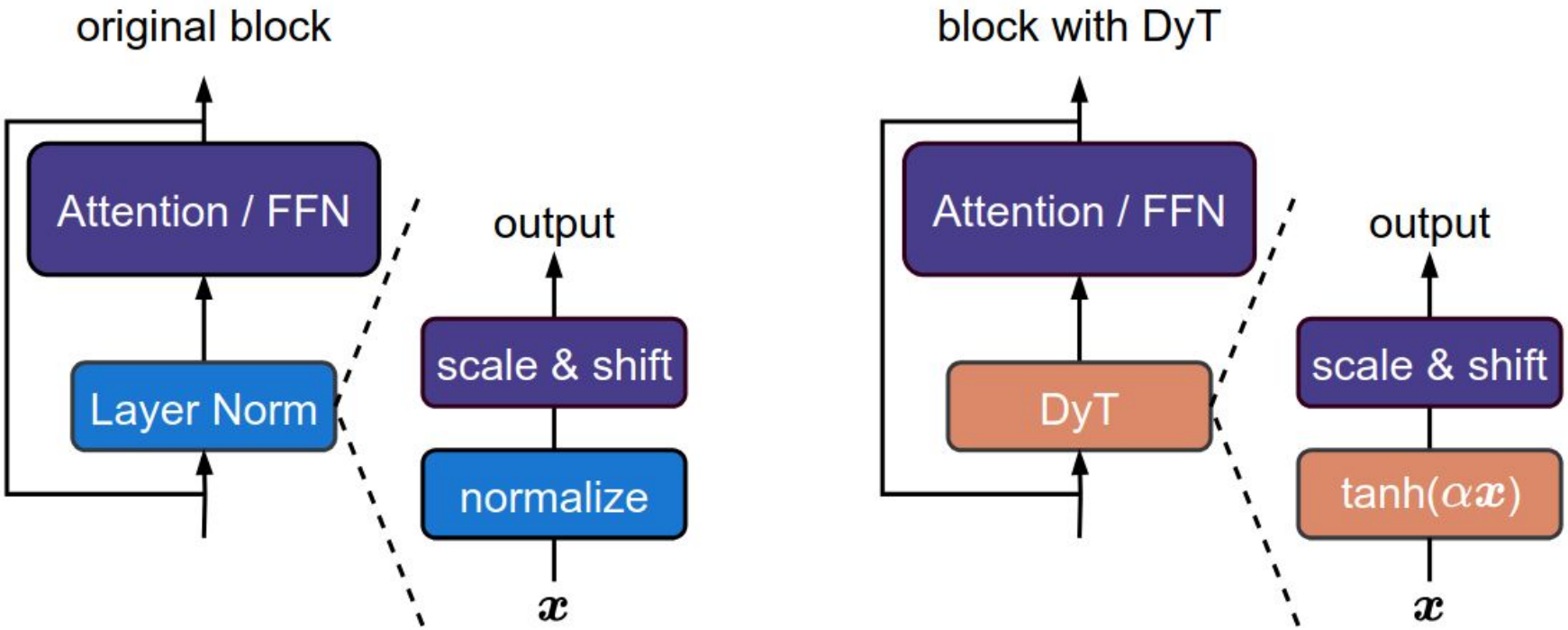
Transformers without Normalization

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, Zhuang Liu

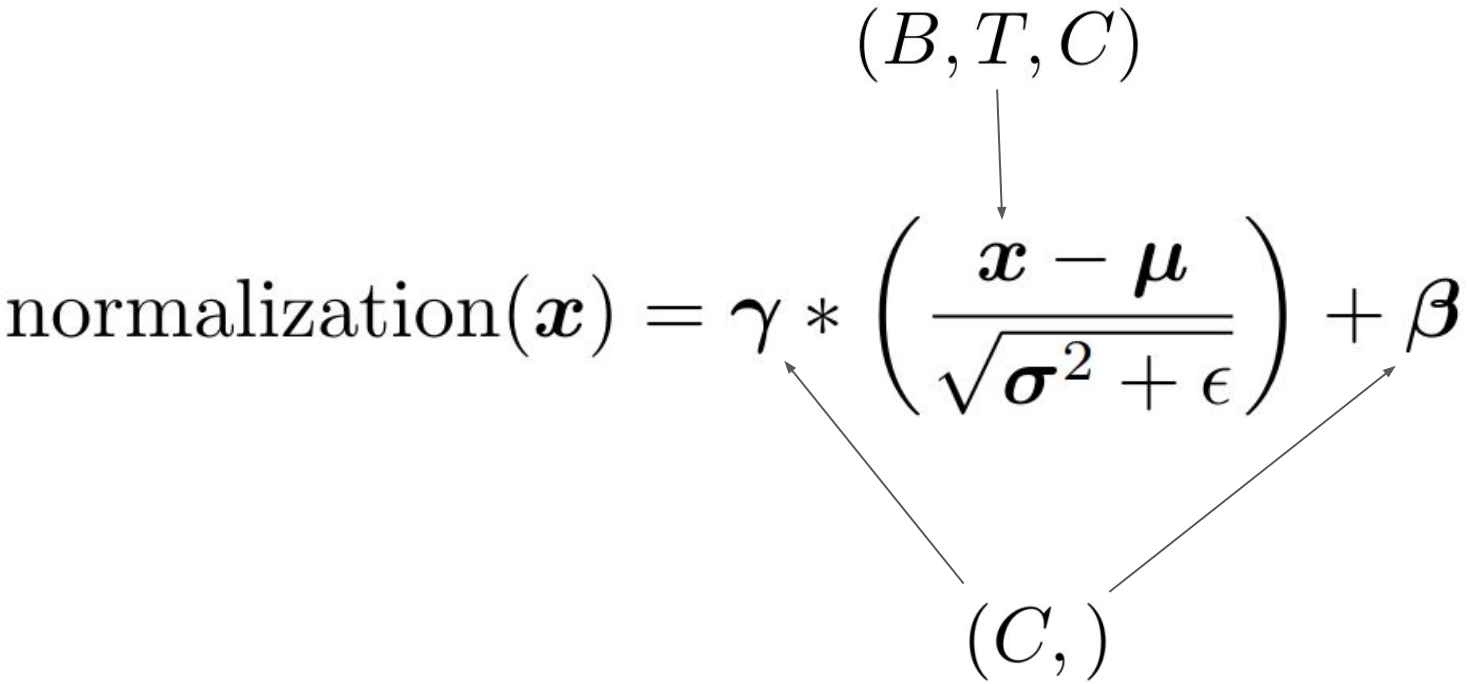
FAIR, Meta

Introduction

Layer Normalization (LN) —> Dynamic Tanh (DyT)



Normalization Layers



(Main) Types of Normalization

Batch Normalization (2015) $\mu_k = \frac{1}{BT}\sum_{i,j} x_{ijk} \quad \sigma_k^2 = \frac{1}{BT}\sum_{i,j}(x_{ijk} - \mu_k)^2$

Layer Normalization (2016) $\mu_{ij} = \frac{1}{C}\sum_k x_{ijk} \quad \sigma_{ij}^2 = \frac{1}{C}\sum_k (x_{ijk} - \mu_{ij})^2$

RMS Normalization (2019) $\mu_{ij} = 0 \qquad \sigma_{ij}^2 = \frac{1}{C}\sum_k x_{ijk}^2$

Methodology

What does Normalization actually do?

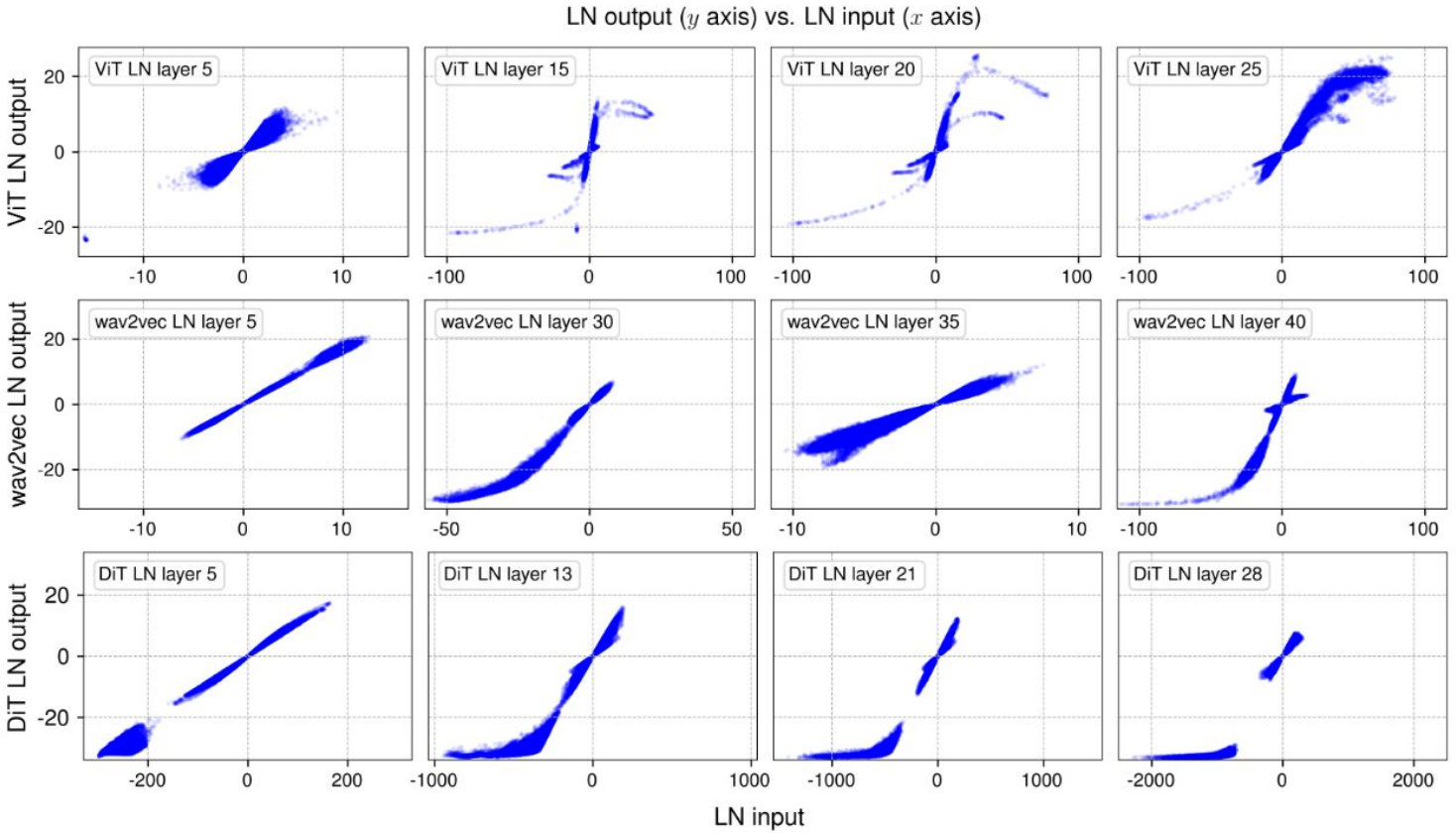


Figure 2 Output vs. input of selected layer normalization (LN) layers in Vision Transformer (ViT) (Dosovitskiy et al., 2020), wav2vec 2.0 (a Transformer model for speech) (Baevski et al., 2020), and Diffusion Transformer (DiT) (Peebles and Xie, 2023).

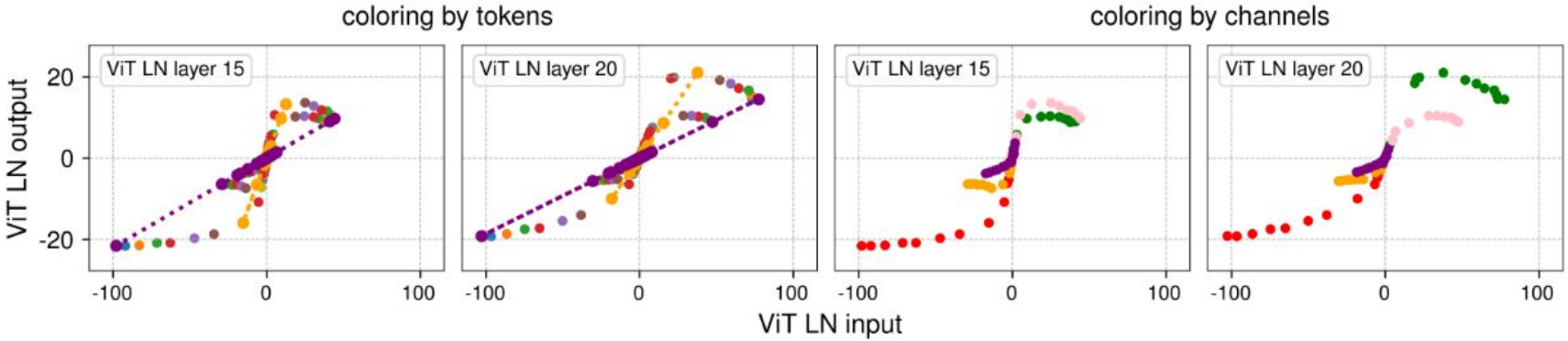


Figure 4 Output vs. input of two LN layers, with tensor elements colored to indicate different channel and token dimensions. The input tensor has a shape of (samples, tokens, and channels), with elements visualized by assigning consistent colors to the same tokens (left two panels) and channels (right two panels). *Left two panels*: points representing the same token (same color) form straight lines across different channels, as LN operates linearly across channels for each token. Interestingly, when plotted collectively, these lines form a non-linear tanh-shaped curve. *Right two panels*: each channel's input spans different ranges on the x-axis, contributing distinct segments to the overall tanh-shaped curve. Certain channels (e.g., red, green, and pink) exhibit more extreme x values, which are squashed by LN.

Resemblance to tanh

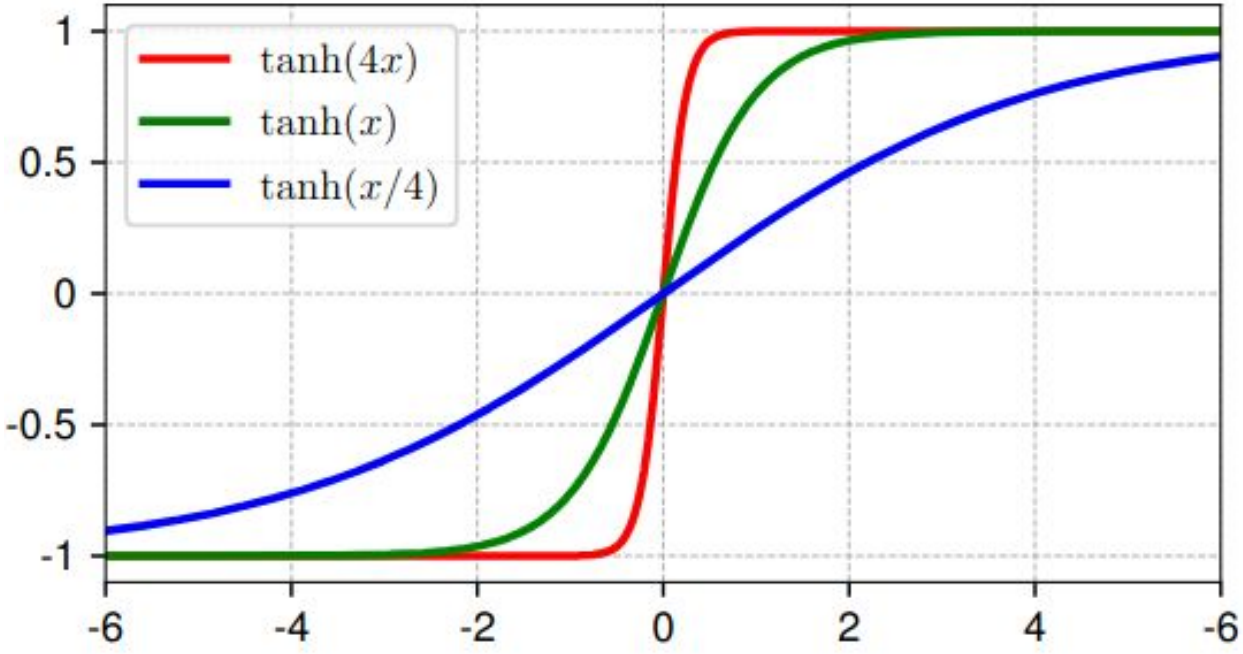


Figure 3 $\tanh(\alpha x)$ with three different α values.

Dynamic Tanh (DyT)

$$\mathrm{DyT}(\boldsymbol{x}) = \boldsymbol{\gamma} * \tanh(\alpha \boldsymbol{x}) + \boldsymbol{\beta}$$

learnable

Algorithm 1 Pseudocode of DyT layer.

```
# input x has the shape of [B, T, C]
# B: batch size, T: tokens, C: dimension

class DyT(Module):
    def __init__(self, C, init_α):
        super().__init__()
        self.α = Parameter(ones(1) * init_α)
        self.γ = Parameter(ones(C))
        self.β = Parameter(zeros(C))

    def forward(self, x):
        x = tanh(self.alpha * x)
        return self.γ * x + self.β
```

Experimental Results

Supervised Vision - ImageNet-1k

model	LN	DyT	change
ViT-B	82.3%	82.5%	↑0.2%
ViT-L	83.1%	83.6%	↑0.5%
ConvNeXt-B	83.7%	83.7%	-
ConvNeXt-L	84.3%	84.4%	↑0.1%

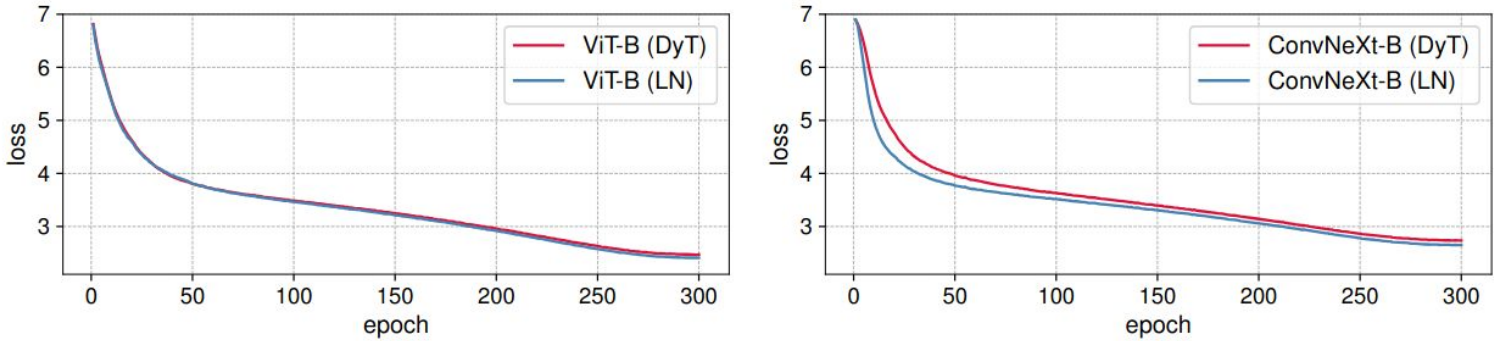


Figure 5 Training loss curves for ViT-B and ConvNeXt-B models. The loss curves for both model types exhibit similar patterns between LN and DyT, suggesting that LN and DyT may share similar learning dynamics.

Self-supervised Vision - ImageNet-1k

model	LN	DyT	change
MAE ViT-B	83.2%	83.2%	-
MAE ViT-L	85.5%	85.4%	↓0.1%
DINO ViT-B (patch size 16)	83.2%	83.4%	↑0.2%
DINO ViT-B (patch size 8)	84.1%	84.5%	↑0.4%

Diffusion Models - ImageNet-1k

model	LN	DyT	change
DiT-B	64.9	63.9	↓1.0
DiT-L	45.9	45.7	↓0.2
DiT-XL	19.9	20.8	↑0.9

Image generation quality (FID, lower is better) on ImageNet.

Large Language Models - The Pile

score / loss	RMSNorm	DyT	change
LLaMA 7B	0.513 / 1.59	0.513 / 1.60	- / ↑0.01
LLaMA 13B	0.529 / 1.53	0.529 / 1.54	- / ↑0.01
LLaMA 34B	0.536 / 1.50	0.536 / 1.50	- / -
LLaMA 70B	0.549 / 1.45	0.549 / 1.45	- / -

Language models' training loss and average performance with 15 zero-shot `lm-eval` tasks.

Large Language Models

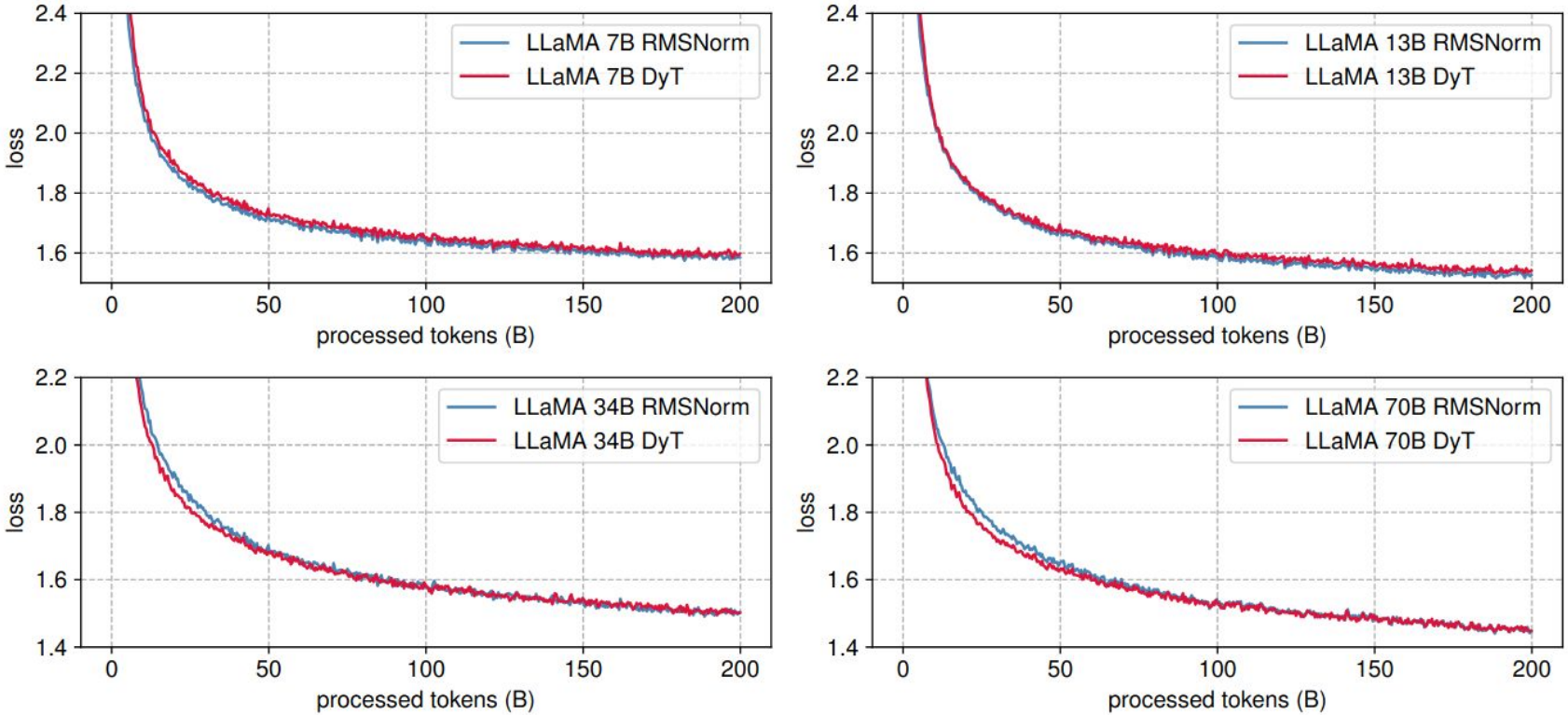


Figure 6 LLaMA pretraining loss. The loss curves of DyT and RMSNorm models are closely aligned across model sizes.

Self-supervised speech - LibriSpeech

model	LN	DyT	change
wav2vec 2.0 Base	1.95	1.95	-
wav2vec 2.0 Large	1.92	1.91	↓0.01

Speech pretraining validation loss on LibriSpeech.

DNA Sequence Modelling - human genome data

model	LN	DyT	change
HyenaDNA (Nguyen et al., 2024)	85.2%	85.2%	-
Caduceus (Schiff et al., 2024)	86.9%	86.9%	-

DNA classification accuracy on GenomicBenchmarks,

Analysis

Efficiency of DyT (!)

	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%

Inference and training latency (BF16 precision) for LLaMA 7B with RMSNorm or DyT.

Ablation study

Replacing tanh function

model	identity	tanh	hardtanh	sigmoid
ViT-S	58.5% → failed	**80.3%**	79.9%	79.6%
ViT-B	61.0% → failed	**82.5%**	82.2%	81.6%

ImageNet-1K classification accuracy with different squashing functions.

Remove alpha

model	tanh	hardtanh	sigmoid
without α	81.1%	80.7%	80.7%
with α	**82.5%**	**82.2%**	**81.6%**

ImageNet-1K classification accuracy with ViT-B.

Ablation study

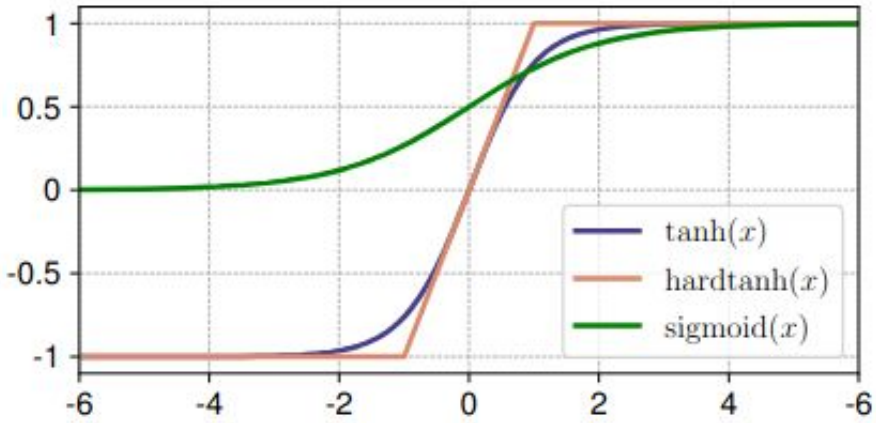


Figure 7 Curves of three squashing functions: tanh, hardtanh, and sigmoid. All three functions squash inputs into a bounded range, but $\tanh(x)$ achieves the best performance when used in DyT layers. We suspect it is due to its smoothness and zero-centered properties.

Values of alpha

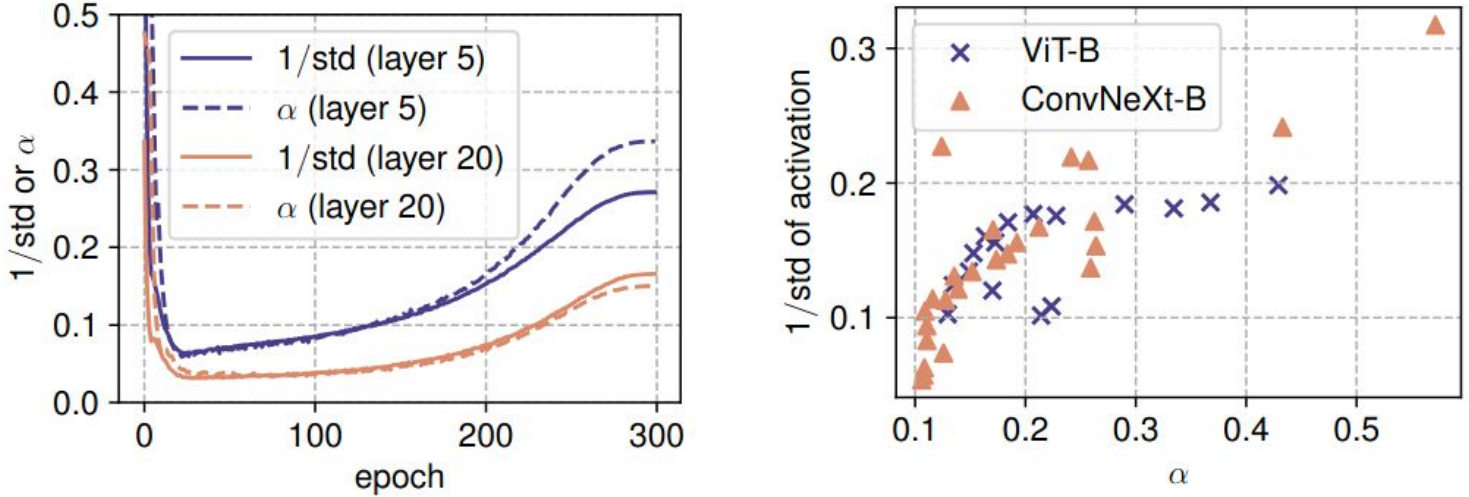


Figure 8 *Left:* For two selected DyT layers from the ViT-B model, we track α and the inverse of the standard deviation (1/std) of activations at the end of each epoch, observing that they evolve together during training. *Right:* We plot the final α values of two trained models, ViT-B and ConvNeXt-B, against the 1/std of the input activations, demonstrating a strong correlation between the two values.

Other non-normalization methods

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%**

Classification accuracy on ImageNet-1K.

Initialization of alpha

Initialization of alpha for non-LLMs - little effect

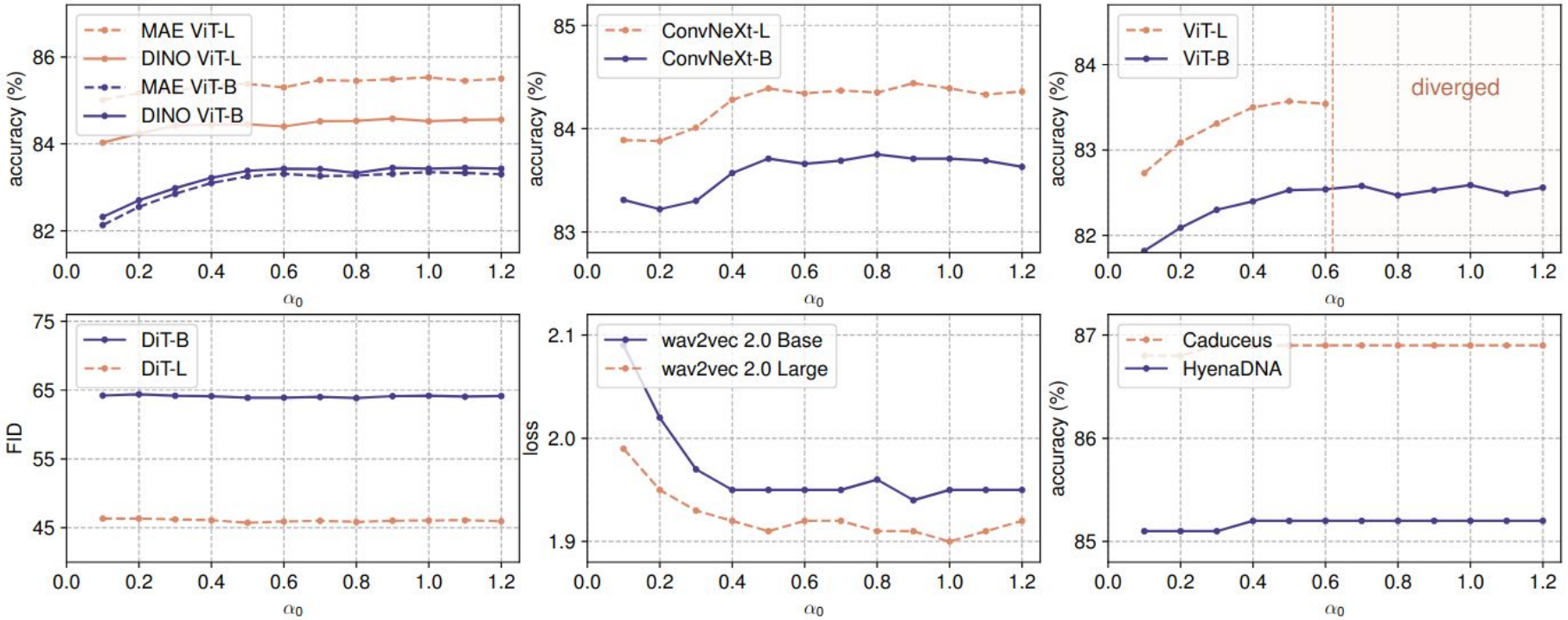


Figure 9 Performance of different tasks across different α_0 values. We benchmark the performance of all non-LLM tasks used in Section 5 with different initial values of α. Performance remains stable across a wide range of α_0 values. The only exception is that supervised ViT-L models (top right panel) will diverge for α_0 values larger than 0.6.

Initialization of alpha for LLMs - significant effect

model	width	depth	optimal α_0 (attention/other)
LLaMA 7B	4096	32	0.8/0.2
LLaMA 13B	5120	40	0.6/0.15
LLaMA 34B	8196	48	0.2/0.05
LLaMA 70B	8196	80	0.2/0.05

Table 11 Optimal α_0 for different LLaMA models. Larger models require smaller α_0 values. We find it is important to initialize α differently in (1) attention blocks ("attention"), versus (2) the FFN blocks, and the final DyT layer before outputs ("other"). α_0 in attention blocks require larger values.

Initialization of alpha for LLMs

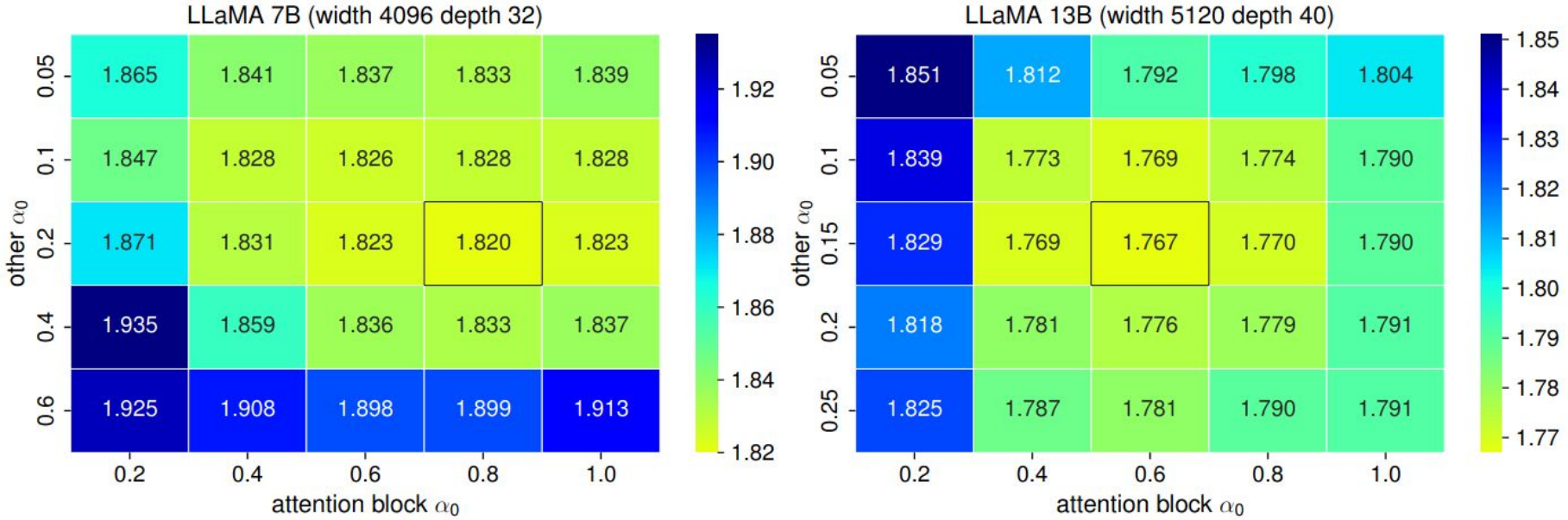


Figure 11 Heatmaps of loss values at 30B tokens for different α_0 settings. Both LLaMA models benefit from increased α_0 in attention blocks.

Initialization of alpha for LLMs

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.

Conclusion

Conclusion and Limitations

- LN is usually tanh-like in practice
- DyT can replace LN in Transformers while keeping performance
- DyT seems to be faster (both inference and train)
- DyT cannot replace BN in ResNets etc