Transformers without Normalization*

Presented by Étienne Perron, Edgar Kappauf and Gaetan Butault

*Published by Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, Zhuang Liu

Introduction

Normalization in Deep Networks

- Normalization layers (e.g., BatchNorm, LayerNorm, RMSNorm) are crucial for stable and efficient training in deep learning models, especially Transformers.

Alternative Approaches

- Fixup, SkipInit, and σ-Reparameterization aim to reduce reliance on normalization through initialization tricks or spectral control but often still need careful tuning.

Dynamic Tanh (DyT)

- [Zhu et al. 2025] is a simpler, stateless alternative.
- Replaces normalization with tanh(αx) + optional affine transformations.
- Matches or exceeds performance of normalized models without extra tuning.

Implication

- DyT questions whether we really need normalization layers: its results suggest that just using a simple squashing function like tanh might be enough, making it possible to build simpler Transformer models.

$$\mathrm{DyT}(x) = \gamma \tanh(\alpha x) + \beta, \quad (1)$$

where

- α is a learnable scalar that rescales the global variance of activations;
- γ and β are learnable per-channel affine parameters identical to those in `LayerNorm`.

Figure 1. Dynamic Tanh (DyT)

Experiences - Summary

We evaluated different normalization strategies within the architecture. Methods tested were **BatchNorm**, **LayerNorm**, and **RMSNorm**.

Each method was applied in three configurations:

- Pre: before the main operation
- Post: after the main operation
- Both: before and after

Identity was used as a control (no normalization), in Pre, Post, or Both positions. Additional experiments were conducted using **DyT** with:

- **Different positions**: Pre, Post, Both
- **Alpha scaling values**: 0.05 and 0.5
- **Activation functions**: Hardtanh and LeakyHardtanh (default: Tanh)

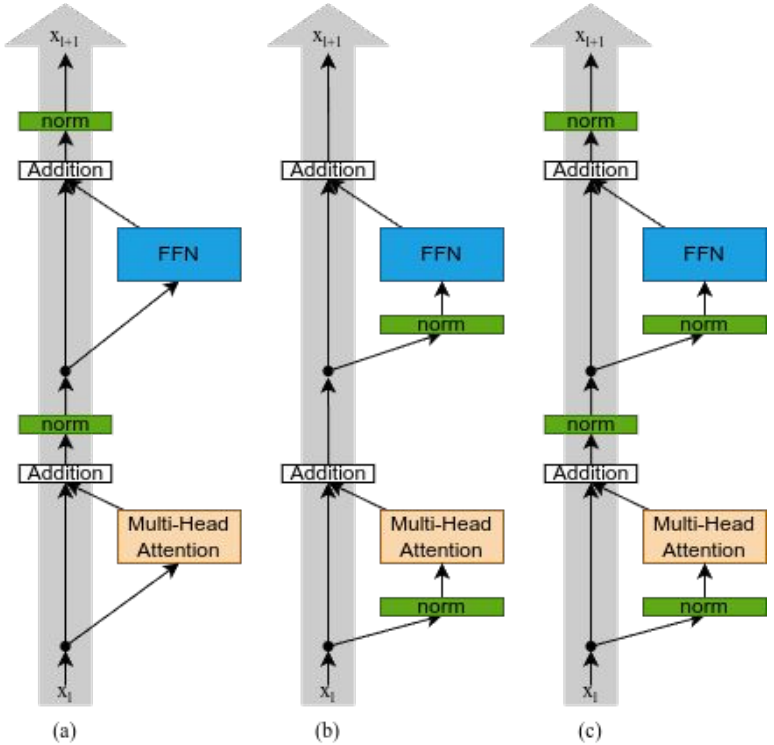

Figure 2. (a) Post-Norm Transformer layer; (b) Pre-Norm Transformer layerl; (c) both-Norm Transformer layer.

Experiences - Results

- **Normalization Position Matters**
  Applying normalization before the main operation (Pre) consistently leads to better performance than applying it after (Post).

- **DyT Performance**
  DyT shows significantly improved results when used in the Pre position but still slightly lags behind traditional methods like LayerNorm or RMSNorm.

- **Activation Function Impact**
  With DyT, using Tanh in the Pre position yields the best results, while Post configurations perform worse across all tested functions.

	Normalization position	
	Post	Pre
BatchNorm	1.74	**1.69**
DyT (default)	3.12	**1.78**
LayerNorm	1.77	**1.70**
RMSNorm	1.78	**1.70**

Figure 3. Loss of different normalization methods with different positions.

	Pre		Post	
Hidden Dimension size	256	512	256	512
Hardtanh	1.78	**1.34**	3.26	3.22
LeakyHardtanh	1.78	**1.34**	3.25	3.12
Tanh	**1.31**	1.35	3.71	3.56

Figure 4. Loss of different combinations of hidden dimensions and DyT functions for Pre and Post contexts.

Experiences - Results

- **Effect of DyT Alpha on Gradient Flow**
  At Epoch 5, different values of alpha in DyT lead to noticeable changes in gradient norms across layers. A moderate alpha (e.g., 0.5) maintains stronger gradients compared to very low values (0.05).

- **Comparison to LayerNorm**
  DyT with well-chosen alpha values shows comparable gradient behavior to LayerNorm, especially in deeper layers.

- **Learning Rate Influence**

  DyT seems to be resilient to learning rates changes than the paper seems to indicate.

- **Normalisation Position**

  Pre-norm always perform better than Post-Norm or both

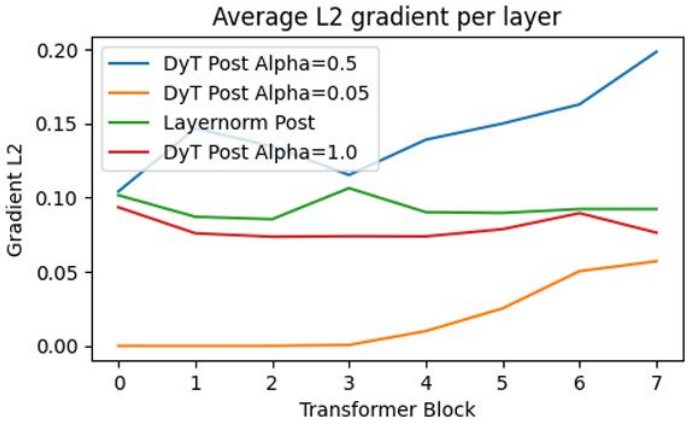

Figure 5. Gradient L2 norm across layers at Epoch 5. Comparing DyT Alpha values to Layernorm Baseline.

Critical Analysis of Our Approach

- **Learning Rate Sensitivity**
  Testing more extreme learning rates (both lower and higher) would clarify how robust DyT is compared to standard normalization methods.

- **Architectural Diversity**
  Applying DyT across different Transformer architectures (e.g., deeper networks, vision or speech models) could test its generalizability.

- **Training Time Consistency**
  Although we scheduled runs by training time, adjustments were needed. A more rigorous setup would involve consistently controlling and reporting training time across all runs.

References

- Figure 1, 3, 4, 5, 6. Étienne Perron, Edgar Kappauf, Gaetan Butault. (2025). Transformers without Normalization.

- Figure 2. Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, Tie-Yan Liu. (2020). On Layer Normalization in the Transformer Architecture. https://arxiv.org/pdf/2002.04745