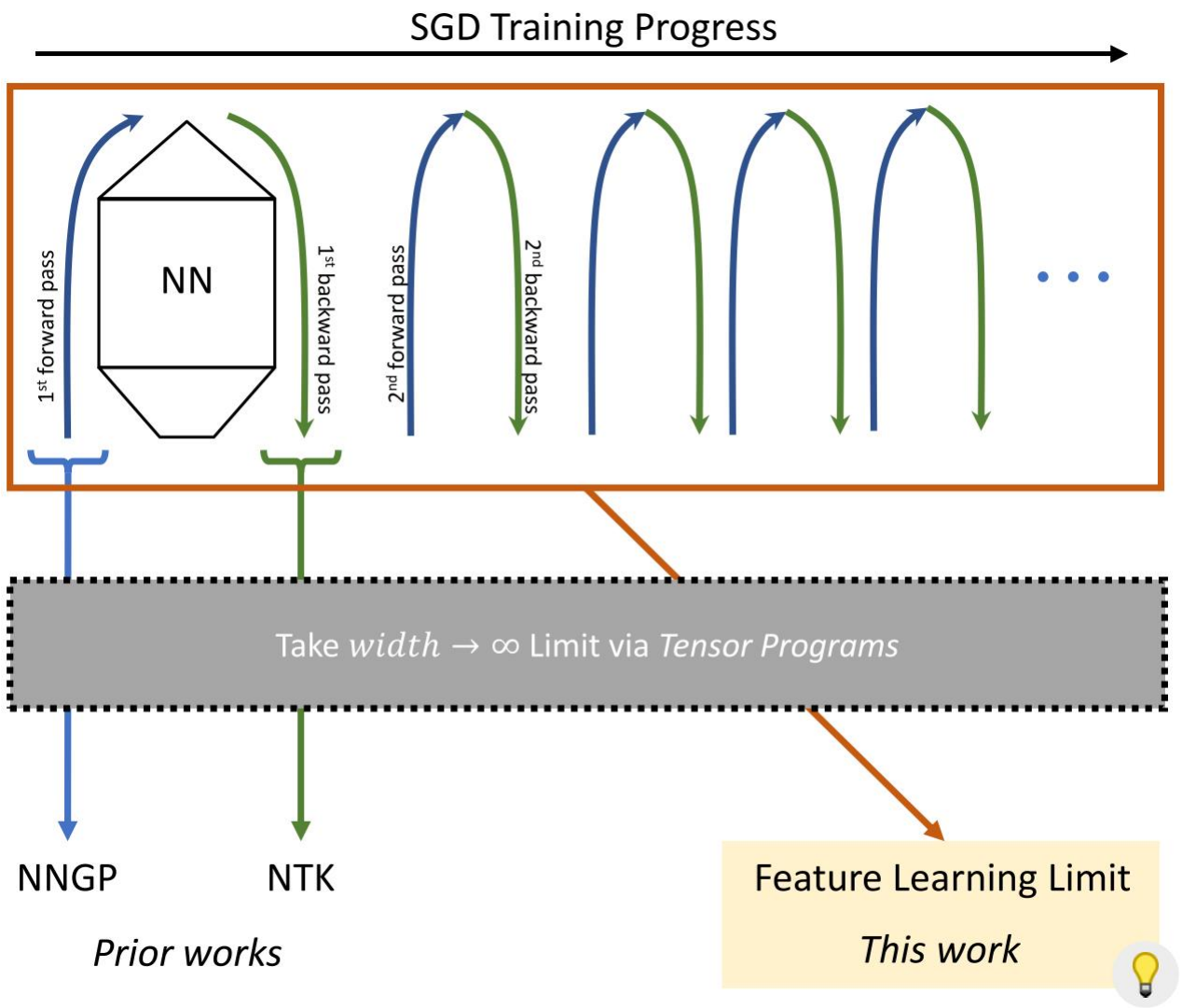
Key Insight 2

∞-*Width Limit for Any Tensor Program*

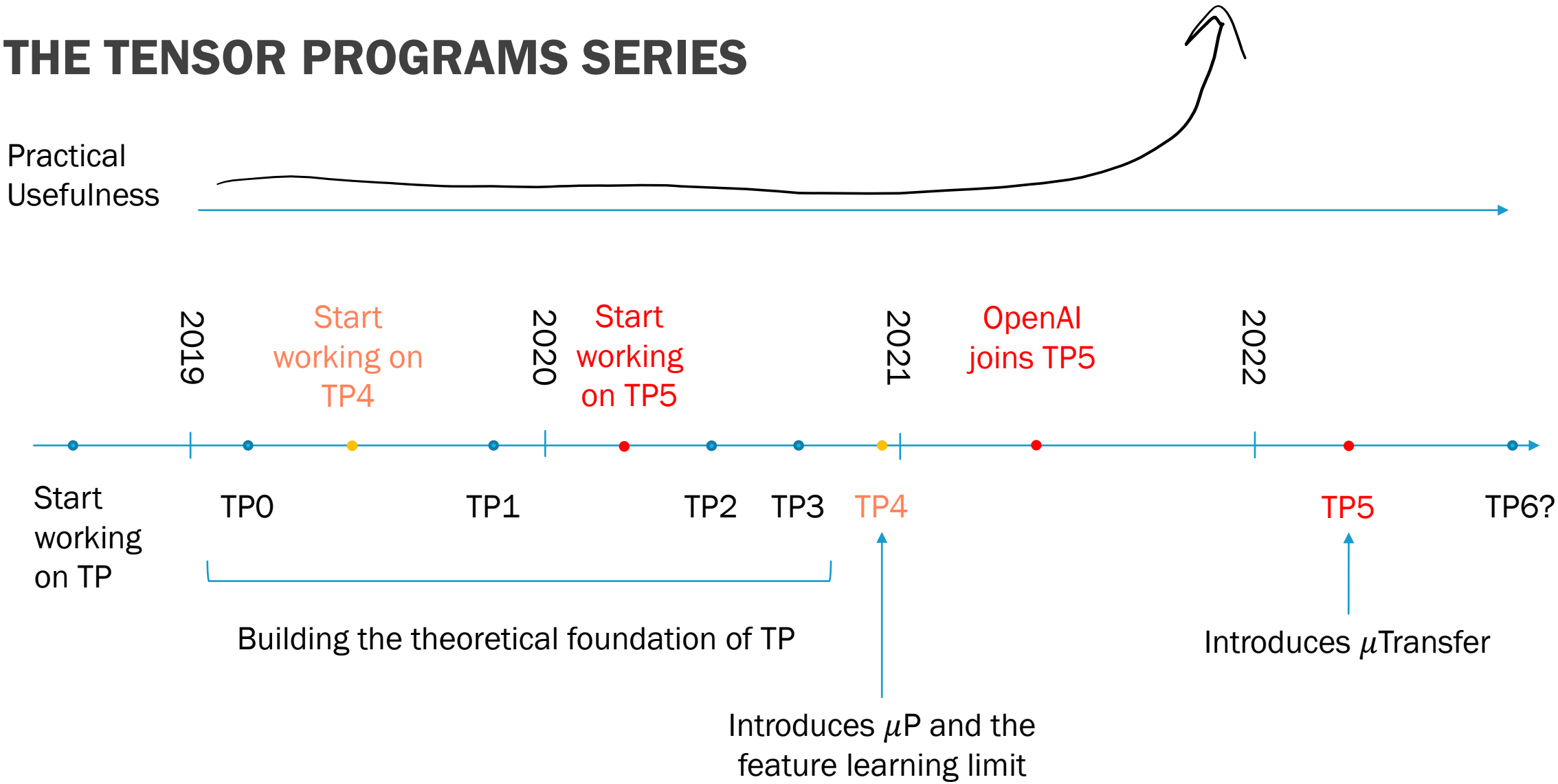
One can prove a *Master Theorem* that reduces the calculation of ∞-width limit to a set of mechanical calculations

Autograd ⟷ gradients

Master Theorem ⟷ ∞-width limit


SGD Training Progress
NN
1st forward pass
1st backward pass
2nd forward pass
2nd backward pass
Take $width \to \infty$ Limit via *Tensor Programs*
NNGP
NTK
Prior works
Feature Learning Limit
This work

THE TENSOR PROGRAMS SERIES

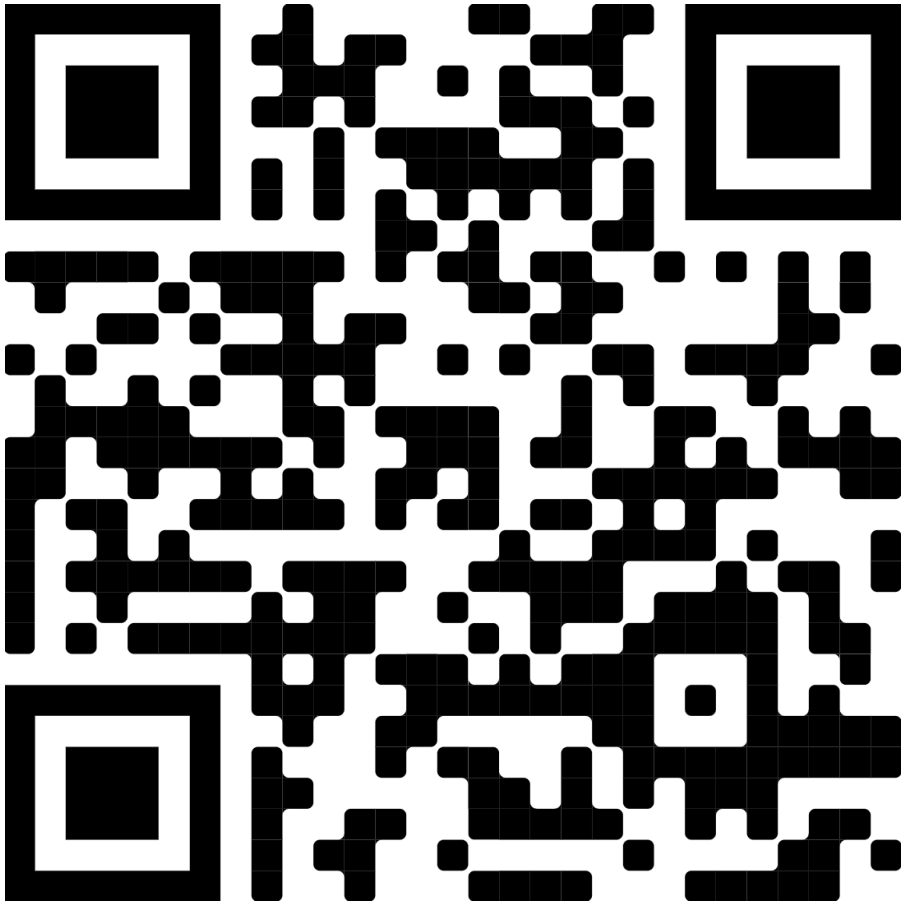


CONCLUSION

LOOKING FORWARD: THE THEORY OF EVERYTHING

- μP solves the transfer problem for width in a principled way. Can we do it for all other compute hyperparameters?
  - Ex: depth, #experts, training time, sequence length
  - Naïve transfer seems to work OK empirically, but as we go to larger scales, likely they will break down
  - Verify the *Optimal Scaling Thesis*
- This is akin to the search for the Theory of Everything in physics
  - The different scale hyperparameters of a neural network are like the different fundamental forces of nature
  - We look for a theory that tells us how to scale every aspect of a model together, just like physicists look for a GUT unifying all fundamental forces of nature

Paper



Pytorch library for μP

```
> pip install mup
```

OR

```
> git clone github.com/microsoft/mup
```