Transformers without Normalization

Jiachen Zhu[1,2], Xinlei Chen[1], Kaiming He[3], Yann LeCun[1,2], Zhuang Liu[1,4,†]

[1]FAIR, Meta, [2]New York University, [3]MIT, [4]Princeton University
[†]Project lead

Normalization layers are ubiquitous in modern neural networks and have long been considered essential. This work demonstrates that Transformers without normalization can achieve the same or better performance using a remarkably simple technique. We introduce Dynamic Tanh (DyT), an element-wise operation $\mathrm{DyT}(\boldsymbol{x}) = \tanh(\alpha \boldsymbol{x})$, as a drop-in replacement for normalization layers in Transformers. DyT is inspired by the observation that layer normalization in Transformers often produces tanh-like, S-shaped input-output mappings. By incorporating DyT, Transformers without normalization can match or exceed the performance of their normalized counterparts, mostly without hyperparameter tuning. We validate the effectiveness of Transformers with DyT across diverse settings, ranging from recognition to generation, supervised to self-supervised learning, and computer vision to language models. These findings challenge the conventional understanding that normalization layers are indispensable in modern neural networks, and offer new insights into their role in deep networks.

归一化层在现代神经网络中无处不在，长期以来被认为是必不可少的。这项工作表明，无需归一化的 Transformers 可以通过一种非常简单的技术实现相同或更好的性能。我们引入了动态 Tanh（DyT），这是一种逐元素操作 $\mathrm{DyT}(\boldsymbol{x}) = \tanh(\alpha \boldsymbol{x})$，作为 Transformers 中归一化层的替代品。DyT 的灵感来自于观察到 Transformers 中的层归一化通常会产生类似 tanh 的 S 形输入输出映射。通过引入 DyT，无需归一化的 Transformers 可以在大多数情况下无需超参数调整的情况下匹配或超越其归一化对应物的性能。我们在从识别到生成、从监督学习到自监督学习、从计算机视觉到语言模型的各种设置中验证了带有 DyT 的 Transformers 的有效性。这些发现挑战了传统观念，即归一化层在现代神经网络中是不可或缺的，并为它们在深度网络中的作用提供了新的见解。

Date: June 28, 2025
Project page and code: jiachenzhu.github.io/DyT
Correspondence: jiachen.zhu@nyu.edu, zhuangl@princeton.edu

Meta

1 Introduction

Over the past decade, normalization layers have solidified their positions as one of the most fundamental components of modern neural networks. It all traces back to the invention of batch normalization in 2015 (37), which enabled drastically faster and better convergence in visual recognition models and quickly gained momentum in the following years. Since then, many variants of normalization layers have been proposed for different network architectures or domains (5; 73; 75; 80). Today, virtually all modern networks use normalization layers, with layer normalization (Layer Norm, or LN) (5) being one of the most popular, particularly in the dominant Transformer architecture (74; 20).

在过去的十年中，归一化层已经巩固了其作为现代神经网络中最基本组件之一的地位。这一切可以追溯到 2015 年批量归一化的发明 (37)，它使得视觉识别模型的收敛速度显著加快且效果更好，并在接下来的几年中迅速获得了广泛的应用。自那时起，针对不同网络架构或领域的归一化层变体被提出 (5; 73; 75; 80)。如今，几乎所有现代网络都使用归一化层，其中层归一化（Layer Norm，或 LN） (5) 是最受欢迎的之一，尤其是在主流的 Transformer 架构中 (74; 20)。

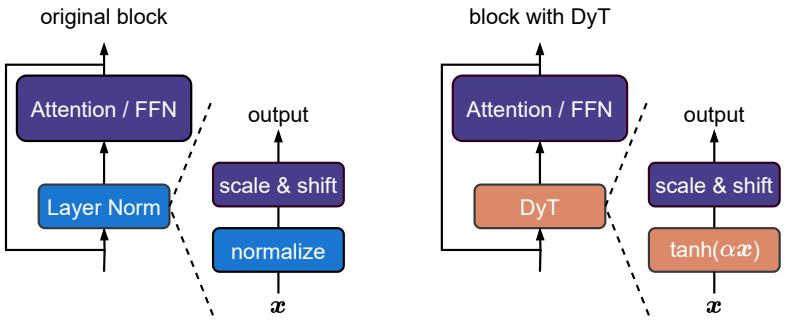

Figure 1 Left: original Transformer block. Right: block with our proposed Dynamic Tanh (DyT) layer. DyT is a straightforward replacement for commonly used Layer Norm (5) (in some cases RMSNorm (80)) layers. Transformers with DyT match or exceed the performance of their normalized counterparts.

The widespread adoption of normalization layers is largely driven by their empirical benefits in optimization (62; 10). In addition to achieving better results, they help accelerate and stabilize convergence. As neural networks become wider and deeper, this necessity becomes ever more critical (11; 36). Consequently, normalization layers are widely regarded as crucial, if not indispensable, for the effective training of deep networks. This belief is subtly evidenced by the fact that, in recent years, novel architectures often seek to replace attention or convolution layers (70; 27; 67; 22), but almost always retain the normalization layers.
归一化层的广泛采用主要归因于其在优化中的经验性优势 (62; 10)。除了获得更好的结果外，它们还有助于加速和稳定收敛。随着神经网络变得越来越宽和越来越深，这种必要性变得更加关键 (11; 36)。因此，归一化层被广泛认为是深度网络有效训练的关键，甚至不可或缺。这一信念在近年来得到了微妙的支持，因为新型架构通常试图替换注意力或卷积层 (70; 27; 67; 22)，但几乎总是保留归一化层。

This paper challenges this belief by introducing a simple alternative to normalization layers in Transformers. Our exploration starts with the observation that LN layers map their inputs to outputs with tanh-like, S-shaped curves, scaling the input activations while squashing the extreme values. Inspired by this insight, we propose an element-wise operation termed Dynamic Tanh (DyT), defined as: $\mathrm{DyT}(\boldsymbol{x}) = \tanh(\alpha\boldsymbol{x})$, where α is a learnable parameter. This operation aims to emulate the behavior of LN by learning an appropriate scaling factor through α and squashing extreme values via the bounded tanh function. Notably, unlike normalization layers, it achieves both effects without the need to compute activation statistics.
本文通过引入一种简单的替代方案来挑战这一信念。我们的探索始于观察到 LN 层将其输入映射到具有类似 tanh 的 S 形曲线的输出，缩放输入激活值并压缩极端值。受此启发，我们提出了一种称为动态 Tanh（DyT）的逐元素操作，定义为：$\mathrm{DyT}(\boldsymbol{x}) = \tanh(\alpha\boldsymbol{x})$，其中 α 是一个可学习的参数。该操作旨在通过 α 学习适当的缩放因子，并通过有界的 tanh 函数压缩极端值来模拟 LN 的行为。值得注意的是，与归一化层不同，它在不需要计算激活统计量的情况下实现了这两种效果。

Employing DyT is straightforward, as shown in Figure 1: we directly replace existing normalization layers with DyT in architectures such as vision and language Transformers. We empirically demonstrate that models with DyT can train stably and achieve high final performance across a wide range of settings. It often does not require tuning the training hyperparameters on the original architecture. Our work challenges the notion that normalization layers are indispensable for training modern neural networks and provides empirical insights into the properties of normalization layers. Moreover, preliminary measurements suggest that DyT improves training and inference speed, making it a candidate for efficiency-oriented network design.
使用 DyT 非常简单，如图 1所示：我们直接在视觉和语言 Transformer 等架构中用 DyT 替换现有的归一化层。我们通过实验证明，使用 DyT 的模型可以在各种设置下稳定训练并达到较高的最终性能。通常不需要在原始架构上调整训练超参数。我们的工作挑战了归一化层对于训练现代神经网络不可或缺的观念，并为归一化层的特性提供了经验性见解。此外，初步测量表明，DyT 提高了训练和推理速度，使其成为面向效率的网络设计的候选方案。

2 Background: Normalization Layers

We begin by reviewing the normalization layers. Most normalization layers share a common formulation. Given an input $\boldsymbol{x}$ 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, the output is generally computed as:
我们首先回顾归一化层。大多数归一化层共享一个共同的公式。给定一个形状为 (B, T, C) 的输入 $\boldsymbol{x}$，其中 B 是批量大小，T 是 token 数量，C 是每个 token 的嵌入维度，输出通常计算为：

$$\text{normalization}(\boldsymbol{x}) = \boldsymbol{\gamma} * \left(\frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \right) + \boldsymbol{\beta} \tag{1}$$

where ϵ is a small constant, and $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learnable vector parameters of shape $(C,)$. They are "scaling" and "shifting" affine parameters that allow the output to be in any range. The terms $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ denote the mean and variance of the input. Different methods mainly differ in how these two statistics are computed. This results in $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ having different dimensions, each with broadcasting applied during computation.
其中 ϵ 是一个小常数，$\boldsymbol{\gamma}$ 和 $\boldsymbol{\beta}$ 是形状为 $(C,)$ 的可学习向量参数。它们是 "缩放" 和 "平移" 仿射参数，允许输出在任何范围内。$\boldsymbol{\mu}$ 和 $\boldsymbol{\sigma}^2$ 表示输入的均值和方差。不同的方法主要在于如何计算这两个统计量。这导致 $\boldsymbol{\mu}$ 和 $\boldsymbol{\sigma}^2$ 具有不同的维度，每个维度在计算时都应用了广播。

Batch normalization (BN) (37) is the first modern normalization layer, and it has been primarily used in ConvNet models (68; 31; 76). Its introduction represents a major milestone in deep learning architecture designs. BN computes the mean and variance across both the batch and token dimensions, specifically: $\mu_k = \frac{1}{BT}\sum_{i,j} x_{ijk}$ and $\sigma_k^2 = \frac{1}{BT}\sum_{i,j}(x_{ijk} - \mu_k)^2$. Other normalization layers popular in ConvNets, such as group normalization (75) and instance normalization (73), were initially proposed for specialized tasks such as object detection and image stylization. They share the same overall formulation but differ in the axes and ranges over which the statistics are computed.
批量归一化（BN）(37) 是第一个现代归一化层，主要用于 ConvNet 模型 (68; 31; 76)。它的引入代表了深度学习架构设计的一个重要里程碑。BN 在批量和 token 维度上计算均值和方差，具体为：$\mu_k = \frac{1}{BT}\sum_{i,j} x_{ijk}$ 和 $\sigma_k^2 = \frac{1}{BT}\sum_{i,j}(x_{ijk} - \mu_k)^2$。其他在 ConvNets 中流行的归一化层，如组归一化 (75) 和实例归一化 (73)，最初是为对象检测和图像风格化等专门任务提出的。它们共享相同的整体公式，但在计算统计量的轴和范围上有所不同。

Layer normalization (LN) (5) and root mean square normalization (RMSNorm) (80) are the major two types of normalization layers used in Transformer architectures. LN computes these statistics independently for each token in each sample, where $\mu_{ij} = \frac{1}{C}\sum_k x_{ijk}$ and $\sigma_{ij}^2 = \frac{1}{C}\sum_k (x_{ijk} - \mu_{ij})^2$. RMSNorm (80) simplifies LN by removing the mean-centering step and normalizing the input with $\mu_{ij} = 0$ and $\sigma_{ij}^2 = \frac{1}{C}\sum_k x_{ijk}^2$. Today, most modern neural networks use LN due to its simplicity and universality. Recently, RMSNorm has gained popularity, particularly in language models like T5 (60), LLaMA (71; 72; 21), Mistral (38), Qwen (8; 79), InternLM (82; 13) and DeepSeek (43; 28). The Transformers we examine in this work all use LN, except that LLaMA uses RMSNorm.
层归一化（LN）(5) 和均方根归一化（RMSNorm）(80) 是 Transformer 架构中使用的两种主要归一化层。LN 为每个样本中的每个 token 独立计算这些统计量，其中 $\mu_{ij} = \frac{1}{C}\sum_k x_{ijk}$ 和 $\sigma_{ij}^2 = \frac{1}{C}\sum_k (x_{ijk} - \mu_{ij})^2$。RMSNorm (80) 通过去除均值中心化步骤并使用 $\mu_{ij} = 0$ 和 $\sigma_{ij}^2 = \frac{1}{C}\sum_k x_{ijk}^2$ 来简化 LN。如今，大多数现代神经网络由于其简单性和通用性而使用 LN。最近，RMSNorm 在语言模型中获得了广泛的应用，特别是在 T5 (60)、LLaMA (71; 72; 21)、Mistral (38)、Qwen (8; 79)、InternLM (82; 13) 和 DeepSeek (43; 28) 等模型中。我们在本文中研究的 Transformer 都使用 LN，除了 LLaMA 使用 RMSNorm。

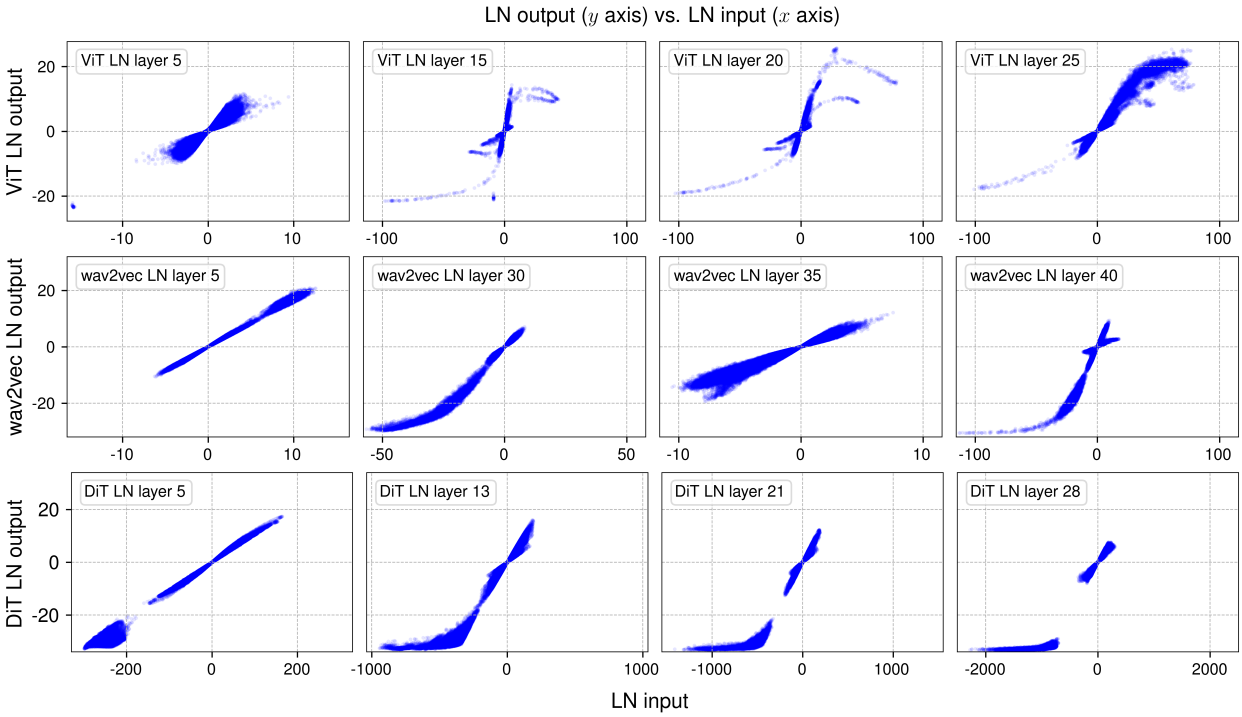

Figure 2 Output vs. input of selected layer normalization (LN) layers in Vision Transformer (ViT) (20), wav2vec 2.0 (a Transformer model for speech) (7), and Diffusion Transformer (DiT) (58). We sample a mini-batch of samples and plot the input / output values of four LN layers in each model. The outputs are before the affine transformation in LN. The S-shaped curves highly resemble that of a tanh function (see Figure 3). The more linear shapes in earlier layers can also be captured by the center part of a tanh curve. This motivates us to propose Dynamic Tanh (DyT) as a replacement, with a learnable scaler α to account for different scales on the x axis.

3 What Do Normalization Layers Do?

Analysis setup. We first empirically study the behaviors of normalization layers in trained networks. For this analysis, we take a Vision Transformer model (ViT-B) (20) trained on ImageNet-1K (19), a wav2vec 2.0 Large Transformer model (7) trained on LibriSpeech (57), and a Diffusion Transformer (DiT-XL) (58) trained on ImageNet-1K. In all cases, LN is applied in every Transformer block and before the final linear projection.

分析设置。我们首先实证研究训练网络中归一化层的行为。为此分析，我们采用在 ImageNet-1K 上训练的 Vision Transformer 模型（ViT-B）（20），在 LibriSpeech 上训练的 wav2vec 2.0 大型 Transformer 模型（7），以及在 ImageNet-1K 上训练的 Diffusion Transformer（DiT-XL）（58）。在所有情况下，LN 都应用于每个 Transformer 块中，并在最终线性投影之前。

For all three trained networks, we sample a mini-batch of samples and do a forward pass through the network. We then measure the input and output for the normalization layers, i.e., tensors immediately before and after the normalization operation, before the learnable affine transformation. Since LN preserves the dimensions of the input tensor, we can establish a one-to-one correspondence between the input and output tensor elements, allowing for a direct visualization of their relationship. We plot the resulting mappings in Figure 2.

对于所有三个训练网络，我们采样一个小批量样本并进行前向传播。然后我们测量归一化层的输入和输出，即在归一化操作之前和之后的张量，在可学习的仿射变换之前。由于 LN 保留了输入张量的维度，我们可以在输入和输出张量元素之间建立一一对应关系，从而直接可视化它们的关系。我们将结果映射绘制在图 2中。

Tanh-like mappings with layer normalization. For all three models, in earlier LN layers (1st column of Figure 2), we find this input-output relationship to be mostly linear, resembling a straight line in an x-y plot. However, the deeper LN layers are places where we make more intriguing observations.
层归一化的类 Tanh **映射**。对于所有三个模型，在较早的 LN 层中（图 2的第一列），我们发现这种输入-输出关系大多是线性的，类似于 x-y 图中的直线。然而，在更深的 LN 层中，我们观察到了一些更有趣的现象。

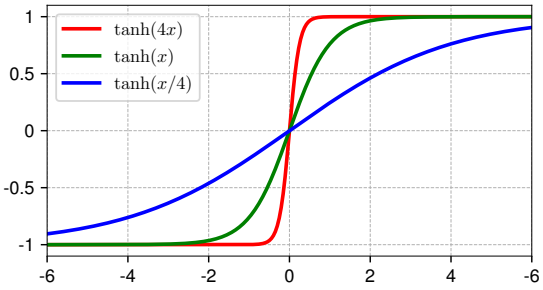

Figure 3 $\tanh(\alpha x)$ with three different α values.

A striking observation from these deeper layers is that most of these curves' shapes highly resemble full or partial S-shaped curves represented by a tanh function (see Figure 3). One might expect LN layers to linearly transform the input tensor, as subtracting the mean and dividing by standard deviation are linear operations. LN normalizes in a per-token manner, only linearly transforming each token's activations. As tokens have different mean and standard deviation values, the linearity does not hold collectively on all activations of the input tensor. Nonetheless, it is still surprising to us that the actual non-linear transformation is highly similar to a scaled tanh function.
从这些更深层次中，一个显著的观察是，这些曲线中的大多数形状高度类似于由 tanh 函数表示的完整或部分 S 形曲线（见图 3）。人们可能期望 LN 层对输入张量进行线性变换，因为减去均值并除以标准差是线性操作。LN 以每个 token 为单位进行归一化，仅对每个 token 的激活进行线性变换。由于 token 具有不同的均值和标准差值，线性性并不适用于输入张量的所有激活。尽管如此，实际的非线性变换与缩放的 tanh 函数高度相似，这仍然让我们感到惊讶。

For such an S-shaped curve, we note that the central part, represented by points with x values close to zero, is still mainly in a linear shape. Most points (∼99%) fall in this linear range. However, there are still many points that clearly fall out of this range, which are considered to have "extreme" values, e.g., those with x larger than 50 or smaller than -50 in the ViT model. Normalization layers' main effect for these values is to squash them into less extreme values, more in line with the majority of points. This is where normalization layers could not approximated by a simple affine transformation layer. We hypothesize this non-linear and disproportional squashing effect on extreme values is what makes normalization layers important and indispensable.
对于这样的 S 形曲线，我们注意到中心部分（由 x 值接近零的点表示）仍然是线性的。大多数点（∼99%）落在这个线性范围内。然而，仍然有许多点明显超出这个范围，这些点被认为是具有"极端"值，例如在 ViT 模型中 x 大于 50 或小于-50 的点。归一化层对这些值的主要效果是将它们压缩为不那么极端的值，更符合大多数点的范围。这是归一化层无法通过简单的仿射变换层近似的地方。我们假设这种对极端值的非线性和不成比例的压缩效果是归一化层重要且不可或缺的原因。

Recent findings by Ni et al. (56) similarly highlight the strong non-linearities introduced by LN layers, demonstrating how the non-linearity enhances a model' s representational capacity. Moreover, this squashing behavior mirrors the saturation properties of biological neurons for large inputs, a phenomenon first observed about a century ago (1; 2; 3).
最近的研究 (56) 同样强调了 LN 层引入的强非线性，展示了非线性如何增强模型的表示能力。此外，这种压缩行为反映了生物神经元在大输入下的饱和特性，这一现象大约在一个世纪前首次被观察到 (1; 2; 3)。

Normalization by tokens and channels. How does an LN layer perform a linear transformation for each token but also squash the extreme values in such a non-linear fashion? To understand this, we visualize the points grouped by tokens and channels, respectively. This is plotted in Figure 4 by taking the second and third subplots for ViT from Figure 2, but with a sampled subset of points for more clarity. When we select the channels to plot, we make sure to include the channels with extreme values.
通过 token **和** channel **进行归一化**。 LN 层如何对每个 token 执行线性变换，同时以这种非线性方式压缩极端值？为了理解这一点，我们分别按 token 和 channel 对点进行可视化。这在图 4中通过从图 2中提取 ViT 的第二和第三子图绘制，但为了更清晰起见，使用了采样的点子集。当我们选择要绘制的 channel 时，我们确保包含具有极端值的 channel。

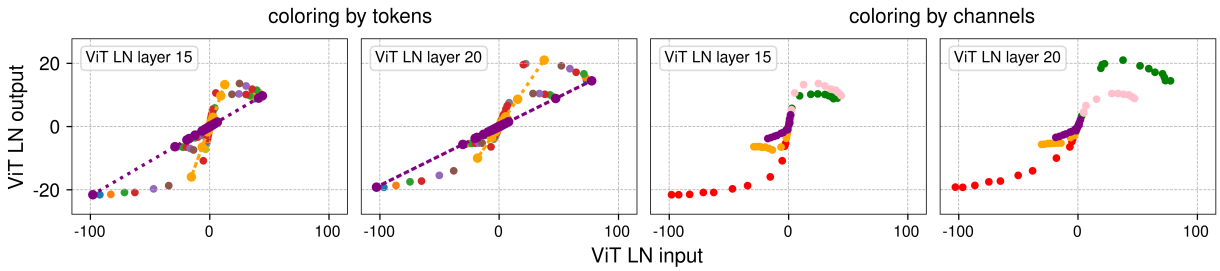

Figure 4 Output vs. input of two LN layers, with tensor elements colored to indicate different channel and token dimensions. The input tensor has a shape of (samples, tokens, and channels), with elements visualized by assigning consistent colors to the same tokens (left two panels) and channels (right 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.

两个 LN **层的输出与输入对比，张量元素着色以表示不同的** channel **和** token **维度**。输入张量的形状为（样本，token，channel），通过为相同的 token（左侧两图）和 channel（右侧两图）分配一致的颜色来可视化元素。*左侧两图*：表示相同 token 的点（相同颜色）在不同 channel 之间形成直线，因为 LN 在每个 token 的 channel 上线性操作。有趣的是，当集体绘制时，这些线形成了一条非线性的 tanh 形曲线。*右侧两图*：每个 channel 的输入在 x 轴上跨越不同的范围，为整体的 tanh 形曲线贡献了不同的部分。某些 channel（例如红色、绿色和粉色）表现出更极端的 x 值，这些值被 LN 压缩。

On the left two panels of Figure 4, we visualize each token's activations using the same color. We observe that all points from any single token do form a straight line. However, since each token has a different variance, the slopes are different. Tokens with smaller input x ranges tend to have smaller variance, and the normalization layer will divide their activations using a smaller standard deviation, hence producing a larger slope in the straight line. Collectively, they form an S-shaped curve that resembles a tanh function. In the two panels on the right, we color each channel's activations using the same color. We find that different channels tend to have drastically different input ranges, with only a few channels (e.g., red, green, and pink) exhibiting large extreme values. These are the channels that get squashed the most by the normalization layer.

在图 4的左侧两图中，我们使用相同的颜色可视化每个 token 的激活值。我们观察到，来自任何单个 token 的所有点确实形成了一条直线。然而，由于每个 token 的方差不同，斜率也不同。输入 x 范围较小的 token 往往具有较小的方差，归一化层将使用较小的标准差来划分它们的激活值，因此在直线中产生较大的斜率。总体而言，它们形成了一条类似于 tanh 函数的 S 形曲线。在右侧的两图中，我们使用相同的颜色为每个 channel 的激活值着色。我们发现不同的 channel 往往具有截然不同的输入范围，只有少数 channel（例如红色、绿色和粉色）表现出较大的极端值。这些是被归一化层压缩最多的 channel。

4 Dynamic Tanh (DyT)

Inspired by the similarity between the shapes of normalization layers and a scaled tanh function, we propose Dynamic Tanh (DyT) as a drop-in replacement for normalization layers. Given an input tensor $\boldsymbol{x}$, a DyT layer is defined as follows

受到归一化层形状与缩放 tanh 函数相似性的启发，我们提出了动态 Tanh（DyT）作为归一化层的替代方案。给定输入张量 $\boldsymbol{x}$，DyT 层定义如下：

$$\mathrm{DyT}(\boldsymbol{x}) = \boldsymbol{\gamma} * \tanh(\alpha \boldsymbol{x}) + \boldsymbol{\beta} \tag{2}$$

where α is a learnable scalar parameter that allows scaling the input differently based on its range, accounting for varying x scales (Figure 2). This is also why we name the whole operation "Dynamic" Tanh. $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learnable, per-channel vector parameters, the same as those used in all normalization layers—they allow the output to scale back to any scales. This is sometimes considered a separate affine layer; for our purposes, we consider them to be part of the DyT layer, just like how normalization layers also include them. See Algorithm 1 for implementation of DyT in Pytorch-like pseudocode.
其中 α 是一个可学习的标量参数，允许根据输入范围对其进行不同的缩放，考虑到不同的 x 尺度（图 2）。这也是我们将整个操作命名为 "动态" Tanh 的原因。$\boldsymbol{\gamma}$ 和 $\boldsymbol{\beta}$ 是可学习的、每个 channel 的向量参数，与所有归一化层中使用的参数相同——它们允许输出缩放到任何尺度。这有时被认为是一个单独的仿射层；出于我们的目的，我们将它们视为 DyT 层的一部分，就像归一化层也包含它们一样。有关 DyT 在类似 Pytorch 的伪代码中的实现，请参见算法 1。

Integrating DyT layers into an existing architecture is straightforward: one DyT layer replaces one normalization layer (see Figure 1). This applies to normalization layers within attention blocks, FFN blocks, and the final normalization layer. Although DyT may look like or be considered an activation function, this study only uses it to replace normalization layers without altering any parts of the activation functions in the original architectures, such as GELU or ReLU. Other parts of the networks also remain intact. We also observe that there is little need to tune the hyperparameters used by the original architectures for DyT to perform well.
将 DyT 层集成到现有架构中非常简单：一个 DyT 层替换一个归一化层（见图 1）。这适用于注意力块、FFN 块和最终归一化层中的归一化层。尽管 DyT 可能看起来或被认为是一个激活函数，但本研究仅使用它来替换归一化层，而不改变原始架构中的任何激活函数部分，例如 GELU 或 ReLU。网络的其他部分也保持不变。我们还观察到，DyT 在表现良好时几乎不需要调整原始架构中使用的超参数。

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
```

On scaling parameters. We always simply initialize $\boldsymbol{\gamma}$ to an all-one vector and $\boldsymbol{\beta}$ to an all-zero vector following normalization layers. For the scaler parameter α, a default initialization of 0.5 is generally sufficient, except for LLM training. A detailed analysis of α initialization is provided in Section ??. Unless explicitly stated otherwise, α is initialized to 0.5 in our subsequent experiments.
关于缩放参数。我们总是简单地将 $\boldsymbol{\gamma}$ 初始化为全 1 向量，将 $\boldsymbol{\beta}$ 初始化为全 0 向量，遵循归一化层的做法。对于缩放参数 α，默认初始化为 0.5 通常足够，除了 LLM 训练。关于 α 初始化的详细分析在章节??中提供。除非另有说明，否则在我们的后续实验中，α 初始化为 0.5。

Remarks. DyT is not a new type of normalization layer, as it operates on each input element from a tensor independently during a forward pass without computing statistics or other types of aggregations. It does, however, preserve 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.
备注。DyT 并不是一种新的归一化层，因为它在前向传播期间独立地对张量中的每个输入元素进行操作，而不计算统计量或其他类型的聚合。然而，它确实保留了归一化层以非线性方式压缩极端值的效果，同时几乎线性地转换输入的中央部分。

5 Experiments

To demonstrate the effectiveness of DyT, we experiment with Transformers and a few other modern architectures across a diverse range of tasks and domains. In each experiment, we replace the LN or RMSNorm in the original architectures with DyT layers and follow the official open-source protocols to train and test both versions of the models. Detailed instructions for reproducing our results are provided in Appendix A. Notably, to highlight the simplicity of adapting DyT, we use hyperparameters identical to those utilized by the normalized counterparts. For completeness, additional experimental results regarding tuning of learning rates and initial values of α are provided in Appendix ??.
为了证明 DyT 的有效性，我们在 Transformers 和其他一些现代架构上进行了多种任务和领域的实验。在每个实验中，我们将原始架构中的 LN 或 RMSNorm 替换为 DyT 层，并遵循官方的开源协议来训练和测试两个版本的模型。详细的复现说明在附录A中提供。值得注意的是，为了突出 DyT 的简单适应性，我们使用了与归一化对应模型相同的超参数。为了完整性，关于学习率和 α 初始值调整的额外实验结果在附录??中提供。

Supervised learning in vision. We train Vision Transformer (ViT) (20) and ConvNeXt (44) of "Base" and "Large" sizes on the ImageNet-1K classification task (19). These models are selected due to their popularity and distinct operations: attention in ViT and convolution in ConvNeXt. Table 1 reports the top-1 classification accuracies. DyT performs slightly better than LN across both architectures and model sizes. We further plot the training loss for ViT-B and ConvNeXt-B in Figure 6. The curves show that the convergence behaviors of DyT and LN-based models are highly aligned.
视觉中的监督学习。我们在 ImageNet-1K 分类任务上训练了 "Base" 和 "Large" 大小的 Vision Transformer (ViT) (20) 和 ConvNeXt (44)。选择这些模型是因为它们的流行性和不同的操作：ViT 中的注意力和 ConvNeXt 中的卷积。表 1报告了 top-1 分类准确率。DyT 在两种架构和模型大小上都略优于 LN。我们进一步在图 6中绘制了 ViT-B 和 ConvNeXt-B 的训练损失曲线。曲线显示，DyT 和基于 LN 的模型的收敛行为高度一致。

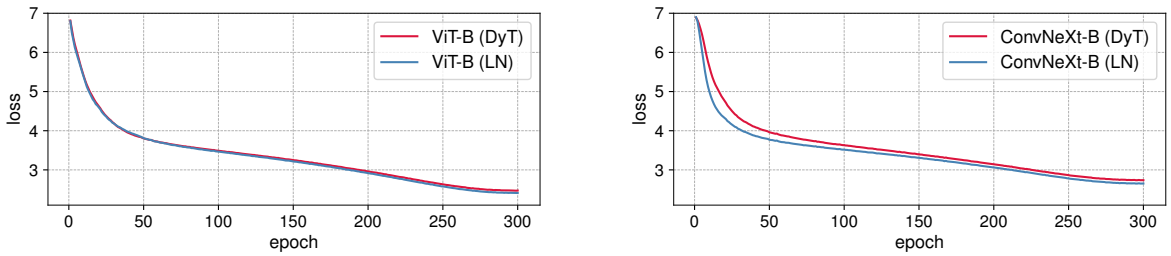

Figure 5 Training loss curves for ViT-B and ConvNeXt-B models. The loss curves for both model types exhibit similar patterns between LN and DyT, suggesting that LN and DyT may share similar learning dynamics.

Self-supervised learning in vision. We benchmark with two popular visual self-supervised learning methods: masked autoencoders (MAE) (32) and DINO (14). Both by default use Vision Transformers as the backbones, but have different training objectives: MAE is trained with a reconstruction loss, and DINO uses a joint-embedding loss (41). Following the standard self-supervised learning protocol, we first pretrain models on ImageNet-1K without using any labels and then test the pretrained models by attaching a classification layer and fine-tuning them with labels. The fine-tuning results are presented in Table 2. DyT consistently performs on par with LN in self-supervised learning tasks.
视觉中的自监督学习。我们使用两种流行的视觉自监督学习方法进行基准测试：掩码自编码器（MAE）(32) 和 DINO (14)。两者默认使用 Vision Transformers 作为骨干网络，但具有不同的训练目标：MAE 使用重建损失进行训练，而 DINO 使用联合嵌入损失 (41)。遵循标准的自监督学习协议，我们首先在 ImageNet-1K 上预训练模型而不使用任何标签，然后通过附加分类层并使用标签进行微调来测试预训练模型。微调结果如表 2所示。DyT 在自监督学习任务中始终与 LN 表现相当。

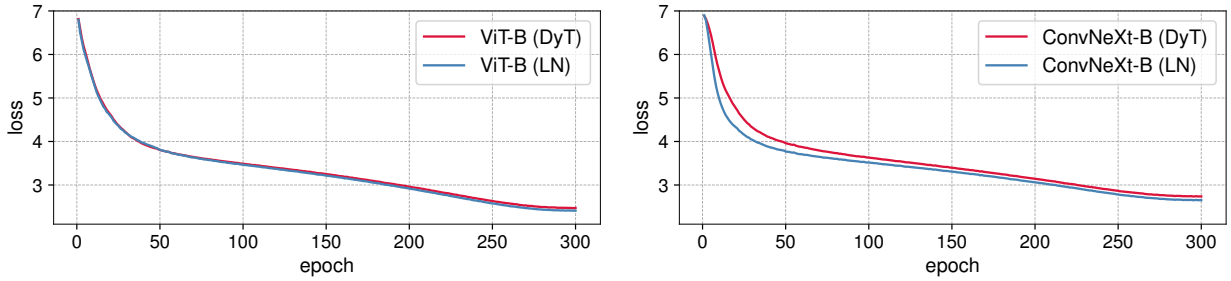

Figure 6 ViT-B **和** ConvNeXt-B **模型的训练损失曲线**。两种模型类型的损失曲线在 LN 和 DyT 之间表现出相似的模式，表明 LN 和 DyT 可能具有相似的学习动态。

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%

Table 1 Supervised classification accuracy on ImageNet-1K. DyT achieves better or similar performance than LN across both architectures and model sizes.
ImageNet-1K **上的监督分类准确率**。 DyT 在两种架构和模型大小上都取得了优于或类似于 LN 的表现。

Diffusion models. We train three Diffusion Transformer (DiT) models (58) of sizes B, L and XL on ImageNet-1K (19). The patch size is 4, 4, and 2, respectively. Note that in DiT, the LN layers' affine parameters are used for class conditioning in DiT, and we keep them that way in our DyT experiments, only replacing the normalizing transformation with the $\tanh(\alpha \boldsymbol{x})$ function. After training, we evaluate the Fréchet Inception Distance (FID) scores using the standard ImageNet "reference batch", as presented in Table 3. DyT achieves comparable or improved FID over LN.
扩散模型。我们在 ImageNet-1K 上训练了三种大小的扩散 Transformer（DiT）模型 (58)，分别为 B、L 和 XL。patch 大小分别为 4、4 和 2。请注意，在 DiT 中，LN 层的仿射参数用于 DiT 中的类别条件化，我们在 DyT 实验中保持这种方式，仅将归一化变换替换为 $\tanh(\alpha \boldsymbol{x})$ 函数。训练后，我们使用标准的 ImageNet "参考批次" 评估 Fréchet Inception Distance（FID）分数，如表 3所示。DyT 在 FID 上取得了与 LN 相当或更好的成绩。

Large Language Models. We pretrain LLaMA 7B, 13B, 34B, and 70B models (71; 72; 21) to assess DyT performance relative to RMSNorm (80), the default normalization layer used in LLaMA. The models are trained on The Pile dataset (25) with 200B tokens, following the original recipe outlined in LLaMA (72). On LLaMA with DyT, we add a learnable scalar parameter after the initial embedding layer, and adjust the initial value of α, as detailed in Section ??. We report the loss value after training and also follow OpenLLaMA (26) to benchmark the models on 15 zero-shot tasks from lm-eval (24). As shown in Table 4, DyT performs on par with RMSNorm across all four model sizes. Figure 7 illustrates the loss curves, demonstrating similar trends across all model sizes, with training losses closely aligned throughout training.
大型语言模型。我们预训练了 LLaMA 7B、13B、34B 和 70B 模型 (71; 72; 21)，以评估 DyT 相对于 LLaMA 中使用的默认归一化层 RMSNorm (80) 的性能。这些模型在 The Pile 数据集 (25) 上训练，使用了 200B 的 token，遵循 LLaMA (72) 中概述的原始配方。在使用 DyT 的 LLaMA 中，我们在初始嵌入层后添加了一个可学习的标量参数，并调整了 α 的初始值，如第 ??节所述。我们报告了训练后的损失值，并遵循 OpenLLaMA (26) 在 15 个零样本任务上对模型进行基准测试，这些任务来自 lm-eval (24)。如表 4所示，DyT 在所有四种模型大小上与 RMSNorm 表现相当。图 7展示了损失曲线，表明所有模型大小的趋势相似，训练损失在整个训练过程中紧密对齐。

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%

Table 2 Self-supervised learning accuracy on ImageNet-1K. DyT performs on par with LN across different pretraining methods and model sizes in self-supervised learning tasks.
ImageNet-1K **上的自监督学习准确率**。 DyT 在不同的预训练方法和模型大小上在自监督学习任务中与 LN 表现相当。

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

Table 3 Image generation quality (FID, lower is better) on ImageNet. DyT achieves comparable or superior FID scores to LN across various
ImageNet **上的图像生成质量（FID，越低越好）**。 DyT 在各种模型大小上取得了与 LN 相当或更好的 FID 分数。

Both analyses suggest that α functions partially as a normalization mechanism by learning values approximating 1/std of the input activations. Unlike LN, which normalizes the activations per token, α normalizes the entire input activations collectively. Consequently, α alone cannot suppress extreme values in a non-linear fashion.
两项分析都表明，α 通过学习近似输入激活的 1/std 的值，部分起到了归一化机制的作用。与 LN（逐 token 归一化激活）不同，α 对整个输入激活进行集体归一化。因此，α 本身无法以非线性的方式抑制极端值。

As can be seen in Table 5, 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.
从表 5中可以看出，网络越宽，对 "attention" 和 "other" 的初始化越不均匀。我们假设 LLM 的 α 初始化的敏感性与其相对于其他模型过大的宽度有关。

6 Related Work

Mechanisms of Normalization layers. There has been a rich line of work investigating normalization layers' role in enhancing model performance through various mechanisms. These include stabilizing gradient flow during training (9; 17; 46), reducing sensitivity to weight initialization (81; 18; 64), moderating outlier eigenvalues (10; 39), auto-tuning learning rates (4; 69), and smoothing the loss landscape for more stable optimization (62). These earlier works focused on studying batch normalization. Recent studies (47; 16; 53) further highlight the connection between normalization layers and sharpness reduction, which contributes to better generalization.
归一化层的机制。有许多研究工作探讨了归一化层通过多种机制提升模型性能的作用。这些机制包括在训练过程中稳定梯度流 (9; 17; 46)，减少对权重初始化的敏感性 (81; 18; 64)，调节异常特征值 (10; 39)，自动调整学习率 (4; 69)，以及平滑损失景观以实现更稳定的优化 (62)。这些早期工作主要集中在研究批量归一化。最近的研究 (47; 16; 53) 进一步强调了归一化层与锐度降低之间的联系，这有助于更好的泛化。

Normalization in Transformers. With the rise of Transformer (74), research has increasingly focused on layer normalization (5), which has proven particularly effective for sequential data in natural language tasks (55; 78; 77). Recent work (56) reveals that layer normalization introduces strong non-linearity, enhancing the model's representational capacity. Additionally, studies (45; 42) demonstrate that modifying the location of normalization layers within Transformers can improve convergence properties.
Transformer **中的归一化**。随着 Transformer 的兴起 (74)，研究越来越关注层归一化 (5)，它已被证明在自然语言任务中对序列数据特别有效 (55; 78; 77)。最近的工作 (56) 揭示了层归一化引入了强非线性，增强了模型的表示能力。此外，研究 (45; 42) 表明，修改 Transformer 中归一化层的位置可以改善收敛特性。

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	- / -

Table 4 Language models' training loss and average performance with 15 zero-shot lm-eval tasks. DyT achieves a comparable zero-shot performance and training loss to RMSNorm.

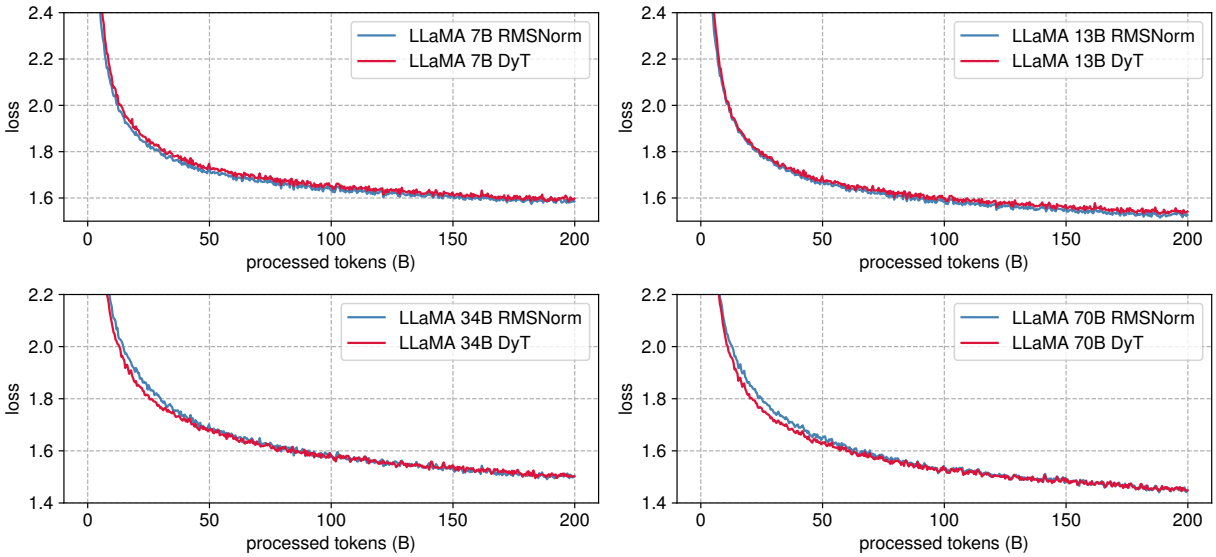

Figure 7 LLaMA pretraining loss. The loss curves of DyT and RMSNorm models are closely aligned across model sizes.

Removing normalization. Many studies have explored how to train deep models without normalization layers. Several works (81; 18; 6) explore alternative weight initialization schemes to stabilize training. The pioneering work by Brock et al. (11, 12) show that high-performing ResNets can be trained without normalization (65) through combination of initialization techniques (18), weight normalization (61; 35; 59), and adaptive gradient clipping (12). Additionally, their training strategy incorporates extensive data augmentation (15) and regularization (66; 34). The studies above are based on various ConvNet models.

去除归一化。许多研究探讨了如何在没有归一化层的情况下训练深度模型。一些工作 (81; 18; 6) 探索了替代的权重初始化方案以稳定训练。Brock et al. (11, 12) 的开创性工作表明，高性能的 ResNet 可以通过结合初始化技术 (18)、权重归一化 (61; 35; 59) 和自适应梯度裁剪 (12) 来在没有归一化的情况下进行训练 (65)。此外，他们的训练策略还结合了广泛的数据增强 (15) 和正则化 (66; 34)。上述研究基于各种 ConvNet 模型。

In Transformer architectures, He and Hofmann (30) explore modifications to Transformer blocks that reduce reliance on normalization layers and skip connections. Alternatively, Heimersheim (33) propose a method to gradually remove LN from pretrained networks by fine-tuning the model after removing each normalization layer. Unlike previous approaches, DyT requires minimal modifications to both the architecture and the training recipe. Despite its simplicity, DyT achieves stable training and comparable performance.

在 Transformer 架构中，He and Hofmann (30) 探索了对 Transformer 块的修改，以减少对归一化层和跳跃连接的依赖。或者，Heimersheim (33) 提出了一种通过微调模型逐步从预训练网络中移除 LN 的方法。与之前的方法不同，DyT 对架构和训练方案的修改都很少。尽管简单，DyT 实现了稳定的训练和可比的性能。

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 5 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.

7 Limitations

We conduct experiments on networks using either LN or RMSNorm because of their popularity in Transformers and other modern architectures. Preliminary experiments (see Appendix ??) indicate that DyT struggles to replace BN directly in classic networks like ResNets. It remains to be studied in more depth whether and how DyT can adapt to models with other types of normalization layers.
我们在使用 LN 或 RMSNorm 的网络上进行实验，因为它们在 Transformer 和其他现代架构中很流行。初步实验（见附录 ??）表明，DyT 在经典网络如 ResNet 中难以直接替代 BN。DyT 是否以及如何适应其他类型的归一化层模型仍有待深入研究。

8 Conclusion

In this work, we demonstrate modern neural networks, in particular Transformers, can be trained without normalization layers. This is done through Dynamic Tanh (DyT), a simple replacement for traditional normalization layers. It adjusts the input activation range via a learnable scaling factor α and then squashes the extreme values through an S-shaped tanh function. Although a simpler function, it effectively captures the behavior of normalization layers. Under various settings, models with DyT match or exceed the performance of their normalized counterparts. The findings challenge the conventional understanding of the necessity of normalization layers in training modern neural networks. Our study also contributes to understanding the mechanisms of normalization layers, one of the most fundamental building blocks in deep neural networks.
在这项工作中，我们证明了现代神经网络，特别是 Transformer，可以在没有归一化层的情况下进行训练。这是通过动态 Tanh（DyT）实现的，它是传统归一化层的一个简单替代品。它通过可学习的缩放因子 α 调整输入激活范围，然后通过 S 形 tanh 函数压缩极端值。尽管是一个更简单的函数，但它有效地捕捉了归一化层的行为。在各种设置下，使用 DyT 的模型匹配或超过了其归一化对应物的性能。这些发现挑战了传统上对训练现代神经网络中归一化层必要性的理解。我们的研究也有助于理解归一化层的机制，这是深度神经网络中最基本的构建块之一。

References

[1] Edgar D Adrian. The impulses produced by sensory nerve endings: Part 1. The Journal of Physiology, 1926.

[2] Edgar D Adrian and Yngve Zotterman. The impulses produced by sensory nerve-endings: Part 2. the response of a single end-organ. The Journal of Physiology, 1926.

[3] Edgar D Adrian and Yngve Zotterman. The impulses produced by sensory nerve endings: Part 3. impulses set up by touch and pressure. The Journal of Physiology, 1926.

[4] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. arXiv preprint arXiv:1812.03981, 2018.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[6] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In UAI, 2021.

[7] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. NeurIPS, 2020.

[8] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

[9] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In ICML, 2017.

[10] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. NeurIPS, 2018.

[11] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. arXiv preprint arXiv:2101.08692, 2021.

[12] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In ICML, 2021.

[13] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. arXiv preprint arXiv:2403.17297, 2024.

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In ICCV, 2021.

[15] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In CVPR Workshops, 2020.

[16] Yan Dai, Kwangjun Ahn, and Suvrit Sra. The crucial role of normalization in sharpness-aware minimization. NeurIPS, 2024.

[17] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. NeurIPS, 2020.

[18] Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. NeurIPS, 2020.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[21] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

[22] Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadegh. Were rnns all we needed? arXiv preprint arXiv:2410.01201, 2024.

[23] Foundation Model Stack. Github: FMS FSDP. https://github.com/foundation-model-stack/fms-fsdp. Accessed: 2025-01-23.

[24] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. https://zenodo.org/records/10256836.

[25] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. arXiv preprint arXiv:2101.00027, 2020.

[26] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, 2023. https://github.com/openlm-research/open_llama.

[27] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

[28] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

[29] HazyResearch. Github: Hyenadna. https://github.com/HazyResearch/hyena-dna.git. Accessed: 2025-01-23.

[30] Bobby He and Thomas Hofmann. Simplifying transformer blocks. arXiv preprint arXiv:2311.01906, 2023.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[32] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In CVPR, 2022.

[33] Stefan Heimersheim. You can remove gpt2's layernorm by fine-tuning. arXiv preprint arXiv:2409.13710, 2024.

[34] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In ECCV, 2016.

[35] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In ICCV, 2017.

[36] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. TPAMI, 2023.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[38] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

[39] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. NeurIPS, 2019.

[40] Kuleshov Group. Github: Caduceus. https://github.com/kuleshov-group/caduceus.git. Accessed: 2025-01-23.

[41] Yann LeCun. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27. Open Review, 2022.

[42] Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-ln: Unleashing the power of deeper layers by combining pre-ln and post-ln. arXiv preprint arXiv:2412.13795, 2024.

[43] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434, 2024.

[44] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In CVPR, 2022.

[45] Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere. arXiv preprint arXiv:2410.01131, 2024.

[46] Ekdeep S Lubana, Robert Dick, and Hidenori Tanaka. Beyond batchnorm: Towards a unified understanding of normalization in deep learning. NeurIPS, 2021.

[47] Kaifeng Lyu, Zhiyuan Li, and Sanjeev Arora. Understanding the generalization benefit of normalization layers: Sharpness reduction. NeurIPS, 2022.

[48] Meta Research. Github: ConvNeXt. https://github.com/facebookresearch/ConvNeXt, . Accessed: 2025-01-23.

[49] Meta Research. Github: DINO. https://github.com/facebookresearch/dino, . Accessed: 2025-01-23.

[50] Meta Research. Github: DiT. https://github.com/facebookresearch/DiT, . Accessed: 2025-01-23.

[51] Meta Research. Github: MAE. https://github.com/facebookresearch/mae, . Accessed: 2025-01-23.

[52] Meta Research. Github: wav2vec 2.0. https://github.com/facebookresearch/fairseq, . Accessed: 2025-01-23.

[53] Maximilian Mueller, Tiffany Vlaar, David Rolnick, and Matthias Hein. Normalization layers are all that sharpness-aware minimization needs. NeurIPS, 2024.

[54] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. NeurIPS, 2024.

[55] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. arXiv preprint arXiv:1910.05895, 2019.

[56] Yunhao Ni, Yuxin Guo, Junlong Jia, and Lei Huang. On the nonlinearity of layer normalization. arXiv preprint arXiv:2406.01255, 2024.

[57] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In ICASSP, 2015.

[58] William Peebles and Saining Xie. Scalable diffusion models with transformers. In ICCV, 2023.

[59] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. arXiv preprint arXiv:1903.10520, 2019.

[60] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR, 2020.

[61] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. NeurIPS, 2016.

[62] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? NeurIPS, 2018.

[63] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. arXiv preprint arXiv:2403.03234, 2024.

[64] Jie Shao, Kai Hu, Changhu Wang, Xiangyang Xue, and Bhiksha Raj. Is normalization indispensable for training deep neural network? NeurIPS, 2020.

[65] Samuel L Smith, Andrew Brock, Leonard Berrada, and Soham De. Convnets match vision transformers at scale. arXiv preprint arXiv:2310.16764, 2023.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 2014.

[67] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. arXiv preprint arXiv:2407.04620, 2024.

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016.

[69] Hidenori Tanaka and Daniel Kunin. Noether' s learning dynamics: Role of symmetry breaking in neural networks. NeurIPS, 2021.

[70] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. NeurIPS, 2021.

[71] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[72] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

[73] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. NeurIPS, 2017.

[75] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018.

[76] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In CVPR, 2017.

[77] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In ICML, 2020.

[78] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. NeurIPS, 2019.

[79] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2024.

[80] Biao Zhang and Rico Sennrich. Root mean square layer normalization. NeurIPS, 2019.

[81] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. arXiv preprint arXiv:1901.09321, 2019.

[82] Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, et al. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. arXiv preprint arXiv:2407.03320, 2024.

Appendix

A Experimental Settings

Supervised image classification. For all supervised classification experiments on ImageNet-1K, we follow the training recipes from ConvNeXt (48). 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 stochastic depth rates are set to 0.1 for ViT-B and 0.4 for ViT-L.
有监督图像分类。对于所有在 ImageNet-1K 上进行的有监督分类实验，我们遵循 ConvNeXt (48) 的训练方案。对于 ConvNeXt-B 和 ConvNeXt-L，我们使用原始的超参数而不进行修改。ViT-B 和 ViT-L 模型使用与 ConvNeXt-B 相同的超参数，除了 ViT-L 的 AdamW 的 beta 参数设置为 (0.9, 0.95)，并且 ViT-B 的随机深度率设置为 0.1，ViT-L 的随机深度率设置为 0.4。

Diffusion models. We use the official implementation (50) 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.
扩散模型。我们使用官方实现 (50) 来训练所有 DiT 模型。我们发现默认的学习率对于本文考虑的模型来说并不是最优的。为了解决这个问题，我们对 LN 模型进行了简单的学习率搜索，并将调整后的学习率直接应用于 DyT 模型。我们还观察到零初始化对 DyT 模型的性能有负面影响。因此，我们保留了 LN 模型的零初始化，但移除了 DyT 模型的零初始化。

Large Language Models. In our implementation of LLaMA models (71; 72; 21) 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 (74) design, which uses a fixed scalar of the same value at the same position.
大语言模型。在我们使用 DyT 实现 LLaMA 模型 (71; 72; 21) 时，我们在嵌入层之后、任何 Transformer 块之前引入了一个额外的可学习标量参数。我们将其初始化为模型嵌入维度的平方根 $\sqrt{d}$。如果没有这个缩放标量，我们发现训练开始时模型激活的幅度太小，训练难以进行。通过引入可学习标量，问题得到了缓解，模型可以正常收敛。这个标量的添加类似于原始 Transformer (74) 的设计，后者在相同位置使用了相同值的固定标量。

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 (72). 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.
我们在 Pile 数据集 (25) 上训练所有 LLaMA 模型。我们使用 FMS-FSDP (23) 的代码库，它为 7B 模型提供了一个默认的训练方案，该方案紧密遵循 LLaMA 2 论文 (72)。我们将学习率保持在默认的 3e-4 对于 7B 和 13B 模型，1.5e-4 对于 34B 和 70B 模型，与 LLaMA 2 保持一致。批量大小设置为 4M 个 token，每个模型总共训练 200B 个 token。

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.
为了评估，我们在 lm-eval (24) 中的 15 个零样本常识推理任务上测试预训练模型：anli_r1, anli_r2, anli_r3, arc_challenge, arc_easy, boolq, hellaswag, openbookqa, piqa, record, rte, truthfulqa_mc1, truthfulqa_mc2, wic, 和 winogrande。选择紧密遵循 OpenLLaMA (26)。我们报告所有任务的平均性能。

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 (52) without modifying hyperparameters for both the Base and Large models. We report the final validation loss.
语音中的自监督学习。对于两个 wav2vec 2.0 模型，我们保留了原始架构中的第一个组归一化层，因为它主要作为数据归一化来处理未归一化的输入数据。我们使用官方实现 (52)，不修改 Base 和 Large 模型的超参数。我们报告最终的验证损失。

Other tasks. For all other tasks, MAE (32), DINO (14), HyenaDNA (54) and Caduceus (63), we directly use the publicly released code (51; 49; 29; 40), without hyperparameter tuning, for both models with LN and DyT.
其他任务。对于所有其他任务，MAE (32), DINO (14), HyenaDNA (54) 和 Caduceus (63)，我们直接使用公开发布的代码 (51; 49; 29; 40)，不进行超参数调优，适用于 LN 和 DyT 模型。

B Hyperparameters

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

Table 6 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.
使用 BN 和 DyT 的 ImageNet-1K 分类准确率。在 ResNet-50 和 VGG19 中用 DyT 替换 BN 会导致性能下降，表明 DyT 无法完全替代这些架构中的 BN

The results are summarized in Table 6. 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.
结果总结在表 6中。用 DyT 替换 BN 导致两个模型的分类准确率显著下降。这些发现表明，DyT 在这些经典的卷积网络中难以完全替代 BN。我们推测这可能与 BN 层在这些卷积网络中出现频率更高有关，它们在每个权重层中出现一次，而在 Transformer 中，LN 每几个权重层才出现一次。