Transformers without Normalization

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, Zhuang Liu
FAIR Meta, New York University, MIT, Princeton University
March 14, 2025

UNIVERSITÉ CÔTE D'AZUR

Inria

cnrs

3iA Côte d'Azur
Institut interdisciplinaire d'intelligence artificielle

Plan

- ❑ Introduction
- ❑ Background: Normalization Layers
- ❑ Dynamic Tanh Observations
- ❑ Dynamic Tanh Experiments
- ❑ Efficiency of Dynamic Tanh: time, ablations, other methods
- ❑ Conclusion
- ❑ Your questions

Introduction

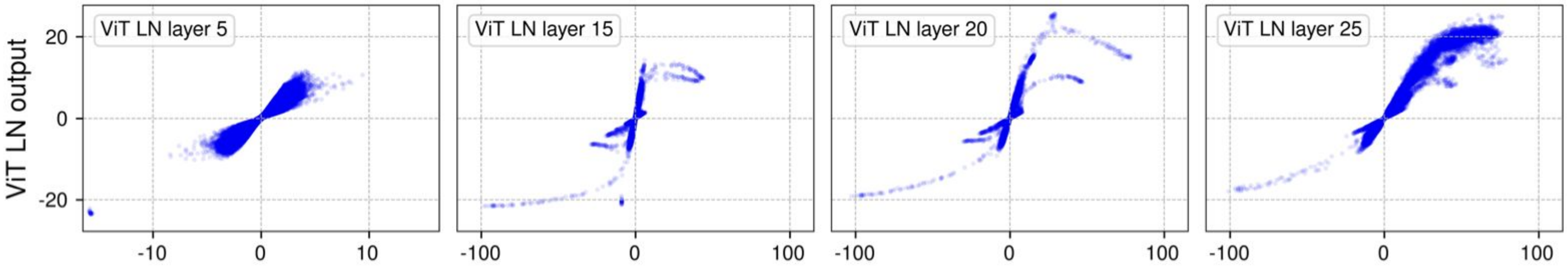
Normalization Layer often produces tanh-like input-output mapping

↓

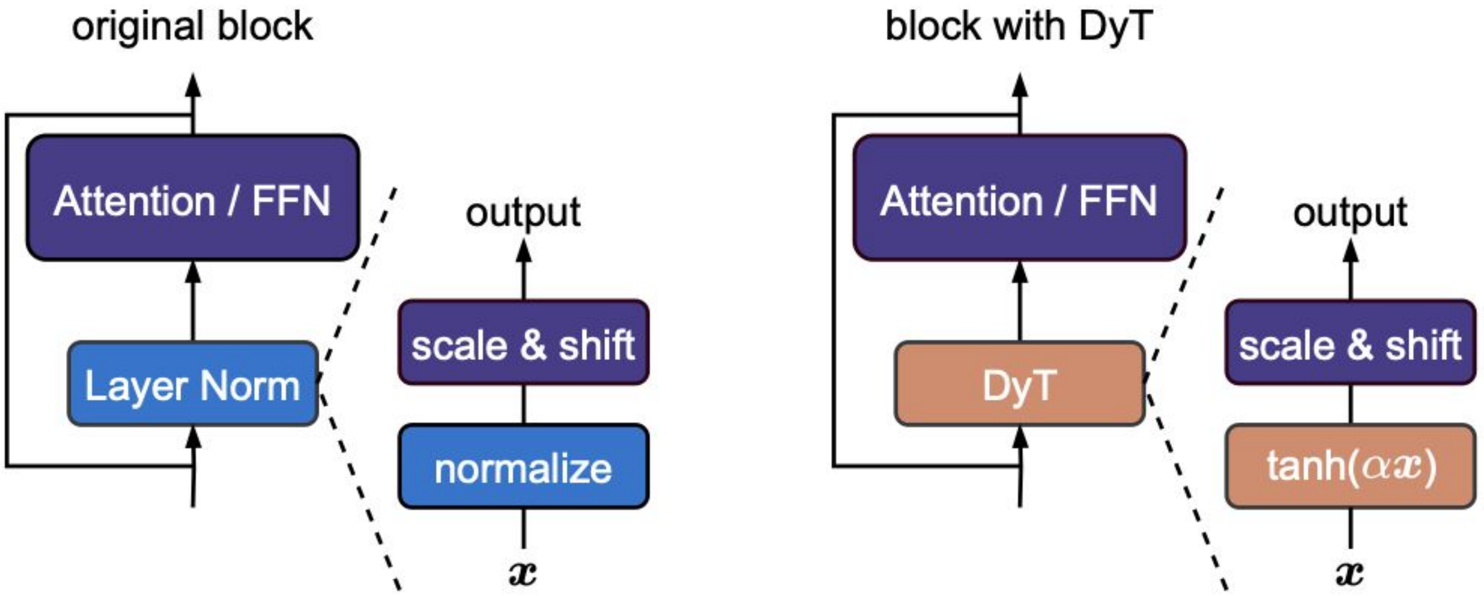
Dynamic Dynamic Tanh (DyT) drop-in replacement for normalization layers in Transformers

$$DyT(x) = \tanh(\alpha x)$$

LN output (y axis) vs. LN input (x axis)



Introduction



Normalization Layers

Layer normalization is a crucial technique in transformer models that helps **stabilize convergence** and **accelerate training** by **normalizing the inputs to each layer**. Due to that, **the model processes information consistently**, regardless of the input's scale or distribution.

Given an input x with shape (B,T,C), where B is the batch size, T is the number of tokens, and C is the embedding dimension per token:

$$\text{normalization}(\boldsymbol{x}) = \boldsymbol{\gamma} * \left(\frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \right) + \boldsymbol{\beta}$$

Batch Normalization (BN)

The first modern normalization layer

It is specifically designed to address **internal covariate shift** : the distribution of activations changes during training due to the constant updates to the network's weights

Batch normalization normalize the activations within each layer, ensuring they follow a consistent distribution with a **mean of zero** and a **standard deviation of one**

Layer Normalization (LN), Root Mean Square Normalization

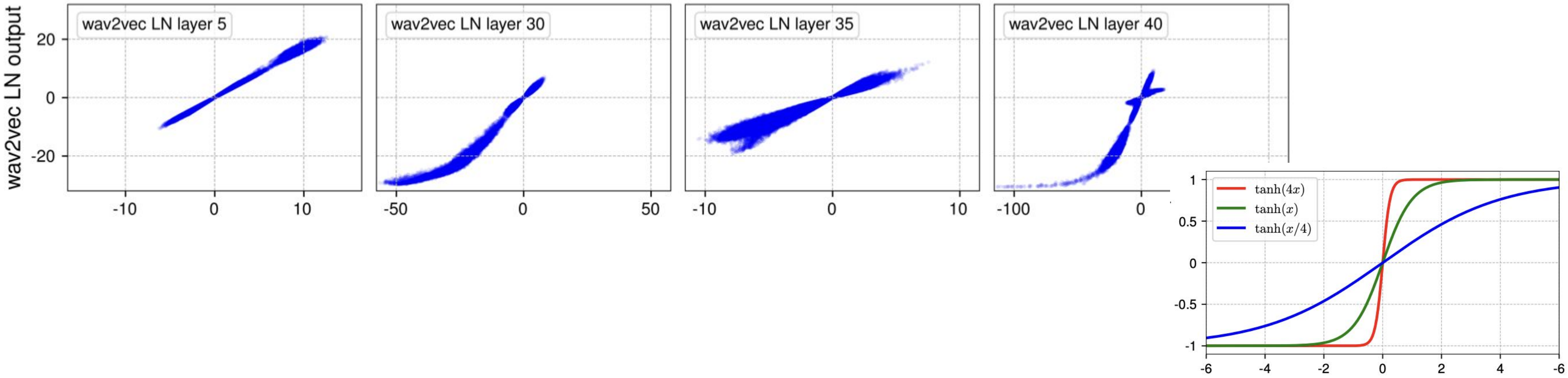
The major types of normalization layers in Transformer architectures

BN does not work effectively with self-attention mechanisms of transformers as it struggles with sequential data, therefore, LN

LN computes the **mean** and **standard deviation for each row across all features**, while in BN the normalization is done across the batch

Tanh-like Mappings with Layer Normalization

For all three models, the input-output relationship in **earlier LN layers** are mostly **linear**, resembling a straight line in an x-y plot. However, the **deeper LN layers** represent **curves** highly resemble **S-shaped curves** represented by a tanh function.



For such an S-shaped curve, the central part represented by points with x values close to zero, is still mainly in a linear shape. However, there are points (“extreme” values) that clearly fall out of this range. Normalization layers’ main effect for these values is to **squash** them into **less extreme values**, more in line with the majority of points.

Dynamic Tanh (DyT)

Given an input tensor x, a DyT layer is defined as follows:

$$DyT(x) = \gamma * \tanh(\alpha x) + \beta$$

- ❏ α is a learnable scalar parameter that allows scaling the input differently based on its range, accounting for varying x scales
- ❏ γ and β are learnable, per-channel vector parameters

Integrating DyT layers into an existing architecture is straightforward: one DyT layer replaces one normalization layer. Other parts of the activation functions or networks themselves remain intact.

Important: DyT is **not** a new type of normalization layer. However, it preserves the effect of normalization layers in **squashing the extreme values** in a non-linear fashion while almost **linearly transforming the very central parts** of the input.

DyT Experiments

Supervised learning in vision (classification accuracy):

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%

Self-supervised learning in vision (accuracy):

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%

DyT Experiments

Diffusion Transformer (DiT) models (image generation quality, lower is better):

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

Large Language Models
(training loss and average performance):

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	- / -

DyT Experiments

Self-supervised learning in speech (validation loss):

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

DNA sequence modeling (classification accuracy):

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

Efficiency of DyT

Time evaluation

To compare and evaluate inference and training time of DyT and LN:
LLaMA 7B with RMSNorm vs LLaMA 7B with DyT to measure the total time taken for inference and for training using a single sequence of 4096 tokens

	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%

Ablations of tanh and α

Removing and replacing tanh by other squashing functions lead to a significant drop in performance (e.g., classification accuracy):

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%

Removing the learnable α while retaining the squashing functions (tanh, hardtanh, and sigmoid) results in performance degradation across all squashing functions:

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

Comparison with Other Methods

- Initialization-based methods: *Fixup* and *SkipInit* ⟶ **adjust the initial parameter values** to prevent large gradients and activations at the start of training enabling stable learning without normalization layers
- Weight-normalization-based methods: *σReparam* ⟶ impose **constraints on network weights** throughout training to maintain stable learning dynamics in the absence of normalization layers

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%**

Conclusion

☆Transformers can be trained without normalization layers☆

★DyT captures the behavior of NLs, thus, it can replace them ★

☆DyT adjusts the input activation range via a learnable scaling factor α☆

★DyT squashes the extreme values through an S-shaped tanh function★