A thorough reproduction and evaluation of μP

Anonymous authors
Paper under double-blind review

Abstract

This paper is an independent empirical reproduction of the claimed benefits of the μP parametrization proposed in Yang & Hu (2020) and Yang et al. (2021). Under the so-called Standard Parametrization (SP), the weights of neural networks are initialized from the Gaussian distribution with variance scaling as the inverse of "fan-in", with the learning rate being the same for every layer. While this guarantees that (pre)activations are $\mathcal{O}(1)$ at initialization with respect to width, it causes their scale to be width-dependent during training. To address this, Yang & Hu (2020) and Yang et al. (2021) proposed the Maximal Update Parametrization (μP), which is also claimed to make the optimal value of various hyperparameters independent of width. However, despite its alleged benefits, μP has not gained much traction among practitioners. Possibly, this could stem from a lack of thorough independent evaluation of μP against SP. We address this by independently reproducing the empirical claims of the original works. At the same time, we substantially increase the scale of the experiments, by training more than 10000 neural networks of sizes from 500 to 0.5B parameters, and empirically investigate μP's effect on outputs, gradient updates, weights, training loss and validation loss. We find that generally μP indeed delivers on its promises, even though this does not always translate to improved generalization.

1 Introduction

1.1 Related works

Deep Learning researchers and practitioners have long understood the importance of initialization and its relation to width. The work LeCun et al. (2002) advocated that weights be sampled from a distribution with mean zero and standard deviation $\frac{1}{\sqrt{\text{fan-in}}}$ (LeCun initialization). Glorot & Bengio (2010) shed further light on why this is helpful, and Sutskever et al. (2013) showed that initialization schemes like this can synergize with momentum methods.

The paper Yang & Hu (2020) recognized that LeCun initialization ensures that (pre)activations are $\mathcal{O}(1)$ at the beginning of training. The authors noted that this property is lost during training, which can cause wide networks to diverge. Starting from the desideratum that (pre)activations are $\mathcal{O}(1)$ throughout training, and using the theory developed in the Tensor Programs (TP) series of papers (Yang, 2019a;b; 2020a; Yang & Littwin, 2021; Yang, 2020b; Yang & Hu, 2020; Littwin & Yang, 2022; Yang et al., 2023a; 2021; 2023b; Yang & Hu, 2020), they arrive at the width scaling scheme μP. For many hyperparameters, μP is also claimed to stabilize optimal values as width varies, a property that is exploited in Yang et al. (2021) for hyperparameter optimization. In this paradigm, called μTransfer, optimal hyperparameters are discovered cheaply for a small, proxy network, and then zero-shot transferred to a big, target network.

Since its proposal, μP has been used in a limited number of published works. For the case of Large Language Models (LLMs), it has been used by Dey et al. (2023a) (Cerebras-GPT), Li et al. (2023) (FLM-101B), Dey et al. (2023b) (BTLM-3B-8K), Liu et al. (2023) (CrystalCoder), Hu et al. (2024) (MiniCPM) and Li et al. (2024) (Tele-FLM). Intriguingly Achiam et al. (2023) (GPT-4) includes Yang & Littwin (2021) in the references without explicitly citing it, leaving it unclear if they use it or not. Outside of the LLM world, μP was used in Cabannes et al. (2023) to ensure that a fixed learning rate was reasonable for every width they tested, and in Beaini et al. (2023), which included μP in their GNN library Graphium targeted at Molecular Learning.

1.2 Objectives

The above works using μP nearly always assume its benefits, taking at face value that μP is preferable over SP without ablating with respect to the parametrization. Besides the original papers (Yang & Hu, 2020; Yang et al., 2021), the only work that investigates the claimed advantages of μP over SP is Lingle (2024). It studies whether μP indeed stabilizes the optimal learning rate for many architectural variations of a Transformer, and finds that it does for most but not all of these variations.

In this paper, we will thoroughly investigate the alleged benefits of μP and compare it head-to-head with SP. We expand the scale of the existing μP versus SP comparisons (Yang & Hu, 2020; Yang et al., 2021; Lingle, 2024), by including additional architectures and domains, scaling to narrower and wider networks, performing a denser hyperparameter sweep, training for more random seeds and training for longer. In total, we train 10752 networks, ranging from 500 to 0.5B parameters. Our ultimate goal is to understand whether and to what extent the promises of μP hold in practice, and if and when it should be preferred over SP.

Our work is fundamentally an independent reproduction of Yang & Hu (2020) and Yang et al. (2021). Hence, we made every effort that our results are reproducible themselves. The complete repository of the training code[1] is already available online.

1.3 Findings

Our findings can be summarized as follows:

1. Inspecting the norm of coordinate-wise outputs reveals that they indeed are $\mathcal{O}(1)$ under μP, while heavily depending on width under SP.
2. In μP, and unlike SP, the best (with respect to the training loss) learning rate indeed stays *approximately* constant as width increases. Thus, μTransfer, in contrast to "naive" hyperparameter tuning with SP, indeed enables zero-shot hyperparameter transfer, from narrow (and thus cheaply trainable) networks to wider ones.
3. Under μP, wider networks *in general* outperform (in training loss) narrower networks. Under SP this trend is much less visible, although sometimes present.
4. Points 2 and 3 do not always translate to better generalization. That is, the optimal μP network often has worse validation loss than the optimal SP network.
5. With SP, we observed some wide networks diverging. Specifically, the wider the network, the more likely it was to diverge. In contrast, none of the networks diverged with μP.
6. The benefits of μP seem to be stronger for transformers.

In summary, we found that μP *mostly* performs as expected.

2 μP summary

We start with a high-level summary of the Tensor Programs framework (Yang, 2019a).

In this framework, the initial weights and learning rates of a neural network are scaled in terms a parameter matrix's "fan-in" and "fan-out". Their precise meaning for different types of layers are as follows:

1. The parameter matrix of biases of a linear layer has fan-in = 1 and fan-out equal to the activation dimension.

[1] ANONYMIZED

Table 1: Standard deviation and learning rate scaling in μP

	fan-out $\to \infty$	fan-in, fan-out $\to \infty$	fan-in $\to \infty$
s	$1/\sqrt{\text{fan-in}}$	$1/\sqrt{\text{fan-in}}$	$1/\text{fan-in}$
γ (SGD)	fan-out	1	$1/\text{fan-in}$
γ (Adam)	1	$1/\text{fan-in}$	$1/\text{fan-in}$

2. Convolutional filters are a kernel_width $\times$ kernel_height-sized collection of parameter matrices, where every such matrix has fan-in = input_channels and fan-out = output_channels.

3. Biases and weights of layer normalization layers are treated the same as biases of linear layers.

4. The class embedding of Vision Transformers (ViTs) has fan-in = 1 and fan-out = d, where d is the model dimension.

5. The embedding operation of a transformers is viewed as a matrix multiplication between the embedding table and a one-hot vector representing a token of the vocabulary. Therefore, the embedding table has fan-in = vocabulary_size and fan-out = d.

With these conventions, assume $\theta \in \mathbb{R}^{\text{fan-in} \times \text{fan-out}}$ is a parameter matrix of a neural network.

Under SP, the initialization and update rules are:

$$\theta_0 \sim \begin{cases} \mathcal{N}(\mu, c^2) & \text{if fan-in} = 1, \\ \mathcal{N}(0, c^2 \cdot \frac{1}{\text{fan-in}}) & \text{if fan-in} > 1, \end{cases} \tag{1}$$

$$\theta_{t+1} \leftarrow \theta_t - k \cdot f(\nabla \theta_t), \tag{2}$$

for a function f, where μ, c and k are hyperparameters that do not scale with width. Note that $c = 0$ is possible (e.g. biases are often initialized to zero). Different choices of f lead to different optimizers (e.g. for $f = \text{id}$ we recover SGD, while another choice leads to Adam).

Under the μP, the initialization and update rules are instead:

$$\theta_0 \sim \begin{cases} \mathcal{N}(\mu, c^2) & \text{if fan-in} = 1, \\ \mathcal{N}(0, c^2 \cdot s^2) & \text{if fan-in} > 1, \end{cases} \tag{3}$$

$$\theta_{t+1} \leftarrow \theta_t - k \cdot \gamma \cdot f(\nabla \theta_t), \tag{4}$$

where s and γ are scaled with width as specified in Table 1. In addition, the scale in a self-attention layer of dimension d should is changed from $\frac{1}{\sqrt{d}}$ to $\frac{1}{d}$.

The constants μ, c and k do not have to match between SP and μP. Moreover, they can be chosen arbitrarily for every parameter matrix of the network. This allows us to make SP and μP *exactly* equivalent for a base width. We can do so by inserting width-independent constants in front of μ, c and k in μP. The constants to be inserted are obtained from equating the initializations (equation 1 and equation 3) and the update rules (equation 2 and equation 4).

3 Experimental setup

We experimented with four architectures, across three tasks. Specifically, we tested a 3-layer MLP on the California Housing dataset, a VGG11 CNN and a ViT on CIFAR-10, and a Transformer on Tiny Shakespeare.

For every architecture we chose a base width, and then trained networks of widths $\zeta \times$ base_width while varying ζ. We ran comprehensive experiments for each architecture and dataset combination. For every combination, we picked multipliers to make SP and μP exactly equivalent for the base width $\zeta = 1$ (as described in the previous section). We swept the learning rate hyperparameter k, training 16 networks for every value.

In the MLP setting, we followed Yang et al. (2021, Figure 5) in plotting the norm of coordinate-wise outputs to test μP's stabilizing effect on them. We also did the same for weights and gradient updates.

In all settings, we compared performance for different hyperparameter values at varying width, producing curves like those of Yang et al. (2021, Figure 1). Specifically, we collected the minimum training and validation losses, and plotted their mean, along with one standard deviation error bars for both parametrizations.

For all the experiments, we set the initialization scale c to $1/10$ and used the Adam optimizer (Kingma & Ba, 2017) with PyTorch's defaults. Additionally, we trained without weight-decay or data augmentation.

In total, we trained 10752 neural networks, spanning from 500 to 0.5B parameters, which needed 2000 hours when using an NVIDIA A100.

3.1 MLP on California Housing

The California Housing dataset (Pace & Barry, 1997) is a tabular regression dataset with the goal of predicting the median house value for a geographical block in California from eight real-valued features. It consists of 20640 samples, out of which we held out 2000 for validation and 2000 for testing.

We used a MLP with two hidden layers, and gave them a base width of 16 (this is the only width that scales for this architecture). We trained networks corresponding to width multipliers from $\zeta = 1$ (width = 16, parameters = 433) to $\zeta = 512$ (width = 8192, parameters = 67M). For each width we trained with 16 different learning rate multipliers k, geometrically spaced between 10^{-5} and one. Each training run consisted of 50000 mini-batches of size 16.

Overall, we trained 5120 MLPs, which took around 200 hours on an NVIDIA A100.

3.2 VGG11 on CIFAR-10

The CIFAR-10 dataset (Krizhevsky et al., 2009) is an image classification dataset where one tries to classify an image in one of ten classes. There are 60000 images, of size $3 \times 32 \times 32$. We held out 10000 images for validation and 10000 for testing.

We used the VGG11 architecture (Simonyan & Zisserman, 2014) with four convolutional stages. The stages had base width[2] 4, 8, 16 and 32 respectively. The classifier head had base width 20, and 0.5 dropout probability. We tested networks from $\zeta = 1$ (max_channels = 32, parameters = 21K) to $\zeta = 128$ (max_channels = 4096, parameters = 336M). We tried eight geometrically spaced values for the learning rate multiplier k, between $6 \cdot 10^{-5}$ and 0.01. Each training run consisted of 50000 mini-batches, of size 32.

In aggregate, we trained 2048 CNNs, in about 500 GPU hours.

3.3 ViT on CIFAR-10

We used the ViT architecture (Dosovitskiy et al., 2020) with a patch size of four and six blocks of base width 32, eight heads, expansion factor of one and 0.1 dropout probability. For positional embeddings we used sinusoidal positional encodings. We tested networks from $\zeta = 1$ (width = 32, parameters = 34K) to $\zeta = 128$ (width = 4096, parameters = 504M). The remaining training details follow Section 3.2.

Collectively, we trained 2048 ViTs, which took 1000 GPU hours.

3.4 Transformer on Tiny Shakespeare

The Tiny Shakespeare (Karpathy, 2015) dataset is a subset of Shakespeare's works in a single 40000 lines file. Language models trained from scratch on this dataset can produce samples that look very close to the

[2] The width of a convolutional layer is simply the number of its output channels.

original. We tokenized the dataset with the GPT-2 (Radford et al., 2019) tokenizer, leading to 300K tokens. We held out 25K tokens for validation and 25K tokens for testing.

We used the transformer architecture Vaswani et al. (2017) with a context of 128 tokens and six blocks of base width 32, eight heads, expansion factor of four and no dropout. For positional embeddings we used sinusoidal positional encodings. We tested networks from $\zeta = 1$ (width = 32, parameters = 3.3M) to $\zeta = 32$ (width = 1024, parameters = 180M). We tried eight geometrically spaced values for the learning rate multiplier k, between $6 \cdot 10^{-4}$ and 0.1. Each training run consisted of 20000 mini-batches of size 32.

Overall, we trained 1536 transformers, in approximately 300 hours.

4 Main results

4.1 MLP on California Housing

4.1.1 Scale of activations

As our first experiment, we measured the average coordinate-wise norm of the output of our MLP architecture, described in Section 3.1. We did this for width multipliers from $\zeta = 1$ to $\zeta = 512$ and for twelve batches. We then compared SP with μP to see the impact of parametrization. According to theory, outputs should be width-dependent under SP, and width-independent under μP. The results are presented in Figure 1. For SP, we can see that the scale of the outputs rapidly increases as we increase the width. On the contrary, for μP, the norm is stable with respect to the width. The results are as expected, and mirror Yang et al. (2021, Figure 5).

We did the same for the gradient updates and the weight norms of the last hidden layer of our MLP. Results are again presented in Figure 1. For the average coordinate-wise norm of the gradient updates, both under SP and μP, we notice that there is an exponential decay with width. The curves are more stable under μP, with small spikes appearing for nearly all batches under SP. Lastly, in terms of the average coordinate-wise norm of the weight values, SP and μP behave similarly.

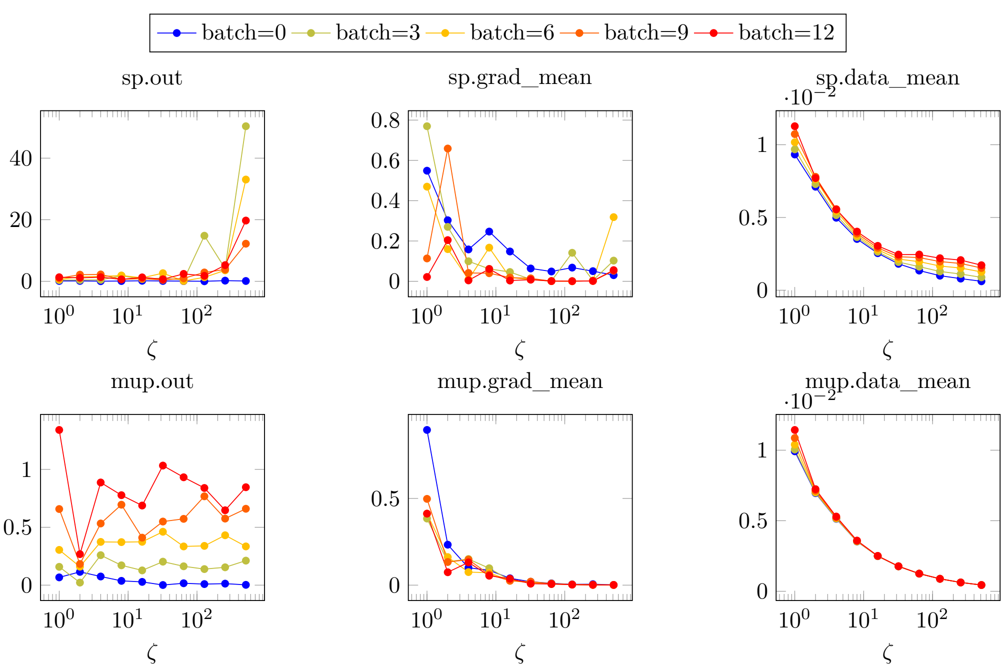

Figure 1: Scale of outputs (left), gradient updates (middle) and weights (right) as function of the width multiplier ζ

4.1.2 Train and validation loss

The results for the stability of the hyperparameters with changing width are shown in Figure 2. We observe that the training loss curves for both SP and μP are quite noisy, with the error bars for different widths overlapping. This indicates that in some cases the benefits of μP are detectable only when averaging over many training runs.

Under SP, the optimal learning rate multiplier k with respect to the training loss shifts around an order of magnitude to the left as the width increases. On the other hand, it stays approximately constant under μP. Moreover, under μP, the curves are somewhat flatter, which means that the networks are less sensitive to the exact value of k.

For SP, wider networks do not consistently outperform narrower ones in terms of training loss, except for a small range of low values of k, and the difference is slight. Meanwhile, this trend is much stronger for μP, and observed for a wider range of k. The validation loss curves show similar behavior, but are less noisy.

Comparing best performing networks with respect to the training loss, we see that the best SP network has $\zeta = 128$, $k = 10^{-4}$ and min_training_loss $= 6.52 \cdot 10^{-2}$, while the best μP network has $\zeta = 512$, $k = 3 \cdot 10^{-4}$ and min_training_loss $= 6.78 \cdot 10^{-2}$. Thus, for SP the third widest network performs the best, while for μP the widest network does. With respect to the validation loss, the best networks have $\zeta = 8$, $k = 6 \cdot 10^{-4}$ and min_val_loss $= 0.48$ for SP and $\zeta = 512$, $k = 0.1$ and min_val_loss $= 0.47$ for μP. Hence, SP has a better best performing network in terms of the training loss in comparison to μP, but a worse one in terms of the validation loss.

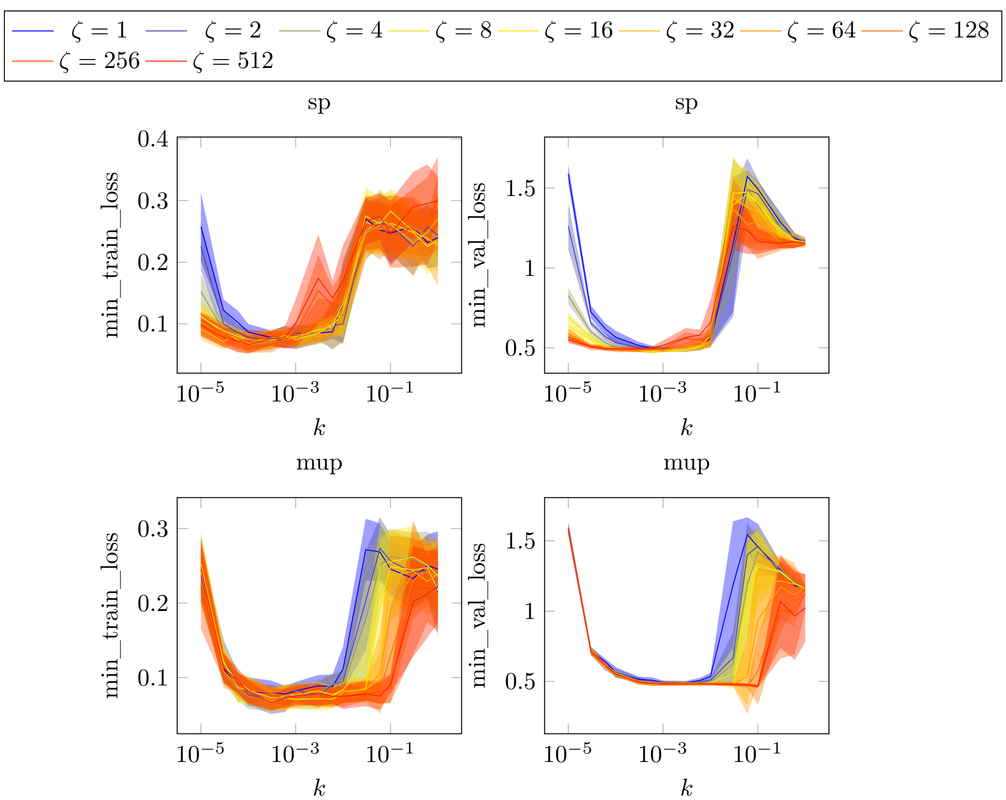

Figure 2: MLP on California Housing

4.2 VGG11 on CIFAR-10

The results for the VGG11 architecture on CIFAR-10 are shown in Figure 3.

As in Figure 2, the training loss curves are quite noisy. Unlike Figure 2, here the noise levels of validation and training loss curves are similar.

Under SP, the best learning rate multiplier k with respect to the training loss shifts around half an order of magnitude to the left as the width increases. On the other hand, it stays roughly constant under μP. For SP, wider networks consistently outperform narrower ones in terms of training loss only for $k \leq 10^{-4}$. Meanwhile, for μP this trend is observed for every k. As for the validation loss curves, they are very similar to the ones for the training loss.

Comparing best performing networks with respect to the training loss, we see that the optimal network for SP has $\zeta = 128$, $k = 6 \cdot 10^{-5}$ and min_training_loss $= 2.61 \cdot 10^{-5}$, while μP has $\zeta = 64$, $k = 3 \cdot 10^{-3}$ and min_training_loss $= 1.27 \cdot 10^{-4}$. Hence, it is actually not the widest network that performs the best for μP. With respect to the validation loss, the best performing network has $\zeta = 128$, $k = 6 \cdot 10^{-1}$ and min_val_loss $= 0.74$ for SP and $\zeta = 128$, $k = 10^{-3}$ and min_val_loss $= 0.76$ for μP. In summary, the best performing SP networks outperform the best performing μP networks, in terms of both the training loss and the validation loss.

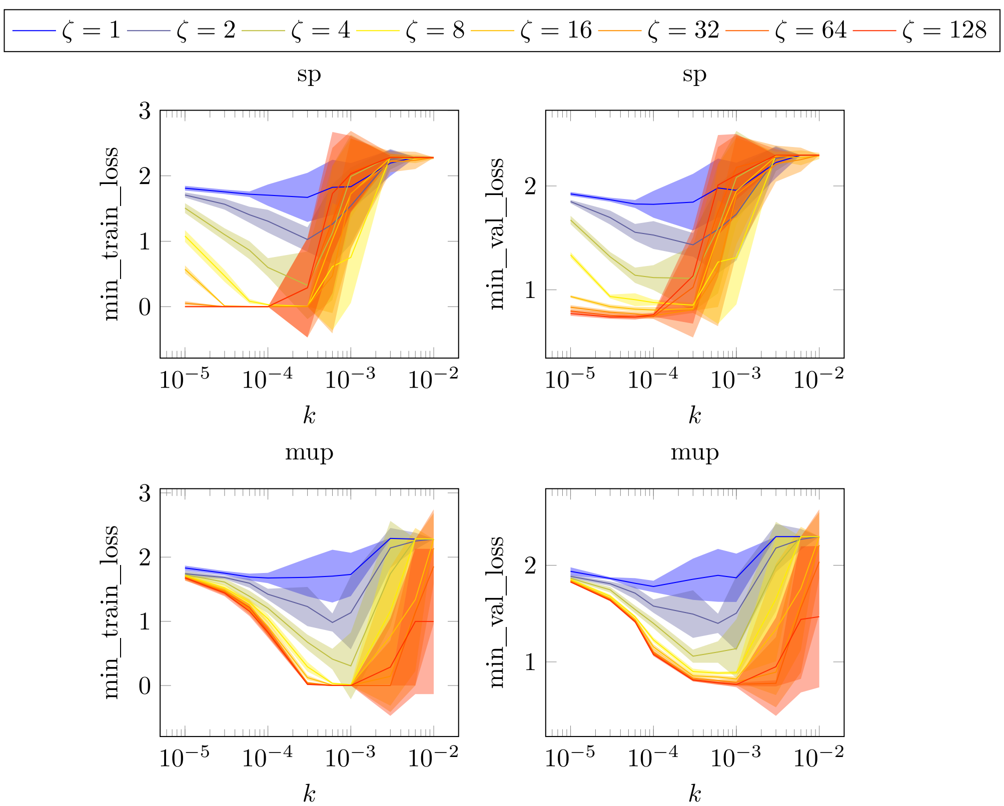

Figure 3: VGG11 on CIFAR-10

4.3 ViT on CIFAR-10

The results for the ViT architecture on CIFAR-10 are shown in Figure 4.

The error bars are much tighter than in Figure 2 and Figure 3, both for the training and for the validation loss curves.

Under SP, we see that the best learning rate multiplier k with respect to the training loss shifts around two orders of magnitude to the left as the width increases. On the other hand, it stays almost constant under μP.

For SP, for some k, we can see wider networks outperforming narrower ones in terms of training loss. However, this can only be observed for a small range of learning rates close to the smallest we tried, and the difference is slight. Meanwhile, for μP the loss is roughly monotonically decreasing for a larger range of k, centered on the (approximately width-independent) optimum.

Comparing best performing networks with respect to the training loss, the best SP network was the second widest, with $\zeta = 64$, $k = 3 \cdot 10^{-5}$ and min_training_loss $= 5.62 \cdot 10^{-3}$, while for μP it was obtained for the third widest, with $\zeta = 32$, $k = 3 \cdot 10^{-3}$ and min_training_loss $= 0.01$. Hence, though the μP networks exhibit better stability than the SP networks in terms of best learning rate, as well as higher monotonicity of the train loss relative to width, the best SP network in fact outperforms the best μP in terms of training loss by half an order of magnitude.

The shape of the validation loss curves is qualitatively similar to the training loss curves, with the notable exception of the validation loss curve of the widest μP network. The pronounced spike is reminiscent of double descent, though we have not investigated this further.

Furthermore, for both SP and μP wider networks perform worse in terms of validation loss. With respect to the validation loss, the best performing SP network had $\zeta = 4$, $k = 10^{-4}$ and min_val_loss $= 1.07$, and the best performing μP network had $\zeta = 4$, $k = 3 \cdot 10^{-3}$ and min_val_loss $= 1.09$. Hence, also in terms of validation loss, the best SP network outperforms the best μP network, though the difference is small.

Another interesting observation is that some SP networks diverged during training. Specifically, one network diverged for $\zeta = 32$, two networks diverged for $\zeta = 64$ and five networks diverged for $\zeta = 128$. By contrast, no μP networks diverged. The pattern suggests that SP networks become increasingly unstable as we increase the width, while μP networks are more stable, consistently with the theory behind μP. Eight networks is a tiny number compared to the 1280 total networks we trained, so this could go unnoticed had the scale of our experiments been smaller. However, this could prove crucial for the training of extremely big networks (e.g. LLMs).

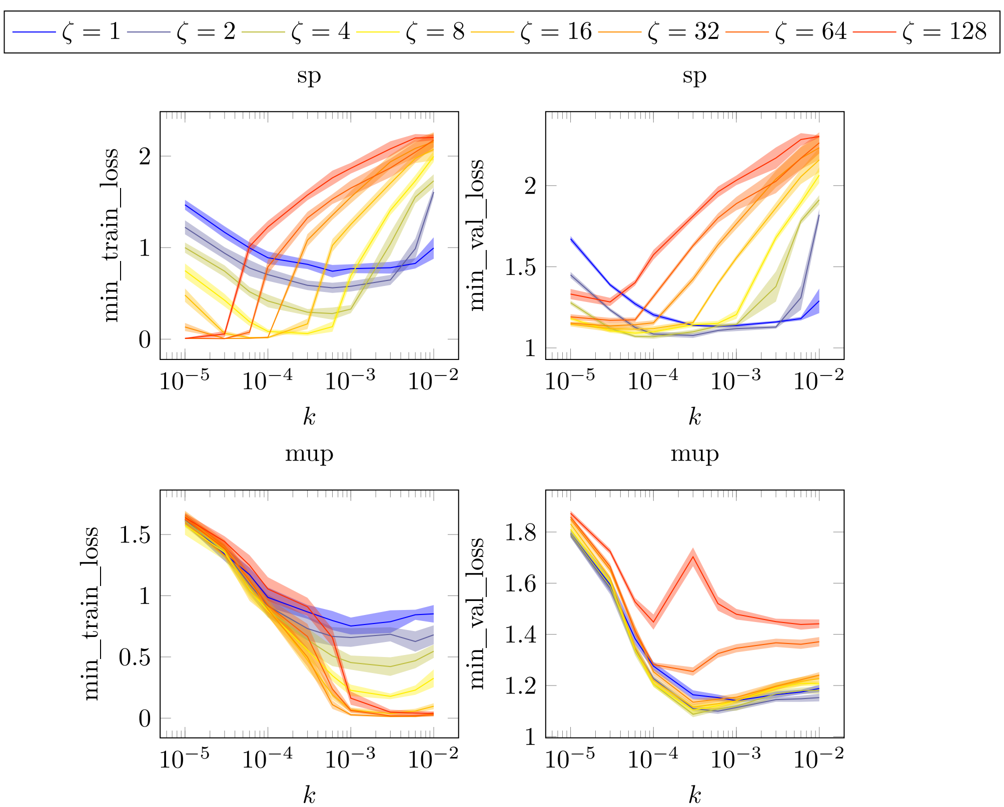

Figure 4: ViT on CIFAR-10

4.4 Transformer on Tiny Shakespeare

The results for the transformer on the Tiny Shakespeare dataset are shown in Figure 5.

The training curves are very similar to those reported for a transformer language model in Yang et al. (2021, Figure 1). Moreover, we notice that the training and validation curves have significantly more noise for SP.

Under SP, we see that the best learning rate multiplier k with respect to the training loss shifts around two orders of magnitude to the left as the width increases. On the other hand, it stays almost constant under μP. Furthermore, like in Figure 2, under μP the curves are flatter, meaning that the networks are less sensitive to the value of k. For a small range of k wider SP networks outperform narrower ones in terms of training loss, while for μP this behavior is much more consistent, for almost every k.

Quantitatively, the best network for SP was obtained for $\zeta = 32$, $k = 3 \cdot 10^{-4}$ with min_training_loss = 0.17, while for μP it was obtained for $\zeta = 32$, $k = 6 \cdot 10^{-3}$ with min_training_loss = 0.18. Hence, in terms of training loss, the best SP network somewhat outperformed the best μP network.

The validation loss curves are qualitatively similar to their training loss counterparts. For SP, the best network in terms of validation loss was obtained for $\zeta = 16$, 10^{-4} with min_val_loss = 4.57, while for μP it was obtained for $\zeta = 32$, $k = 6 \cdot 10^{-3}$ with min_val_loss = 4.72. Hence, also in terms of validation loss, the best SP network outperformed the best μP network.

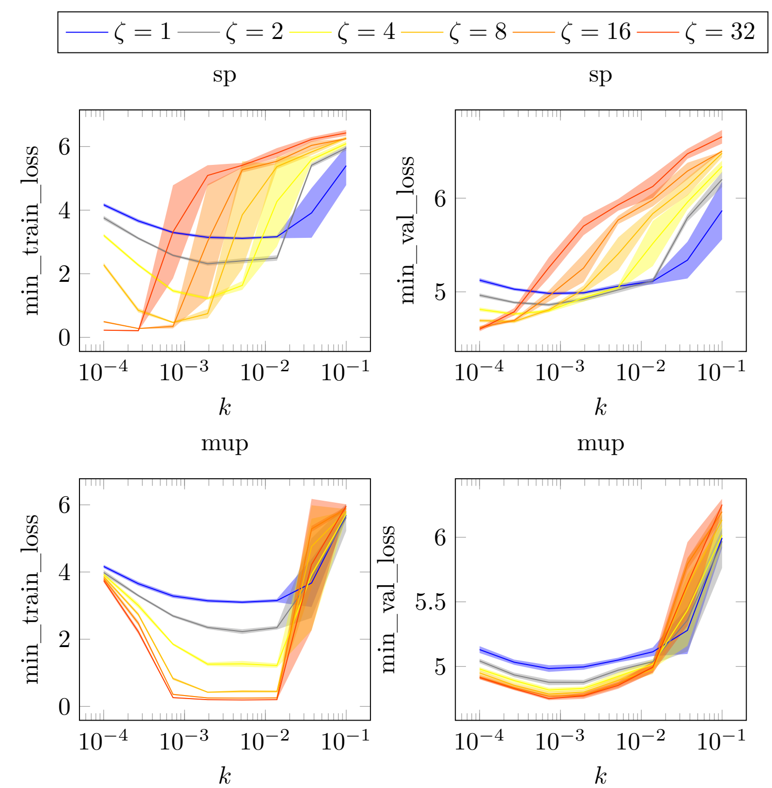

Figure 5: Transformer on Tiny Shakespeare

5 Conclusion

This paper is a head-to-head comparison between SP and μP. We independently reproduced the empirical claims of Yang & Hu (2020) and Yang et al. (2021), while at the same time significantly increasing the scale of the experiments. We confirm that μP indeed has a number of benefits over SP, even though one might not observe all of them in every setup. In general, μP stabilizes the optimal learning rate as a function

Table 2: Summary of our results

Architecture	Parametrization	min_train_loss	min_val_loss	Networks diverged
MLP	SP	$\mathbf{6.52 \cdot 10^{-2}}$	0.48	**0**
	μP	$6.78 \cdot 10^{-2}$	**0.47**	**0**
VGG	SP	$\mathbf{2.61 \cdot 10^{-5}}$	**0.74**	**0**
	μP	$1.27 \cdot 10^{-4}$	0.76	**0**
ViT	SP	$\mathbf{5.62 \cdot 10^{-3}}$	**1.07**	8
	μP	0.01	1.09	**0**
Transformer	SP	**0.17**	**4.57**	74
	μP	0.18	4.72	**0**

of width and makes wider networks outperform narrow ones. Furthermore, it alleviates divergence issues. However, in terms of both train and validation error, the best μP network is quite often worse than the best SP network.

Our results do confirm that transferring hyperparameters from a narrow network to a wider ones works under μP, but not under SP. In practice, for SP, it is more common to optimize hyperparameters by training the same sized network for only a few iterations while varying the hyperparameters. It would be interesting to compare that protocol to μTransfer for the same compute budget.

Since μP is theoretically well-founded and empirically has a consistent stabilizing effect, it merits further investigation. In particular, future research should investigate under what circumstances μP is better than SP in terms of absolute performance, and whether μP can be made to consistently outperform SP.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, Jama Hussein Mohamud, et al. Towards foundational models for molecular learning on large-scale multi-task datasets. *arXiv preprint arXiv:2310.04292*, 2023.

Vivien Cabannes, Elvis Dohmatob, and Alberto Bietti. Associative memories with heavy-tailed data. In *NeurIPS 2023 Workshop Heavy Tails in Machine Learning*, 2023.

Nolan Dey, Gurpreet Gosal, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, Joel Hestness, et al. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv:2304.03208*, 2023a.

Nolan Dey, Daria Soboleva, Faisal Al-Khateeb, Bowen Yang, Ribhu Pathria, Hemant Khachane, Shaheer Muhammad, Robert Myers, Jacob Robert Steeves, Natalia Vassilieva, et al. Btlm-3b-8k: 7b parameter performance in a 3b parameter model. *arXiv preprint arXiv:2309.11568*, 2023b.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015. URL `https://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Accessed: 2024-06-26.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Xuying Meng, Siqi Fan, Peng Han, Jing Li, Li Du, Bowen Qin, et al. Flm-101b: An open llm and how to train it with $100 k budget. *arXiv preprint arXiv:2309.03852*, 2023.

Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Chao Wang, Xinzhang Liu, Zihan Wang, Yu Zhao, Xin Wang, Yuyao Huang, et al. Tele-flm technical report. *arXiv preprint arXiv:2404.16645*, 2024.

Lucas Lingle. A large-scale exploration of μ-transfer. *arXiv preprint arXiv:2404.05728*, 2024.

Etai Littwin and Greg Yang. Adaptive optimization in the ∞-width limit. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. Llm360: Towards fully transparent open-source llms. *arXiv preprint arXiv:2312.06550*, 2023.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3): 291–297, 1997.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019b.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020a.

Greg Yang. Tensor programs iii: Neural matrix laws. *arXiv preprint arXiv:2009.10685*, 2020b.

Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pp. 11762–11772. PMLR, 2021.

Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023a.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Feature learning in infinite depth neural networks. In *The Twelfth International Conference on Learning Representations*, 2023b.