APPENDIX

Anonymous authors
Paper under double-blind review

1 MORE ANALYSIS

Error Comparison of Scaling Laws. Based on Table 8 of the Cerebras-GPT and the scaling law's power function $L = ax^b + c$, we conducted an error comparison analysis of the predicted losses for Cerebras-GPT, Pythia, and nanoLM. As shown in table 1, Cerebras-GPT fitted the 13B model's loss using $111M - 6.7B$ models, Pythia for $12B$ with $7M - 6.9B$, and nanoLM for $52B$ from $77M - 3.4B$, with their respective errors being 0.025, 0.019, and 0.022. When fitting losses for models down to 10B, the errors are 0.034, 0.049, and 0.018, respectively. Additionally, we calculated the covariances of the fitted coefficients $\{a, b, c\}$, finding that nanoLM's covariances are significantly lower than those of Cerebras-GPT and Pythia. These experimental results demonstrate the (a) μP infinite neural network is theoretically correct and (b) scaling laws are empirically reliable on any scale. (c) The loss prediction of nanoLM is more stable and reliable. Loss prediction validation is costly, and we have conducted as many experiments as possible within our computational power limits (nanoLM has reached 52B for this purpose, while Pythia and Cerebras-GPT have only reached 13B). Further experimentation would be prohibitively expensive in terms of computational costs.

Table 1: Comparison of Fitted Results: The "Coeffs & Cov" denote the coefficients {a, b, c} and the covariance of the power-law function $y = aC^b + C$.

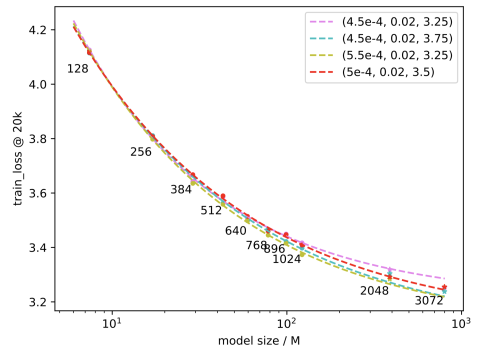
Model	**Size & Loss**									**Coeffs & Cov**		
Cerebras-GPT	0.111	0.256	0.59	1.30	2.700	6.700	13.00	-	-	6.76e1	-8.45e-2	7.25e1
	2.608	2.349	2.181	1.997	1.834	1.704(0.034)	1.572(0.025)	-	-	4.84e1	2.10e-2	3.44e-1
Pythia	0.070	0.160	0.410	1.000	1.400	2.800	6.900	12.00	-	9.67e6	-0.34	1.42
	2.549	2.204	1.989	1.858	1.889	1.724	1.644(0.049)	1.601(0.019)	-	3.89e7	8.89e-2	1.50e-1
nanoLM	0.077	0.153	0.254	0.381	0.532	0.709	0.911	3.432	5.24e1	0.25	-0.47	2.82
	3.656	3.389	3.298	3.215	3.198	3.087	3.080	2.958(0.018)	2.883(0.022)	7.33e-2	8.50e-2	7.66e-2

Embedding counts as model size. We demonstrate that our scaling law fits worse if embedding weights are not counted in the model sizes, in contrast to **?**.This is potentially because μP concluded that the learning rate of embedding layers should not be scaled down with widths while **?** searched for a unified learning rate for all layers on each model size, making embeddings learned too slow, and matrix-like parameters dominate the training dynamics.

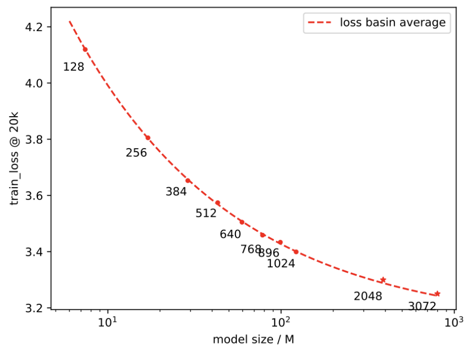
Scaling law fail outside the loss basins. Theoretically, μP suggests similar train ing dynamics across different widths for arbitrary HP, but we observe that the scaling laws fit well only in loss basins. According to our follow-up experiments, this observation is regardless of data or training steps and still exists when models are more sufficiently trained. Thus, we suggest searching for the best HPs first anyway.

Smaller models are more vulnerable. We grid-search for the best HPs for 6-layer models with batch size 32 and found $(5e-4, 0.02, 3.5)$ being inside the loss basin. As shown in Figure 1a (red line), nanoLM works perfectly for this single point. We then explored other points around it and found that the scaling laws have larger deviations than 12-layer models. This is potentially because small models are more vulnerable to slight misalignment of loss landscapes across μ-Transfer. However, we easily balance-off this deviation by fitting scaling laws with the average results across all these HPs near the loss basin. This works perfectly as shown in Figure 1b, and can be very practical in applying nanoLM because we observe in Figure 1a that larger widths (*e.g.,* 2048, 3072) have low variance in training loss w.r.t different HPs.

General conditions for scaling laws. Previous scaling laws directly search for HPs on each scale, and the optimal HPs do not satisfy μP function. This indicates that μP is a sufficient but not necessary condition for scaling laws, and scaling law itself may represent a higher level of universality.



(a) Scaling law for training loss with different HPs for 6-layer models.



(b) Scaling law for average training loss in the loss basin of 6-layer models.

Figure 1: Results with 6-layer Models.

Table 2: Pre-training data ratio.

Dataset	Sampling prop(%)	Total tokens(B)
Arkiv	6.04	28.31
Books	5.22	24.46
Falcon RefinedWeb	20.81	97.49
Falcon RefinedWeb(wiki-like)	49.78	233.21
OpenWebText2	3.11	14.59
StackExchange	3.81	17.84
Github	10.18	47.70
Wikipedia	1.03	4.82

2 Pre-training data ratio

3 The hyperparameter settings for all experiments

The specific parameters of the experiment are as follows. (1) The parameters of the model are: vocab_size = 50304; block_size = 1204; n_layer = [12, 32, 64]; num_heads = 12; dropout = 0.0; output_mult = 1.0; zero_query = True; zero_emb = True. hp_tune_actual_width = [128, 256, 384, 512, 640, 768, 896, 1024, 2048, 4096, 8192]; (2) The parameters of the data are: input_length = 512; mlm_probability = 0.15; mean_noise_span_length = 3.0; num_workers = 2; (3) The parameters of the optimizer are: name = adamwscale; batch_size = [16, 512]; total_steps = [7000, 10000]; warmup_steps = 5000; lr_scheduler = cosine; weight_decay = 0.0; grad_clip = 1.0; grad_acc = 1; final_cosine = 1e-5; base_lr = [5e-4, 1e-3, 5e-3, 1e-2, 3e-2, 5e-2, 7e-2, 1e-1].

4 Based on the basic width of 256, the grid search results

Table 3: grid search on base width = 256. The specific parameters of the experiment are: n_layer = 12, batch_size = 16, hp_tune_actual_width = 256, total_steps = 7000, base_lr = [5e-4, 1e-3, 5e-3, 1e-2, 3e-2, 5e-2, 7e-2, 1e-1].

lr	5e-4	1e-3	5e-3	1e-2	3e-2	5e-2	7e-2	1e-1
12-layer BERT loss	7.37	7.27	5.01	4.39	3.9	4.17	5.24	6.97
12-layer GPT loss	7.3	7.03	5.97	5.57	3.74	5.86	7.22	7.25
12-layer T5 loss	6.85	6.33	5.37	5.13	4.71	5.14	5.28	6.45

Table 4: grid search on base width = 256. The specific parameters of the experiment are: n_layer = 64, batch_size = 512, hp_tune_actual_width = 256, total_steps = 10000, base_lr = [1e-4, 5e-4, 7e-4, 1e-3, 3e-3, 5e-3, 7e-3, 1e-2].

lr	1e-4	5e-4	7e-4	1e-3	3e-3	5e-3	7e-3	1e-2
64-layer GPT loss	4.35	3.73	3.69	3.64	8.37	13.3	9.66	8.12

5 Specific loss value

5.1 nanoLM on C4

Table 5: training loss on 12-layer@7k steps. The specific parameters of the experiment are: n_layer = 12, batch_size = [16, 512], hp_tune_actual_width = [128, 256, 384, 512, 640, 768, 896, 1024], base_lr = [1e-3, 3e-2].

width	128	256	384	512	640	768	896	1024
BERT w/o μP	4.25	3.71	3.59	3.52	3.47	3.42	3.37	3.40
BERT with μP	4.45	3.74	3.63	3.56	3.49	3.47	3.44	3.43
GPT w/o μP	4.73	4.48	4.50	4.42	4.36	4.33	4.29	4.31
GPT with μP	4.52	4.25	4.16	4.10	4.04	4.01	4.00	3.98
T5 w/o μP	5.14	5.06	4.82	4.81	4.66	6.50	5.71	6.03
T5 with μP	5.18	4.71	4.57	4.61	4.60	4.60	4.52	4.49

5.2 nanoLM on MC4

Table 6: training loss on 12-layer@20k steps. The specific parameters of the experiment are: n_layer = 12, batch_size = [16, 512], hp_tune_actual_width = [128, 256, 384, 512, 640, 768, 896, 1024], base_lr = [1e-3, 5e-2].

width	128	256	384	512	640	768	896	1024
Bert	2.79	1.36	1.24	1.17	1.14	1.10	1.08	1.06
T5	2.15	2.05	1.76	1.62	1.54	1.51	1.45	1.41
GPT	1.50	1.38	1.34	1.32	1.32	1.31	1.30	1.29
LLAMA	1.58	1.48	1.46	1.44	1.40	1.40	1.39	1.38

5.3 nanoLM on pretrain data benchmark With fsdp

Table 7: training loss on 32-layer@7k steps. The specific parameters of the experiment are: n_layer = 32, batch_size = 512, hp_tune_actual_width = [256, 384, 512, 640, 768, 896, 1024, 2048, 4096, 8192], total_steps = 7000, base_lr = 5e-2.

width	256	384	512	640	768	896	1024	2048	8192
GPT with μP	3.92	3.76	3.65	3.59	3.54	3.49	3.47	3.45	3.41

5.4 Megatron on pretrain data benchamark

Table 8: training loss on 64-layer@10k steps. The specific parameters of the experiment are: n_layer = 64, batch_size = 512, hp_tune_actual_width = [384, 512, 640, 768, 896, 1024, 2048, 8192], total_steps = 10000, base_lr = 1e-3.

width	256	384	512	640	768	896	1024	2048	8192
GPT with μP	3.656	3.389	3.298	3.215	3.198	3.087	3.080	2.958	2.883