A. Experimental Settings

Supervised image classification. For all supervised classification experiments on ImageNet-1K, we follow the training recipes from ConvNeXt [54]. For ConvNeXt-B and ConvNeXt-L, we use the original hyperparameters without modification. ViT-B and ViT-L models use the same hyperparameters as ConvNeXt-B, except that for ViT-L, the beta parameters for AdamW are set to (0.9, 0.95), and the depth path rates are set to 0.1 for ViT-B and 0.4 for ViT-L.

Diffusion models. We use the official implementation [56] for training all DiT models. We find that the default learning rate is suboptimal for the models considered in this paper. To address this, we conduct a simple learning rate search with the LN models and apply the tuned learning rates directly to the DyT models. We also observe that the zero initialization negatively affects the performance of DyT models. Therefore, we retain the zero initialization for LN models but remove the zero initialization for DyT models.

Large Language Models. In our implementation of LLaMA models [21, 79, 80] with DyT, we introduce an additional learnable scalar parameter immediately after the embedding layer, before any Transformer blocks. We initialize it to the square root of the model embedding dimension $\sqrt{d}$. Without this scaling scalar, we find that the magnitudes of model activations at the beginning of training are too small, and the training struggles to progress. The issue is mitigated by incorporating a learnable scalar, and the model can converge normally. This addition of a scalar is similar to the original Transformer [82] design, which uses a fixed scalar of the same value at the same position.

We train all our LLaMA models on the Pile dataset [25]. We use the codebase from `FMS-FSDP` [23], which provides a default training recipe for the 7B model that closely follows the LLaMA 2 paper [80]. We maintain the learning rate at the default 3e-4 for 7B and 13B and 1.5e-4 for 34B and 70B, in line with LLaMA 2. The batch size is set to 4M tokens and each model is trained on a total of 200B tokens.

For evaluation, we test the pretrained models on 15 zero-shot commonsense reasoning tasks from `lm-eval` [24]: `anli_r1`, `anli_r2`, `anli_r3`, `arc_challenge`, `arc_easy`, `boolq`, `hellaswag`, `openbookqa`, `piqa`, `record`, `rte`, `truthfulqa_mc1`, `truthfulqa_mc2`, `wic`, and `winogrande`. The selection closely follows that of OpenLLaMA [26]. We report the average performance across all tasks.

Self-supervised learning in speech. For both wav2vec 2.0 models, we retain the first group normalization layer from the original architecture, as it functions primarily as data normalization to handle the unnormalized input data. We use the official implementation [58] without modifying hyperparameters for both the Base and Large models. We report the final validation loss.

Other tasks. For all other tasks, MAE [34], DINO [14], HyenaDNA [60] and Caduceus [69], we directly use the publicly released code [31, 46, 55, 57], without hyperparameter tuning, for both models with LN and DyT.

B. Efficiency of DyT

We benchmark the LLaMA 7B model with RMSNorm or DyT by measuring the total time required for 100 forward passes (inference) and 100 forward-backward passes (training) on a single sequence of 4096 tokens. We follow the official LLaMA instructions to load the model from Hugging Face [40]. Table 10 reports the time taken for RMSNorm and DyT layers, as well as for the entire model, when running on an Nvidia H100 GPU with BF16 precision. This set of measurements is taken without performance optimizations. DyT layers reduce computation time compared to RMSNorm layers.

	inference		training	
LLaMA 7B	layer	model	layer	model
RMSNorm	2.1s	14.1s	8.3s	42.6s
DyT	1.0s	13.0s	4.8s	39.1s
reduction	↓52.4%	↓7.8%	↓42.2%	↓8.2%

Table 10. **Inference and training latency (BF16 precision) for LLaMA 7B with RMSNorm or DyT.** DyT achieves a substantial reduction in both inference and training time. Results are measured without any performance optimizations for both layers.

We also measure the forward and backward pass latency after compiling the DyT and RMSNorm layers using `torch.compile`. We do not compile the entire LLaMA model, as we find that compiling the full model increases latency for this particular LLaMA implementation, and only compiling the DyT/RMSNorm layers produces the most efficient runs. The results, shown in Table 11, indicate that after compilation, the latency of RMSNorm and DyT layers becomes nearly identical.

	inference		training	
LLaMA 7B	layer	model	layer	model
RMSNorm	0.3s	12.3s	3.9s	38.9s
DyT	0.3s	12.3s	3.9s	38.9s

Table 11. **Inference and training latency (BF16 precision) for a compiled LLaMA 7B with RMSNorm or DyT.** After compilation, the latency of RMSNorm and DyT layers are nearly identical.

An important distinction of DyT is that it is an element-wise operation and does not require a reduction operation within itself, compared to normalization layers. This could make it faster on hardware where reduction is a bottleneck. Additionally, even on conventional GPUs, DyT could offer opportunities for further optimization, e.g., fusing it

with the preceding matrix multiplication layer from the last residual block.

C. Results on Initialization of α for LLMs

To further illustrate the impact of α_0 tuning, Figure 10 presents heatmaps of loss values at 30B tokens of two LLaMA models. Both models benefit from higher α_0 in attention blocks, leading to reduced training loss.

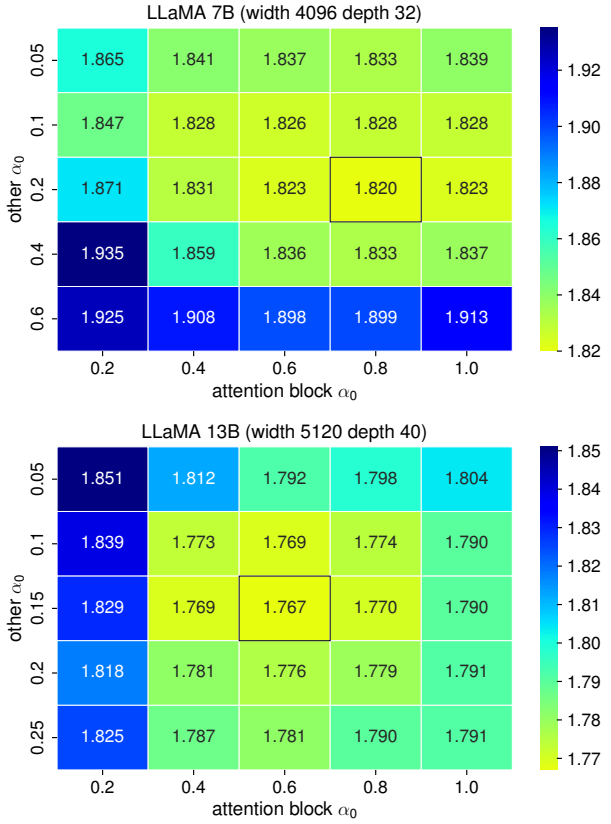

Figure 10. **Heatmaps of loss values for different α_0.** Both LLaMA models benefit from increased α_0 in attention blocks.

Model width primarily determines α_0. We also investigate the influence of model width and depth on the optimal α_0. We find that the model width is critical in determining the α_0, while model depth has minimal influence. Table 12 shows the optimal α_0 values across different widths and depths, showing that wider networks benefit from smaller α_0 values for optimal performance. On the other hand, model depth has negligible impact on the choice of α_0.

As can be seen in Table 12, the wider the network, the more uneven initialization for "attention" and "other" is needed. We hypothesize that the sensitivity of LLM's α initialization is related to their excessively large widths compared to other models.

width / depth	8	16	32	64
1024	1.0/1.0	1.0/1.0	1.0/1.0	1.0/1.0
2048	1.0/0.5	1.0/0.5	1.0/0.5	1.0/0.5
4096	0.8/0.2	0.8/0.2	0.8/0.2	0.8/0.2
8192	0.2/0.05	0.2/0.05	0.2/0.05	0.2/0.05

Table 12. **Optimal α_0 (attention / other) across model widths and depths in LLaMA training.** Model width significantly impacts the choice of α_0, with wider networks requiring smaller values. In contrast, model depth has negligible influence.

D. Comparison with Other Methods

To further assess DyT's effectiveness, we compare it with other methods that enable training Transformers without normalization layers. These methods can be broadly categorized into initialization-based and weight-normalization-based methods. We consider two popular initialization-based methods, Fixup [39, 90] and SkipInit [6, 18]. Both methods aim to mitigate training instabilities by adjusting the initial parameter values to prevent large gradients and activations at the start of training, thereby enabling stable learning without normalization layers. In contrast, weight-normalization-based methods impose constraints on network weights throughout training to maintain stable learning dynamics in the absence of normalization layers. We include one such method, σReparam [88], which controls the spectral norm of the weights to promote stable learning.

model	LN	Fixup	SkipInit	σReparam	DyT
ViT-B	82.3%	77.2%	74.1%	82.5%	**82.8%**
ViT-L	83.1%	78.1%	75.6%	83.0%	**83.6%**
MAE ViT-B	83.2%	73.7%	73.1%	83.2%	**83.7%**
MAE ViT-L	85.5%	74.1%	74.0%	85.4%	**85.8%**

Table 13. **Classification accuracy on ImageNet-1K.** DyT consistently achieves superior performance over other methods.

Table 13 summarizes the results of two ViT-based tasks. We closely follow the original protocols outlined in their respective papers. However, we find that both initialization-based methods, Fixup and SkipInit, require much lower learning rates to prevent divergence. To ensure a fair comparison, we conduct a learning rate search for all methods, including DyT. This produces results that differ from those reported in Section 5, where no hyperparameter is tuned. Overall, the results show that DyT consistently outperforms all other tested methods across different configurations.

E. Replacing Batch Normalization with DyT

We investigate the potential of replacing BN with DyT in classic ConvNets such as ResNet-50 [33] and VGG19 [71]. Both models are trained on the ImageNet-1K dataset [19] using the training recipes provided by `torchvision`. The

model	LN (original)	DyT (original)	LN (tuned)	DyT (tuned)
ViT-B	82.3% (4e-3)	82.5% (4e-3)	-	82.8% (6e-3)
ViT-L	83.1% (4e-3)	83.6% (4e-3)	-	-
ConvNeXt-B	83.7% (4e-3)	83.7% (4e-3)	-	-
ConvNeXt-L	84.3% (4e-3)	84.4% (4e-3)	-	-
MAE ViT-B	83.2% (2.4e-3)	83.2% (2.4e-3)	-	83.7% (3.2e-3)
MAE ViT-L	85.5% (2.4e-3)	85.4% (2.4e-3)	-	85.8% (3.2e-3)
DINO ViT-B (patch size 16)	83.2% (7.5e-4)	83.4% (7.5e-4)	83.3% (1e-3)	-
DINO ViT-B (patch size 8)	84.1% (5e-4)	84.5% (5e-4)	-	-
DiT-B	64.9 (4e-4)	63.9 (4e-4)	-	-
DiT-L	45.9 (4e-4)	45.7 (4e-4)	-	-
DiT-XL	19.9 (4e-4)	20.8 (4e-4)	-	-
wav2vec 2.0 Base	1.95 (5e-4)	1.95 (5e-4)	-	1.94 (6e-4)
wav2vec 2.0 Large	1.92 (3e-4)	1.91 (3e-4)	-	-
HyenaDNA	85.2% (6e-4)	85.2% (6e-4)	-	-
Caduceus	86.9% (8e-3)	86.9% (8e-3)	-	-

Table 14. **Performance comparison between original and tuned learning rates for LN and DyT models.** Results show that tuning learning rates provide only modest performance improvements for DyT models, suggesting that the default hyperparameters optimized for LN models are already well-suited for DyT models. Entries marked with "-" indicate no performance gain over the original learning rate. The values in parentheses represent the learning rate used.

Model	LN	DyT ($\alpha_0 = 0.5$)	DyT (tuned)
ViT-B	82.3%	82.5%	82.6% ($\alpha_0 = 1.0$)
ViT-L	83.1%	83.6%	-
ConvNeXt-B	83.7%	83.7%	-
ConvNeXt-L	84.3%	84.4%	-
MAE ViT-B	83.2%	83.2%	83.4% ($\alpha_0 = 1.0$)
MAE ViT-L	85.5%	85.4%	-
DINO ViT-B (patch 16)	83.2%	83.4%	-
DINO ViT-B (patch 8)	84.1%	84.5%	-
DiT-B	64.9	63.9	-
DiT-L	45.9	45.7	-
DiT-XL	19.9	20.8	-
wav2vec 2.0 Base	1.95	1.95	-
wav2vec 2.0 Large	1.92	1.91	1.90 ($\alpha_0 = 1.0$)
HyenaDNA	85.2%	85.2%	-
Caduceus	86.9%	86.9%	-

Table 15. **Impact of tuning the α_0 in DyT models.** Optimizing α_0 from the default value ($\alpha_0 = 0.5$) yields only minor performance gains for select DyT models, implying the default initialization already achieves near-optimal performance. Entries marked with "-" indicate no improvement over the default α_0.

DyT models are trained using the same hyperparameters as their BN counterparts.

model	BN	DyT
ResNet-50	76.2%	68.9%
VGG19	72.7%	71.0%

Table 16. **ImageNet-1K classification accuracy with BN and DyT.** Replacing BN with DyT in ResNet-50 and VGG19 results in a performance drop, indicating that DyT cannot fully substitute BN in these architectures.

The results are summarized in Table 16. Replacing BN with DyT led to a noticeable drop in classification accuracy for both models. These findings indicate that DyT is struggling to fully replace BN in these classic ConvNets. We hypothesize this could be related to BN layers being more frequent in these ConvNets, where they appear once with every weight layer, but LN only appears once per several weight layers in Transformers.

F. Hyperparameters

We present additional experiments to evaluate the impact of hyperparameter tuning, specifically focusing on the learning rate and initialization of α for all non-LLM models.

Tuning learning rate. Table 14 summarizes performance comparisons between models trained with original versus tuned learning rates. Results indicate that tuning the learning rate provides only modest performance improvements for DyT models. This suggests that the original hyperparameters, initially optimized for LN models, are already well-suited for DyT models. This observation underscores the inherent similarity between the DyT and LN models.

Tuning initial value of α. We also investigate the effects of optimizing α_0 for DyT models, as presented in Table 15. Findings show minor performance enhancements for select models when α_0 is tuned, indicating that the default value ($\alpha_0 = 0.5$) generally achieves near-optimal performance.