PARALLEL LAYER NORMALIZATION FOR UNIVERSAL APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Layer Normalization (LN) is not only an optimization component: between affine layers it also supplies nonlinearity, and grouping changes how this nonlinearity scales with width. We study this mechanism through universal approximation. In Parallel Layer Normalization networks (PLN-Nets), fixed-dimensional LN units are applied independently to groups between two affine layers. We prove that a shallow LN-Net with one LN unit is non-universal regardless of width, whereas PLN-Nets become universal as the number of parallel groups grows. The mechanism extends to deep networks, Root Mean Square Normalization (RMSNorm) variants, position-wise feed-forward networks, and convolutional networks under suitable grouping structures. Controlled MLP and physics-informed neural network (PINN) experiments support PLN's practical viability, while CNN and ViT experiments show that group count shapes the finite-sample fitting envelope but does not act alone: the grouped axes, replacement site, and label structure lead to distinct fitting and generalization behavior.

1 INTRODUCTION

Normalization has become a fundamental component of modern neural networks. Batch Normalization (BN; Ioffe & Szegedy, 2015) was originally introduced to stabilize and accelerate the training of deep networks, while Layer Normalization (LN; Ba et al., 2016) was developed to avoid dependence on batch statistics. LN and its variants are now standard components of Transformers (Vaswani et al., 2017), and group-based normalization operators are widely used in convolutional and structured architectures (Wu & He, 2018). Most existing studies explain normalization through its effects on optimization, including scale invariance, conditioning, and training stability. Its contribution to the representation capacity of neural networks, however, remains substantially less understood.

A recent study (Ni et al., 2024) initiated this representation perspective by treating LN as a multivariate nonlinearity. It established finite-sample classification results for deep LN-Nets and showed that grouping can amplify LN's nonlinearity. Finite-sample fitting and stronger nonlinearity, however, do not establish universal approximation. It remains unclear whether increasing the number of groups, rather than merely widening one LN unit, can enable LN-Nets to approximate arbitrary continuous functions.

We answer this question first in multilayer perceptrons (MLPs; Cybenko, 1989; Hornik, 1991). We define Parallel Layer Normalization (PLN; Definition 3.2) as a direct sum of fixed-dimensional LN units placed between affine layers. A shallow network with one LN unit is non-universal regardless of its width, whereas shallow PLN-Nets become universal as the number of groups grows (Theorems 4.1–4.3). The same construction yields deep-network approximation results with explicit bounds. An exact function-class equivalence with Parallel Layer Scaling (PLS) extends these results to Root Mean Square Normalization (RMSNorm)-based networks (Zhang & Sennrich, 2019).

Convolutional neural networks (CNNs; LeCun et al., 2015) and Transformers (Vaswani et al., 2017) are more complex than MLPs, yet retain the affine–nonlinear building blocks of MLPs. We prove exact representation results for convolutional networks and position-wise feed-forward networks: in both cases, affine layers share parameters across positions while PLN acts independently on feature or channel coordinates. Token and spatial dimensions also admit grouping domains that span multiple positions, beyond the position-wise PLN structure covered by our theory. We therefore investigate these other grouping schemes empirically.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We conduct regression experiments on MLPs to empirically demonstrate the approximation capability of finite PLN-Nets, consistent with our universal-approximation result. We also use PINN experiments to explore their potential for physics-informed learning. We further investigate grouped normalization under different grouping schemes in CNNs and Transformers, finding that its performance depends on the grouping scheme and data structure. The detailed experimental settings and results are presented in Section 6.

2 Related Work

2.1 Universal Approximation and Neural Network Expressivity

The universal approximation property of neural networks has been studied for more than three decades. Early results established that single-hidden-layer networks with sigmoidal activations are universal approximators (Cybenko, 1989), and were subsequently generalized to broad classes of bounded nonconstant activations (Hornik, 1991). Later work characterized approximation rates (Barron, 1993) and examined architectural constraints including arbitrary depth (Gripenberg, 2003), bounded depth and width (Maiorov & Pinkus, 1999), and minimal width (Park et al., 2020; Shin et al., 2025). Expressivity has also been studied through linear regions (Montufar et al., 2014) and VC dimension (Bartlett et al., 2019). Universal approximation analyses have further been extended from MLPs to CNNs (Oono & Suzuki, 2023), RNNs (Li et al., 2022), and Transformers (Yun et al., 2020).

Many constructive UAT proofs exploit properties specific to the activation. For example, the piecewise-linear structure of ReLU permits localized constructions and explicit approximation rates (Yarotsky, 2017; Yun et al., 2020), whereas smooth activations such as tanh require different error-control arguments (De Ryck et al., 2021). Normalization instead introduces a coupled, multivariate nonlinear transformation (Ni et al., 2024). Consequently, most approximation analyses of structured architectures omit normalization for analytical tractability. Our results treat normalization itself as the intermediate nonlinear operator.

2.2 Theoretical Understanding of Layer and Group Normalization

Normalization has primarily been analyzed through optimization. Existing theory attributes its effectiveness to scale invariance and stabilized training dynamics (Arora et al., 2019; Ba et al., 2016; Huang et al., 2023), as well as improved conditioning of the optimization problem (Cai et al., 2019; Daneshmand et al., 2020; Ghorbani et al., 2019; Karakida et al., 2019; Lyu et al., 2022; Santurkar et al., 2018). These analyses explain important training benefits, but the role of normalization in representation and function approximation remains less understood.

From a representation perspective, prior work identified the intrinsic multivariate nonlinearity of LN and showed that LN-Nets with three neurons per layer and $O(m)$ LN layers can shatter any set of m labeled samples (Ni et al., 2024). That result obtains classification capacity through depth. In contrast, we study function approximation through width: we ask how arranging multiple normalization units in parallel changes the representation class and when normalization itself can provide universal approximation.

Grouping introduces an additional structural choice. Standard Group Normalization (GN; Wu & He, 2018) partitions image channels and computes each group's statistics jointly over spatial positions. Other designs alter the normalization domain: PONO normalizes channels independently at each spatial position (Li et al., 2019), while LN-G-Position partitions those channels further (Ni et al., 2024). Related position-wise grouped RMS-style normalizers also appear in RetNet (Sun et al., 2023) and Mamba-2 (Dao & Gu, 2024), whereas temporal GN variants may pool statistics across sequence positions (Baevski et al., 2020; Chi et al., 2023). Despite their similar names, these operators lack a unified naming convention and may perform substantially different computations.

We formalize PLN as a direct sum of fixed-dimensional LN units whose number grows with width, derive its representation consequences, and extend the construction to sequence and image tensors.

3 PRELIMINARIES AND PARALLEL LAYER NORMALIZATION

We first introduce the network notation used in the approximation results, then define RMS normalization, LN, and their parallel forms, and finally establish two structural properties of the resulting networks. We use lowercase letters for scalars, boldface lowercase letters for vectors, and boldface uppercase letters for matrices. The notation $[\mathbf{x}]_i$ denotes the i-th coordinate of a vector.

3.1 NEURAL NETWORKS

Affine layers. An affine layer $\varphi : \mathbb{R}^d \to \mathbb{R}^m$ is of the form $\varphi(\mathbf{x}) = \boldsymbol{W}\mathbf{x} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$. For consistency with common deep-learning terminology, we continue to call such maps linear layers when no confusion can arise.

Networks with intermediate operators. We consider feed-forward networks obtained by composing affine layers with a class of intermediate operators ϕ.

Definition 3.1 (ϕ-networks)**.** A ϕ-network (or ϕ-Net) of depth L is a function $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ of the form

$$f = \varphi_{L+1} \circ \phi_L \circ \varphi_L \circ \cdots \circ \phi_1 \circ \varphi_1, \tag{1}$$

where the φ_ℓ are affine layers of compatible dimensions and each $\phi_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$ belongs to the same operator class ϕ.

Remark 3.1. If $\phi : \mathbb{R} \to \mathbb{R}$ is an element-wise activation and $\mathbf{x} = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$, we write its coordinate-wise action as $\phi(\mathbf{x}) := (\phi(x_1), \ldots, \phi(x_d))^\top$. In contrast, a normalization operator maps several coordinates jointly. Definition 3.2 applies such fixed-dimensional normalization maps independently to multiple blocks.

Depth and Width. The depth of Eqn.1 is L and its width is $\max\{d_1, \ldots, d_L\}$. We denote by $\mathcal{F}(\phi; L, N)$ the class of ϕ-networks of depth L and width at most N. The case $L = 1$ is called shallow. For simplicity, throughout the paper, function classes compared by inclusion or equality are taken to have the same input and output spaces.

3.2 PARALLEL LAYER NORMALIZATION

RMS normalization and Layer Normalization. For $\mathbf{h} \in \mathbb{R}^d$, define the affine-free RMSNorm core (Zhang & Sennrich, 2019), the centering projection, and LN (Ba et al., 2016) by

$$\mathrm{LS}(\mathbf{h}) = \mathbf{h}/\sqrt{d^{-1}\|\mathbf{h}\|_2^2}, \qquad \boldsymbol{P}_d = \boldsymbol{I}_d - d^{-1}\mathbf{1}\mathbf{1}^\top, \qquad \mathrm{LN}(\mathbf{h}) = \mathrm{LS}(\boldsymbol{P}_d\mathbf{h}). \tag{2}$$

Here LS stands for layer scaling, our shorthand for the RMS normalization operation. We complete the definition at zero denominator by setting $\mathrm{LS}(\mathbf{0}) = \mathbf{0}$, which also gives $\mathrm{LN}(c\mathbf{1}) = \mathbf{0}$. We write $\mathrm{LS}[d]$ and $\mathrm{LN}[d]$ for these maps on $\mathbb{R}^d$, omitting the dimension when clear. Practical implementations typically include learnable coordinate-wise scale and shift parameters (Ba et al., 2016) and add a small number $\varepsilon > 0$[1] to each denominator. We omit the affine parameters because they can be absorbed into an adjacent affine layer.

PLN partitions the feature coordinates into groups and applies LN to all groups in parallel.[2]

Definition 3.2 (Parallel Layer Normalization)**.** Let $d = n_s N$ and write $\mathbf{h} = \mathbf{h}_1 \oplus \cdots \oplus \mathbf{h}_N \in \mathbb{R}^d$, where $\oplus$ denotes concatenation and $\mathbf{h}_i \in \mathbb{R}^{n_s}$. Parallel Layer Normalization with norm size n_s is

$$\mathrm{PLN}[n_s](\mathbf{h}) = \bigoplus_{i=1}^{N} \mathrm{LN}[n_s](\mathbf{h}_i) = \left[\mathrm{LN}[n_s](\mathbf{h}_1)^\top, \ldots, \mathrm{LN}[n_s](\mathbf{h}_N)^\top\right]^\top. \tag{3}$$

Replacing LN by LS in the same direct sum defines Parallel Layer Scaling, $\mathrm{PLS}[n_s]$. We fix n_s and allow the number of groups N to grow with width.

[1] The stabilized definition and its consequences for our theoretical results are discussed in Appendix A.8.

[2] The name PLN distinguishes this vector construction from Group Normalization (GN), which is defined for convolutional feature maps and computes statistics over each channel group and its spatial positions (Wu & He, 2018). Here, each group contains feature coordinates only.

3.3 Structural Properties of PLN Networks

Following Definition 3.1, networks using $\mathrm{LN}[n_s]$ and $\mathrm{PLN}[n_s]$ as intermediate operators are called LN-Nets and PLN-Nets, respectively. The direct-sum structure in Definition 3.2 becomes a sum after the output affine layer. Thus, a shallow PLN-Net is precisely a parallel composition of independently parameterized, fixed-dimensional LN-Nets.

Lemma 3.1. *Given N shallow LN-Nets $\hat{f}_i \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $\hat{f}_i : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ for $1 \le i \le N$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ such that*

$$f(\mathbf{x}) = \sum_{i=1}^{N} \hat{f}_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbb{R}^{d_x}. \tag{4}$$

Conversely, any such PLN-Net can be decomposed into a sum of N shallow LN-Nets with the same input and output dimensions.

Figure 1 illustrates the scalar case; the formal proof for arbitrary input and output dimensions is given in Appendix A.2.

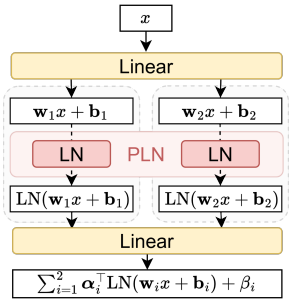


Figure 1: PLN decomposition (Lemma 3.1).

Although the rank-$(d-1)$ centering projection $\boldsymbol{P}_d$ is not invertible on $\mathbb{R}^d$, its image can be linearly identified with $\mathbb{R}^{d-1}$. This yields exact conversions between LN- and LS-Nets and, blockwise, between PLN- and PLS-Nets.

Lemma 3.2. *Shallow LN-Nets with width n_s and shallow LS-Nets with width $n_s - 1$ can represent each other:*

$$\mathcal{F}(\mathrm{LN}[n_s]; 1, n_s) = \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1), \tag{5}$$

for $n_s \ge 2$. Furthermore,

$$\mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N) = \mathcal{F}(\mathrm{PLS}[n_s - 1]; 1, (n_s - 1)N). \tag{6}$$

The first identity is a single-block equivalence; the second follows by combining it with the parallel decomposition in Lemma 3.1. The proof is given in Appendix A.3. Consequently, the subsequent PLN representation and approximation results also apply to PLS whenever they depend only on the represented function class, after shifting each normalization block from width n_s to $n_s - 1$ and the corresponding total width from $n_s N$ to $(n_s - 1)N$. This equivalence also simplifies the subsequent analysis: we may work with LS or PLS when convenient and transfer the resulting conclusions to LN or PLN.

4 Universal Approximation by PLN Networks

We now establish the central representation result for MLPs. A single LN is a multivariate nonlinearity and can realize useful scalar activations after composition with affine layers. However, a shallow network with only one full-width LN remains non-universal, because increasing its width enlarges one coupled nonlinear unit while the number of units remains one. PLN removes this obstruction by placing fixed-dimensional LN units in parallel. We first make this contrast precise and then prove universal approximation for both univariate and multivariate functions.

4.1 Representation Capacity of Shallow LN-Nets

We now characterize the scalar nonlinearities realized by one fixed-dimensional LN unit, the basic component of a PLN-Net.

We first show that one $\mathrm{LN}[n_s]$ unit between two affine layers can exactly represent[3] simple scalar activations, beginning with the sign function.

[3]We use "represent" for exact realization (zero approximation error), and "approximate" when a nonzero error is allowed.

Lemma 4.1. *Given any $\hat{f} \in \mathcal{F}(\mathrm{sign}; 1, 1)$ with $\hat{f} : \mathbb{R} \to \mathbb{R}$, there exists $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$ such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\mathrm{sign}; 1, 1) \subseteq \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s), \tag{7}$$

where $\mathrm{sign}(x)$ is the element-wise sign function taking values 1 if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$.

The proof is given in Appendix A.1. For the special case $n_s = 2$, Figure 2 illustrates the construction.

The sign construction gives an explicit one-dimensional rate, but its derivative vanishes almost everywhere, hindering backpropagation (Amari, 1993). A wider LN unit ($n_s \geq 3$) also realizes the smooth non-polynomial activation $\phi(x) = x/\sqrt{x^2 + 1}$ used for the general UAT.

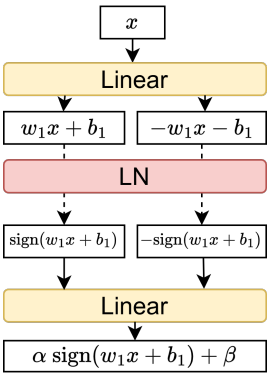


Figure 2: Construction for Lemma 4.1.

Lemma 4.2. *Given any $\hat{f} \in \mathcal{F}(\phi; 1, 1)$ with $\hat{f} : \mathbb{R} \to \mathbb{R}$, there exists $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $n_s \geq 3$ such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\phi; 1, 1) \subseteq \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s), \tag{8}$$

where $\phi(x) = x/\sqrt{x^2 + 1}$.

The proof is provided in Appendix A.4. Using Lemma 3.2, it reduces to proving

$$\mathcal{F}(\phi; 1, 1) \subseteq \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1). \tag{9}$$

For $n_s = 3$, Figure 3 illustrates this construction.

These scalar constructions do not imply universal approximation:

Theorem 4.1 (Weak representation capacity of shallow LN-Nets)**.** *There exists a continuous function $\hat{f}$ on $\Omega \subset \mathbb{R}$ such that every shallow LN-Net $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$ satisfies $|f - \hat{f}|_{L^\infty(\Omega)} \geq 1$.*

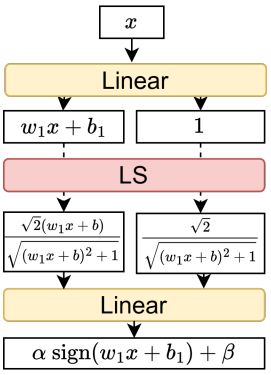


Figure 3: Construction for Lemma 4.2.

The counterexample used in the proof is the smooth elementary function $\hat{f}(x) = \cos(\pi x)$, showing that the limitation stems from the normalization structure. Please see Appendix A.5 for the detailed proof.

Theorem 4.1 shows that increasing the dimension of one LN unit is insufficient for shallow universal approximation. PLN instead fixes the unit dimension and increases the number of units.

4.2 Universal Approximation by PLN Networks

The structural decomposition in Lemma 3.1 lets the number of fixed-dimensional nonlinear units grow with width. We first use it to lift shallow activation networks to PLN-Nets.

A shallow width-N ϕ-Net is a sum of N width-one ϕ-Nets. Together with Lemma 3.1, this gives the following corollary.

Corollary 4.1 (Representation of shallow activation networks)**.** *Suppose every width-one shallow ϕ-Net can be represented by an $\mathrm{LN}[n_s]$-Net. Then, for any input and output dimensions and any $\hat{f} \in \mathcal{F}(\phi; 1, N)$, there exists $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_sN)$ such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\phi; 1, N) \subseteq \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_sN). \tag{10}$$

The construction assigns one LN block to each scalar activation unit; the complete proof is given in Appendix A.6. This yields:

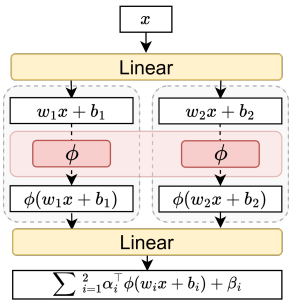


Figure 4: ϕ-Net decomposition (Corollary 4.1).

Theorem 4.2 (Universal approximation of univariate Lipschitz functions by shallow PLN-Nets)**.** *Given any L-Lipschitz continuous function $\hat{f} \in C([0,1])$ and every $\epsilon > 0$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ with $n_s \geq 2$ and $N = \lfloor L/(2\epsilon) \rfloor + 1$ such that*

$$|f - \hat{f}|_{L^\infty} < \epsilon. \tag{11}$$

Proof Sketch. The proof consists of three steps. Step 1: Construct $\bar{f} \in \mathcal{F}(\mathrm{sign}; 1, N)$ such that $|\bar{f} - \hat{f}|_{L^\infty} < \epsilon$. Step 2: Use Lemmas 4.1 and 3.1 to represent $\bar{f}$ exactly by a PLN-Net f. Step 3: Combine the two bounds to obtain $|f - \hat{f}|_{L^\infty} < \epsilon$. The complete proof is given in Appendix A.7.

Remark 4.1. The proof above uses a sign network as an intermediate representation. Since Step 2 represents this network exactly instead of introducing another approximation, the argument and its explicit width bound are particularly direct. When $n_s \geq 3$, a network with the smooth activation $\phi(x) = x/\sqrt{1+x^2}$ can instead serve as the intermediate representation. The same construction applies to practical LN with a stabilizing constant; Appendix A.8 gives the exact representation. The explicit width bound in Theorem 4.2 is specific to the sign-based route.

The same representation transfers standard multivariate approximation rates for smooth non-polynomial activations to PLN-Nets:

Theorem 4.3 (Multivariate approximation by shallow PLN-Nets)**.** *Let $d \geq 2$, $s \geq 1$, and let $\hat{f} : B^d \to \mathbb{R}$ belong to $\mathcal{W}^{s,\infty}(B^d)$ with $\|\hat{f}\|_{\mathcal{W}^{s,\infty}} \leq 1$, where $B^d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq 1\}$. For every $N \geq 1$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ with $n_s \geq 3$ such that*

$$\|f - \hat{f}\|_{L^\infty(B^d)} \leq C N^{-s/d}, \tag{12}$$

where $C = C(d, s)$ is independent of N.

Proof Sketch. The smooth non-polynomial activation $\phi(x) = x/\sqrt{1+x^2}$ admits the stated classical shallow-network rate (Pinkus, 1999). Lemma 4.2 and Corollary 4.1 represent the resulting width-N ϕ-network exactly by a width-$n_s N$ PLN-Net. The complete argument is in Appendix B.2.

Extension to deep networks. The preceding results concern shallow networks. The same representation mechanism extends layer by layer: each scalar activation in a deep activation network is replaced by one fixed-dimensional LN block, and the blocks at the same layer are arranged in parallel.

Lemma 4.3. *Given any $\hat{f} \in \mathcal{F}(\phi; L, N)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, there exists a PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; L, n_s N)$ with $n_s \geq 3$ such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\phi; L, N) \subseteq \mathcal{F}(\mathrm{PLN}[n_s]; L, n_s N), \tag{13}$$

where $\phi(x) = x/\sqrt{x^2+1}$.

Lemma 4.3 is the layer-wise extension of Corollary 4.1. A deep ϕ-Net is a composition of shallow ϕ layers, and the corresponding shallow PLN constructions can be composed in the same order. The detailed proof is given in Appendix B.1.

This exact representation also allows approximation results for deep ReLU networks to be transferred to PLN-Nets.

Theorem 4.4 (Approximation of ReLU Networks by PLN-Nets)**.** *Let $\hat{f} \in \mathcal{F}(\mathrm{ReLU}; L, N)$ with $\hat{f} : [0,1]^d \to \mathbb{R}^m$, and let $\epsilon > 0$. Then there exists a PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 2L, 3n_s N)$ with $n_s \geq 3$ such that $\|f - \hat{f}\|_{L^\infty([0,1]^d)} < \epsilon$.*

Theorem 4.4 follows by combining a deep ϕ-network approximation of the target ReLU network (Zhang et al., 2024) with Lemma 4.3. The detailed proof is provided in Appendix B.3.

Summary. LN already induces nontrivial scalar nonlinearities, but a single full-width LN unit remains non-universal in shallow networks. PLN converts fixed-dimensional normalization units into a width-scalable direct sum, allowing classical activation-network approximation results to transfer to PLN-Nets in arbitrary input dimension. The exact PLN–PLS equivalence established in Lemma 3.2 transfers these conclusions directly to PLS-Nets. We next lift the PLN constructions to structured architectures.

5 Extensions to Structured Architectures

For sequence and image data, we identify PLN designs that preserve the approximation guarantees established above.

5.1 Sequence Models

Definition 5.1 (Sequence PLN)**.** Let $\mathbf{X} \in \mathbb{R}^{B\times T\times C}$ be a sequence tensor, where B is the batch size, T is the sequence length, and $C = Nn_s$ is the feature dimension. For $G_r = \{(r-1)n_s+1, \dots, rn_s\}$, sequence PLN acts token-wise as $[\mathrm{PLN}^{\mathrm{seq}}_{n_s}(\mathbf{X})]_{b,t,G_r} = \mathrm{LN}[n_s](\mathbf{X}_{b,t,G_r})$.

Position-wise affine layers. Suppressing the batch index, a position-wise affine layer shared across tokens has the form $\varphi(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W} + \mathbf{1}_T\boldsymbol{b}^\top$, where $\boldsymbol{X} \in \mathbb{R}^{T\times d}$, $\boldsymbol{W} \in \mathbb{R}^{d\times m}$, and $\boldsymbol{b} \in \mathbb{R}^m$.

Theorem 5.1 (Exact representation of position-wise FFNs by PLN)**.** *Let $\hat{f} = \hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1$ be a position-wise FFN of width N, where $\phi(x) = x/\sqrt{1+x^2}$ acts coordinate-wise. For every $n_s \geq 3$, there exist position-wise affine layers φ_1 and φ_2 of intermediate width n_sN such that $f = \varphi_2 \circ \mathrm{PLN}^{\mathrm{seq}}_{n_s} \circ \varphi_1$ equals $\hat{f}$ on every sequence input.*

Position-wise and ordinary affine layers both act on the last input dimension, so the MLP construction applies at every token with shared parameters (Appendix C.1). CNNs require a separate argument because their kernels mix neighboring positions.

5.2 Convolutional Neural Networks

For image tensors, Figure 5 contrasts PLN with LN, conventional GN, and a mixed-axis alternative, clarifying why we distinguish PLN from conventional GN.

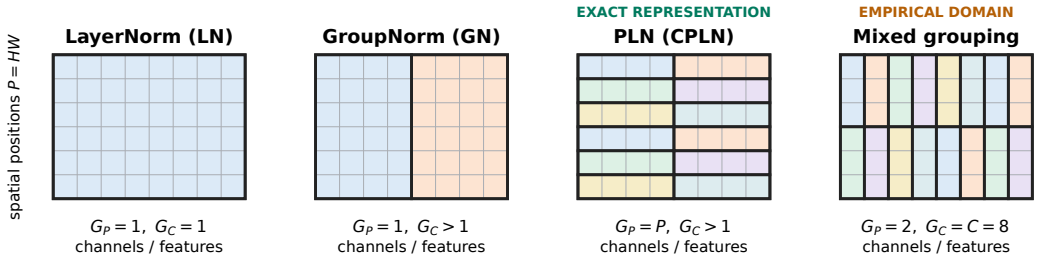


Figure 5: CNN normalization domains (schematic $P = HW = 6$, $C = 8$); boxes mark independent statistics. From left to right: (a) LN uses one $P \times C$ domain; (b) GN groups channels across all positions (Wu & He, 2018); (c) CPLN groups channels at each position; and (d) mixed-axis grouping ($G_P = 2, G_C = C$) is studied empirically.

Definition 5.2 (Image PLN)**.** Let $\mathbf{X} \in \mathbb{R}^{B\times C\times H\times W}$ be an image tensor with $C = Nn_s$. Image PLN acts independently at each spatial position as $[\mathrm{PLN}^{\mathrm{img}}_{n_s}(\mathbf{X})]_{b,G_r,h,w} = \mathrm{LN}[n_s](\mathbf{X}_{b,G_r,h,w})$. We write $\mathrm{CPLN}_{n_s} = \mathrm{PLN}^{\mathrm{img}}_{n_s}$.

Definition 5.3 (Affine Convolutional Layer)**.** Suppressing the batch index and using zero-based spatial indices, an affine two-dimensional convolution with stride (s_h, s_w), dilation (d_h, d_w), and zero padding (p_h, p_w) has the form

$$[\mathcal{C}(\mathbf{X})]_{r,i,j} = b_r + \sum_{c=1}^{C_{\mathrm{in}}} \sum_{(u,v)\in S} K_{r,c,u,v}\, \widetilde{\mathbf{X}}_{c,\, s_h i+d_h u-p_h,\, s_w j+d_w v-p_w}. \tag{14}$$

Here $S \subset \mathbb{Z}^2$ is the kernel support, and $\widetilde{\mathbf{X}}$ is the zero-padded extension of $\mathbf{X}$. The output indices (i, j) range over the grid determined by these spatial hyperparameters.

Theorem 5.2 (Exact representation of convolutional blocks by CPLN). *Let $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$ be arbitrary affine convolutional layers mapping channel dimensions $C_x \to N$ and $N \to C_y$, respectively, and let $\phi(t) = t/\sqrt{1+t^2}$ act element-wise. For every norm size $n_s \geq 3$, there exist affine convolutional layers $\mathcal{C}_1$ and $\mathcal{C}_2$ with the same kernel supports, strides, dilations, and padding as $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$, and intermediate width $n_s N$, such that $\mathcal{C}_2 \circ \mathrm{CPLN}_{n_s} \circ \mathcal{C}_1 = \hat{\mathcal{C}}_2 \circ \phi \circ \hat{\mathcal{C}}_1$ for every input tensor.*

The proof is in Appendix C.2. Layer-wise replacement and folding the intervening 1×1 convolutions into adjacent layers gives an exact deep CPLN network with unchanged depth and spatial specifications (Appendix C.3).

Summary. The MLP representation mechanism extends exactly to position-wise FFNs and convolutional networks through shared affine maps and pointwise 1×1 convolutions. These results cover groups formed from feature coordinates independently at each token or spatial position. Thus, the guarantee applies to position-wise PLN, whereas normalization domains that mix tokens or spatial positions are studied only empirically in Section 6.

6 Experiments

We first use MLP regression and physics-informed experiments to evaluate PLN's approximation behavior. We then compare normalization domains in CNNs and ViTs, including cross-spatial and cross-token groupings that lie outside the preceding theory.

6.1 Empirical Viability of PLN

Function approximation. On a function-regression task, shallow PLN[4] and PLN[8] reach mean test MSEs of 0.00483 and 0.00793, respectively. This demonstrates effective fitting with a trainable multivariate nonlinearity in a single hidden layer; full settings and further results are in Appendix D.1.1.

Physics-informed neural networks. We solve the one-dimensional Helmholtz equation with PINNs (Raissi et al., 2019) and no additional normalization. Table 1 reports the best relative $\mathcal{L}_2$ error at widths 128 and 256; PLN[4] performs best at both, preliminarily indicating its potential for PINNs. Full results are in Appendix D.1.2.

Table 1: Best relative $\mathcal{L}_2$ error for the Helmholtz PINN (lower is better). The lowest error at each width is shown in bold.

Width	PLN[4]	SAT	Tanh	ReLU
128	**0.000035**	0.000104	0.000250	0.993593
256	**0.000036**	0.000052	0.000188	0.998485

6.2 Grouping and Normalization Domains in Image Models

Following Zhang et al. (2017), we use fixed random labels to probe finite-sample fitting capacity. We vary group count and grouped axes within a fixed model. Clean-label training and test accuracy separate fitting from generalization.

CIFAR-10 classification with CNNs. In a depth-4, width-128 CNN, grouped normalization replaces every nonlinear layer. We split channels into G_C groups and each spatial axis into contiguous patches, so every normalization domain is one channel group within one patch. Conventional GN uses the full spatial map, whereas position-wise PLN uses single-position patches, as illustrated in Figure 5(b) and (c). Table 2 compares the empirical expressive capacity of LN, GN, and PLN through random-label fitting. Under the same CNN architecture, PLN fits all labels and outperforms GN, which in turn outperforms LN. The full mixed-axis grid is in Appendix D.2.1.

Table 2: Representative CNN normalization domains on the $50{,}000$-example CIFAR-10 training set with randomized labels (three-seed means).

Operator	Normalization domain	Fitting accuracy (%)
LN	$128 \times 32 \times 32$	24.6
Best GN	$2 \times 32 \times 32$	95.8
PLN[8]	$8 \times 1 \times 1$	**100.0**
PLN[4]	$4 \times 1 \times 1$	**100.0**

In a separate VGG-16 study, we compare PLN[8] with other activation functions across learning rates and initialization scales. PLN[8] trains reliably over a broad range of these settings; detailed results are in Appendix D.2.2.

CIFAR-10 classification with ViTs. We test both clean and randomized labels. Randomization breaks the natural input–label relationship and therefore provides a substantially stricter test of finite-sample fitting capacity. We evaluate grouped normalization at two sites: *all-normalization replacement* replaces LN and retains GELU, while *activation replacement* retains LN and replaces FFN GELUs. For $\mathbb{R}^{B\times T\times C}$, we form $G_T G_C$ token–channel blocks; $G_T = T$ is PLN and $G_T < T$ is Mix-Norm. Complete ViT configuration grids and protocols are in Appendix D.3.

Table 3 shows that, on $20{,}000$ samples with random labels, PLN fits better than the LN–GELU baseline at both replacement sites, while Mix-Norm, whose cross-token grouping is not covered by our theoretical results, achieves the highest three-seed mean at both sites. This indicates stronger empirical fitting capacity for grouped normalization in these finite ViTs.

Table 3: ViT accuracy (%). Baseline is the unchanged LN–GELU model; arrows indicate replacement of all LN layers or all FFN GELUs. Each cell reports the highest three-seed mean / highest single-run accuracy across configurations in that row, selected independently.

Setting	Random 20k	Clean train	Clean test
Baseline	48.2 / 72.8	100.0 / 100.0	71.6 / 72.3
LN→PLN	75.3 / 92.1	100.0 / 100.0	72.3 / 72.9
LN→Mix-Norm	89.5 / 91.2	100.0 / 100.0	**73.5 / 74.5**
GELU→PLN	73.7 / 97.3	99.4 / 100.0	69.7 / 71.0
GELU→Mix-Norm	**91.7 / 98.2**	100.0 / 100.0	70.2 / 70.7

Interestingly, the PLN–LN contrast changes markedly between our random-label fitting and clean-label classification experiments. Compared with LN, PLN shows a clear advantage in fitting random labels, but its advantage in clean-label classification is limited. Its clean-test accuracy is only slightly higher under normalization replacement (72.3% versus 71.6%) and lower under activation replacement (69.7% versus 71.6%). Thus, greater random-label fitting capacity does not translate into a comparable gain in clean-label generalization.

Summary. The shallow MLP experiments provide finite-network evidence consistent with PLN's universal-approximation results, while the PINN study suggests its potential for physics-informed learning. Across CNNs and ViTs, comparisons among LN, PLN, GN, and Mix-Norm show that performance depends not only on group count but also on the grouped axes, replacement site, and evaluation task (random-label fitting versus clean-label generalization).

7 Conclusion

This work identifies a distinction between two ways to scale normalization-based networks: widening one LN unit does not make a shallow LN-Net universal, whereas adding parallel fixed-dimensional units does. By formalizing the latter as PLN, we establish shallow universal-approximation results with explicit bounds and transfer approximation constructions to deep PLN-Nets. The exact PLN–PLS equivalence extends these results to RMSNorm-based networks. Besides, our representation results carry the local mechanism to position-wise feed-forward and convolutional networks. Finite-model experiments support PLN's approximation behavior and show that normalization-domain choices affect CNN and ViT performance.

Limitations and Future Work. Our theorems cover affine–PLN compositions with normalization local to a token or spatial position, but not cross-position domains or arbitrary compositions with additional nonlinear activations. The experiments provide finite-model evidence rather than empirical proof of universality, and the PINN results remain preliminary. In future work, we will extend the theory and experiments to broader tasks and cross-position normalization.

AI Use Statement

We used generative AI tools to assist with manuscript editing, proof checking, literature search and citation checking, code implementation and debugging, and the preparation and interpretation of

synthetic experiments and figures. The research question, mathematical claims and constructions, and experimental objectives were formulated by the authors. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content.

REFERENCES

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5): 185–196, 1993.

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *ICLR*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *NeurIPS*, volume 33, 2020.

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. In *ICML*, 2019.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Hadi Daneshmand, Jonas Moritz Kohler, Francis R. Bach, Thomas Hofmann, and Aurélien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In *NeurIPS*, 2020.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Tim De Ryck, Samuel Lanthaler, and Siddhartha Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *ICML*, 2019.

Gustaf Gripenberg. Approximation by neural networks with a bounded number of nodes at each level. *Journal of approximation theory*, 122(2):260–266, 2003.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2): 251–257, 1991.

Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

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

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Boyi Li, Felix Wu, Kilian Q. Weinberger, and Serge Belongie. Positional normalization. In *NeurIPS*, volume 32, 2019.

Zhong Li, Jiequn Han, Qianxiao Li, et al. Approximation and optimization theory for linear continuous-time recurrent neural networks. *Journal of Machine Learning Research*, 23(42):1–85, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

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

Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1-3):81–91, 1999.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

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

Kenta Oono and Taiji Suzuki. Approximation and non-parametric estimation of resnet-type convolutional neural networks, 2023. URL https://arxiv.org/abs/1903.10047.

Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. *arXiv preprint arXiv:2006.08859*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 2019.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

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

Jonghyun Shin, Namjun Kim, Geonho Hwang, and Sejun Park. Minimum width for universal approximation using squashable activation functions, 2025. URL https://arxiv.org/abs/2504.07371.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks, 2017. URL `https://arxiv.org/abs/1610.01145`.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions?, 2020. URL `https://arxiv.org/abs/1912.10077`.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Shijun Zhang, Jianfeng Lu, and Hongkai Zhao. Deep network approximation: Beyond relu to diverse activation functions. *Journal of Machine Learning Research*, 25(35):1–39, 2024.

A Mathematical Proofs in Section 4

A.1 Proof of Lemma 4.1

Lemma 4.1 Given any $\hat{f} \in \mathcal{F}(\text{sign}; 1, 1)$ with $\hat{f} : \mathbb{R} \to \mathbb{R}$, there exists $f \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$ such that $f = \hat{f}$. In short,

$$\mathcal{F}(\text{sign}; 1, 1) \subseteq \mathcal{F}(\text{LN}[n_s]; 1, n_s), \tag{15}$$

where $\text{sign}(x)$ is the element-wise sign function taking values 1 if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$.

Proof. For any $\hat{f} \in \mathcal{F}(\text{sign}; 1, 1)$ and $\hat{f} : \mathbb{R} \to \mathbb{R}$, we have $\hat{f}(x) = w_2 \,\text{sign}(w_1 x + b_1) + b_2$.

For $f \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$, we construct $f(x) = \boldsymbol{v}_2^\top \text{LN}(\boldsymbol{v}_1 x + \boldsymbol{c}_1) + c_2$, where $\boldsymbol{v}_1, \boldsymbol{c}_1, \boldsymbol{v}_2 \in \mathbb{R}^{n_s}$ and $c_2 \in \mathbb{R}$.

Let

$$\boldsymbol{v}_1 = \begin{bmatrix} w_1 \\ -w_1 \\ \mathbf{0}_{n_s-2} \end{bmatrix}, \boldsymbol{c}_1 = \begin{bmatrix} b_1 \\ -b_1 \\ \mathbf{0}_{n_s-2} \end{bmatrix}, \boldsymbol{v}_2 = \begin{bmatrix} \sqrt{2/n_s} w_2 \\ 0 \\ \mathbf{0}_{n_s-2} \end{bmatrix}, c_2 = b_2. \tag{16}$$

We can identify that the first element of $\text{LN}(\boldsymbol{v}_1 x + \boldsymbol{c}_1)$ is

$$\frac{w_1 x + b_1}{\sqrt{2(w_1 x + b_1)^2 / n_s}} = \sqrt{\frac{n_s}{2}} \frac{w_1 x + b_1}{|w_1 x + b_1|} = \sqrt{\frac{n_s}{2}} \,\text{sign}(w_1 x + b_1), \tag{17}$$

if $w_1 x + b_1 \neq 0$.

For the case $w_1 x + b_1 = 0$, we have $\sigma^2 = 2(w_1 x + b_1)^2 / n_s = 0$, then $\text{LN}(\boldsymbol{v}_1 x + \boldsymbol{c}_1) = \mathbf{0}$ by its definition.

Therefore, the first element of $\text{LN}(\boldsymbol{v}_1 x + \boldsymbol{c}_1)$ is equal to $\sqrt{\frac{n_s}{2}} \,\text{sign}(w_1 x + b_1)$, thus

$$\begin{aligned} f(x) &= \boldsymbol{v}_2^\top \text{LN}(\boldsymbol{v}_1 x + \boldsymbol{c}_1) + c_2 \\ &= \sqrt{2/n_s} w_2 \cdot \sqrt{\frac{n_s}{2}} \,\text{sign}(w_1 x + b_1) + c_2 \\ &= w_2 \,\text{sign}(w_1 x + b_1) + b_2 = \hat{f}(x). \end{aligned} \tag{18}$$

This completes the proof. □

We can extend Lemma 4.1 to the version $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$.

Corollary A.1. *Given any $\hat{f} \in \mathcal{F}(\text{sign}; 1, 1)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, there exists $f \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$, such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\text{sign}; 1, 1) \subseteq \mathcal{F}(\text{LN}[n_s]; 1, n_s), \tag{19}$$

where $\text{sign}(x)$ is the element-wise sign function, which takes 1 if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$.

Proof. For any $\hat{f} \in \mathcal{F}(\text{sign}; 1, 1)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, write $\hat{f}(\mathbf{x}) = \boldsymbol{w}_2 \,\text{sign}(\boldsymbol{w}_1^\top \mathbf{x} + b_1) + \boldsymbol{b}_2$, where

$$\boldsymbol{w}_1 \in \mathbb{R}^d, b_1 \in \mathbb{R}, \boldsymbol{w}_2, \boldsymbol{b}_2 \in \mathbb{R}^m. \tag{20}$$

Define

$$\boldsymbol{V}_1 = \begin{bmatrix} \boldsymbol{w}_1^\top \\ -\boldsymbol{w}_1^\top \\ \boldsymbol{O}_{(n_s-2)\times d} \end{bmatrix} \in \mathbb{R}^{n_s \times d}, \qquad \boldsymbol{c}_1 = \begin{bmatrix} b_1 \\ -b_1 \\ \mathbf{0}_{n_s-2} \end{bmatrix} \in \mathbb{R}^{n_s}, \tag{21}$$

and

$$\boldsymbol{V}_2 = \begin{bmatrix} \sqrt{2/n_s}\, \boldsymbol{w}_2 & \boldsymbol{O}_{m\times(n_s-1)} \end{bmatrix} \in \mathbb{R}^{m \times n_s}. \tag{22}$$

Then $f(\mathbf{x}) = \boldsymbol{V}_2 \text{LN}(\boldsymbol{V}_1 \mathbf{x} + \boldsymbol{c}_1) + \boldsymbol{b}_2 = \hat{f}(\mathbf{x})$.

The following proof is almost the same as that in the proof of Lemma 4.1. □

A.2 Proof of Lemma 3.1

Lemma 3.1 Given N shallow LN-Nets $\hat{f}_i \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $\hat{f}_i : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ for $1 \le i \le N$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ such that

$$f(\mathbf{x}) = \sum_{i=1}^{N} \hat{f}_i(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbb{R}^{d_x}. \tag{23}$$

Conversely, any such PLN-Net can be decomposed into a sum of N shallow LN-Nets with the same input and output dimensions.

Proof. Given N shallow LN-Nets $\hat{f}_i \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ for $1 \le i \le N$, write $\hat{f}_i(\mathbf{x}) = \boldsymbol{W}_{i2}\mathrm{LN}(\boldsymbol{W}_{i1}\mathbf{x} + \boldsymbol{b}_{i1}) + \boldsymbol{b}_{i2}$, where

$$\boldsymbol{W}_{i1} \in \mathbb{R}^{n_s \times d_x}, \quad \boldsymbol{b}_{i1} \in \mathbb{R}^{n_s}, \quad \boldsymbol{W}_{i2} \in \mathbb{R}^{d_y \times n_s}, \quad \boldsymbol{b}_{i2} \in \mathbb{R}^{d_y}. \tag{24}$$

Define $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ by $f(\mathbf{x}) = \boldsymbol{W}_2\mathrm{PLN}(\boldsymbol{W}_1\mathbf{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$, where

$$\boldsymbol{W}_1 \in \mathbb{R}^{n_s N \times d_x}, \quad \boldsymbol{b}_1 \in \mathbb{R}^{n_s N}, \quad \boldsymbol{W}_2 \in \mathbb{R}^{d_y \times n_s N}, \quad \boldsymbol{b}_2 \in \mathbb{R}^{d_y}. \tag{25}$$

Set

$$\boldsymbol{W}_1 = \begin{bmatrix} \boldsymbol{W}_{11} \\ \boldsymbol{W}_{21} \\ \vdots \\ \boldsymbol{W}_{N1} \end{bmatrix}, \boldsymbol{b}_1 = \begin{bmatrix} \boldsymbol{b}_{11} \\ \boldsymbol{b}_{21} \\ \vdots \\ \boldsymbol{b}_{N1} \end{bmatrix}, \boldsymbol{W}_2 = [\boldsymbol{W}_{12} \quad \boldsymbol{W}_{22} \quad \cdots \quad \boldsymbol{W}_{N2}], \boldsymbol{b}_2 = \sum_{i=1}^{N} \boldsymbol{b}_{i2}. \tag{26}$$

By blockwise application of PLN, this construction gives $f(\mathbf{x}) = \sum_{i=1}^{N} \hat{f}_i(\mathbf{x})$. Conversely, split the input matrix, hidden bias, and output matrix of any PLN-Net into N conforming blocks and distribute its output bias across the N LN-Nets. This yields the stated decomposition. □

A.3 Proof of Lemma 3.2

Lemma 3.2 can simplify most the proofs, so we prove it at the early time.

Lemma 3.2 Shallow LN-Nets with width n_s and shallow LS-Nets with width $n_s - 1$ can represent each other, namely

$$\mathcal{F}(\mathrm{LN}[n_s]; 1, n_s) = \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1), \tag{27}$$

for $n_s \ge 2$. Furthermore,

$$\mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N) = \mathcal{F}(\mathrm{PLS}[n_s - 1]; 1, (n_s - 1)N). \tag{28}$$

We first require Lemma A.2, which follows from Lemma 6 of Ni et al. (2024).

Lemma A.1 (Ni et al., 2024, Lemma 6)**.** *There is some orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$, such that $\boldsymbol{z} = \boldsymbol{Q}\begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix} \in \{\boldsymbol{z} \in \mathbb{R}^d : z^{(1)} + \cdots + z^{(d)} = 0\}$ (namely $\boldsymbol{z}$ is centralized), for $\mathbf{x} \in \mathbb{R}^{d-1}$,*

Lemma A.2. *Denote $\mathrm{LN}(\cdot)$ as the LN operation in $\mathbb{R}^d (d \ge 3)$, and $\mathrm{LS}(\cdot)$ as the LS operation in $\mathbb{R}^{d-1}$. We can find some linear transformations φ_1 and φ_2, such that*

$$\mathrm{LS}(\cdot) = \varphi_2 \circ \mathrm{LN}(\cdot) \circ \varphi_1. \tag{29}$$

Besides, we can find some linear transformations φ_1^* and φ_2^*, such that

$$\mathrm{LN}(\cdot) = \varphi_2^* \circ \mathrm{LS}(\cdot) \circ \varphi_1^*. \tag{30}$$

Proof. For $\mathbf{x} \in \mathbb{R}^{d-1}$ and $\mathbf{y} \in \mathbb{R}^d$, Equation 2 gives

$$\mathrm{LS}(\mathbf{x}) = \sqrt{d-1}\,\mathbf{x}/\|\mathbf{x}\|_2, \qquad \mathrm{LN}(\mathbf{y}) = \sqrt{d}\,\boldsymbol{P}_d\mathbf{y}/\|\boldsymbol{P}_d\mathbf{y}\|_2. \tag{31}$$

Based on the orthogonal matrix $\boldsymbol{Q}$, we obtain

$$\|\boldsymbol{z}\|_2 = \left\|\boldsymbol{Q}\begin{bmatrix}\mathbf{x}\\0\end{bmatrix}\right\|_2 = \left\|\begin{bmatrix}\mathbf{x}\\0\end{bmatrix}\right\|_2 = \|\mathbf{x}\|_2. \tag{32}$$

By Lemma A.1, we have $\mathbf{1}^\top\boldsymbol{z} = 0$ and $\boldsymbol{P}_d\boldsymbol{z} = \boldsymbol{z}$. We hence find that

$$\mathrm{LN}(\boldsymbol{z}) = \sqrt{d}\,\boldsymbol{z}/\|\boldsymbol{z}\|_2. \tag{33}$$

Let $\boldsymbol{I}_d$ denotes the identity matrix in $\mathbb{R}^{d\times d}$. We thus have

$$\begin{aligned}
\frac{\sqrt{d-1}}{\sqrt{d}}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathrm{LN}(\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x}) =& \frac{\sqrt{d-1}}{\sqrt{d}}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathrm{LN}(\boldsymbol{z})\\
=& \frac{\sqrt{d-1}}{\sqrt{d}}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\sqrt{d}\,\boldsymbol{z}/\|\boldsymbol{z}\|_2\\
=& \frac{\sqrt{d-1}}{\sqrt{d}}\sqrt{d}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\boldsymbol{Q}\begin{bmatrix}\mathbf{x}\\0\end{bmatrix}/\|\mathbf{x}\|_2\\
=& \sqrt{d-1}\,\mathbf{x}/\|\mathbf{x}\|_2\\
=& \mathrm{LS}(\mathbf{x}).
\end{aligned} \tag{34}$$

Let $\varphi_1(\mathbf{x}) = \boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x}$ and $\varphi_2(\mathbf{x}) = \dfrac{\sqrt{d-1}}{\sqrt{d}}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{x}$. We observe that

$$\mathrm{LS}(\cdot) = \varphi_2 \circ \mathrm{LN}(\cdot) \circ \varphi_1. \tag{35}$$

Now we prove the inverse equation. According to $\mathbf{z} = \boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x}$, we have

$$\begin{aligned}
\mathrm{LS}([\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{z}) &= \mathrm{LS}([\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x})\\
&= \mathrm{LS}([\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x})\\
&= \mathrm{LS}(\boldsymbol{I}_{d-1}\,\mathbf{x})\\
&= \mathrm{LS}(\mathbf{x}).
\end{aligned} \tag{36}$$

We thus have

$$\begin{aligned}
\frac{\sqrt{d}}{\sqrt{d-1}}\,\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathrm{LS}([\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{z}) &= \frac{\sqrt{d}}{\sqrt{d-1}}\,\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathrm{LS}(\mathbf{x})\\
&= \frac{\sqrt{d}}{\sqrt{d-1}}\,\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\sqrt{d-1}\,\mathbf{x}/\|\mathbf{x}\|_2\\
&= \sqrt{d}\,\mathbf{z}/\|\mathbf{z}\|_2\\
&= \mathrm{LN}(\mathbf{z}).
\end{aligned} \tag{37}$$

The calculation above covers every centered input $\boldsymbol{z}$. For an arbitrary $\mathbf{y} \in \mathbb{R}^d$, let $\boldsymbol{z} = \boldsymbol{P}_d\mathbf{y}$. The first $d-1$ columns of $\boldsymbol{Q}$ span the centered subspace, so $[\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{y} = [\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{z}$, while $\mathrm{LN}(\mathbf{y}) = \mathrm{LN}(\mathbf{z})$. This also holds at zero variance under the stated zero-output conventions.

Let $\varphi_1^*(\mathbf{x}) = [\boldsymbol{I}_{d-1} \quad \mathbf{0}]\,\boldsymbol{Q}^\top\mathbf{x}$ and $\varphi_2^*(\mathbf{x}) = \dfrac{\sqrt{d}}{\sqrt{d-1}}\,\boldsymbol{Q}\,[\boldsymbol{I}_{d-1} \quad \mathbf{0}]^\top\mathbf{x}$. We observe that

$$\mathrm{LN}(\cdot) = \varphi_2^* \circ \mathrm{LS}(\cdot) \circ \varphi_1^*. \tag{38}$$

□

For $d = 2$, the same two identities follow directly without Lemma A.1. Under the conventions $\mathrm{LS}(0) = 0$ and $\mathrm{LN}(c\mathbf{1}) = \mathbf{0}$, define

$$\varphi_1(t) = (t, 0)^\top, \qquad \varphi_2(z_1, z_2) = z_1, \tag{39}$$

$$\varphi_1^*(z_1, z_2) = z_1 - z_2, \qquad \varphi_2^*(t) = (t, -t)^\top. \tag{40}$$

Since $\mathrm{LS}(t) = \mathrm{sign}(t)$ and $\mathrm{LN}(z_1, z_2) = \big(\mathrm{sign}(z_1 - z_2), -\mathrm{sign}(z_1 - z_2)\big)^\top$, these maps satisfy

$$\mathrm{LS} = \varphi_2 \circ \mathrm{LN} \circ \varphi_1, \qquad \mathrm{LN} = \varphi_2^* \circ \mathrm{LS} \circ \varphi_1^*. \tag{41}$$

Now we prove Lemma 3.2.

Proof. Step 1: Represent LN-Nets by LS-Nets. Namely given any $\hat{f} \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$, there exists $f \in \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1)$, such that $f = \hat{f}$.

Set $\hat{f} = \boldsymbol{W}_2 \circ \mathrm{LN} \circ \boldsymbol{W}_1$, where $\boldsymbol{W}_1 : \mathbb{R}^d \to \mathbb{R}^{n_s}$ and $\boldsymbol{W}_2 : \mathbb{R}^{n_s} \to \mathbb{R}^m$ are affine layers. For $n_s \geq 3$, use Lemma A.2; for $n_s = 2$, use the explicit maps above. In either case, $\mathrm{LN} = \varphi_2^* \circ \mathrm{LS} \circ \varphi_1^*$. Hence choose $\boldsymbol{V}_1 = \varphi_1^* \circ \boldsymbol{W}_1$ and $\boldsymbol{V}_2 = \boldsymbol{W}_2 \circ \varphi_2^*$ to obtain $f = \boldsymbol{V}_2 \circ \mathrm{LS} \circ \boldsymbol{V}_1 = \hat{f}$.

Step 2: Represent LS-Nets by LN-Nets. Namely given any $\hat{f} \in \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1)$, there exists $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$, such that $f = \hat{f}$.

Set $\hat{f} = \boldsymbol{W}_2 \circ \mathrm{LS} \circ \boldsymbol{W}_1$, where $\boldsymbol{W}_1 : \mathbb{R}^d \to \mathbb{R}^{n_s - 1}$ and $\boldsymbol{W}_2 : \mathbb{R}^{n_s - 1} \to \mathbb{R}^m$ are affine layers. Using $\mathrm{LS} = \varphi_2 \circ \mathrm{LN} \circ \varphi_1$, choose $\boldsymbol{V}_1 = \varphi_1 \circ \boldsymbol{W}_1$ and $\boldsymbol{V}_2 = \boldsymbol{W}_2 \circ \varphi_2$. Then $f = \boldsymbol{V}_2 \circ \mathrm{LN} \circ \boldsymbol{V}_1 = \hat{f}$.

Step 3: Extend to parallel structures.

For simplicity, we define the sum of networks as

$$\mathcal{S}(m, \phi; L, N) = \left\{ f = \sum_{i=1}^{m} f_i : f_i \in \mathcal{F}(\phi; L, N), 1 \leq i \leq m \right\}, \tag{42}$$

which represents the sum of m functions from $\mathcal{F}(\phi; L, N)$.

Then by Lemma 3.1, we obtain that

$$\mathcal{S}(N, \mathrm{LN}[n_s]; 1, n_s) = \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N). \tag{43}$$

With the similar proof of Lemma 3.1, we will obtain

$$\mathcal{S}(N, \mathrm{LS}[n_s - 1]; 1, n_s - 1) = \mathcal{F}(\mathrm{PLS}[n_s - 1]; 1, (n_s - 1)N). \tag{44}$$

Since $\mathcal{F}(\mathrm{LN}[n_s]; 1, n_s) = \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1)$, it is easy to identify that

$$\mathcal{S}(N, \mathrm{LS}[n_s - 1]; 1, n_s - 1) = \mathcal{S}(N, \mathrm{LN}[n_s]; 1, n_s), \tag{45}$$

by the definition of $\mathcal{S}$.

Therefore, we obtain that

$$\mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N) = \mathcal{F}(\mathrm{PLS}[n_s - 1]; 1, (n_s - 1)N). \tag{46}$$

Here we complete the proof. □

A.4 Proof of Lemma 4.2

Lemma 4.2 Given any $\hat{f} \in \mathcal{F}(\phi; 1, 1)$ with $\hat{f} : \mathbb{R} \to \mathbb{R}$, there exists $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $n_s \geq 3$ such that $f = \hat{f}$. In short,

$$\mathcal{F}(\phi; 1, 1) \subseteq \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s), \tag{47}$$

where $\phi(x) = x/\sqrt{x^2 + 1}$.

Proof. To simplify the proof, we apply Lemma 3.2 first, which says that $\mathcal{F}(\mathrm{LN}[n_s]; 1, n_s) = \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1)$

For any $\hat{f} \in \mathcal{F}(\phi; 1, 1)$, we have $\hat{f}(x) = w_2\phi(w_1x + b_1) + b_2$, where $\phi(x) = x/\sqrt{x^2 + 1}$.

For $f \in \mathcal{F}(\mathrm{LS}[n_s - 1]; 1, n_s - 1)$ with $n_s \geq 3$, construct $f(x) = \boldsymbol{v}_2^\top \mathrm{LS}(\boldsymbol{v}_1x + \boldsymbol{c}_1) + c_2$, where $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{c}_1 \in \mathbb{R}^{n_s-1}$ and $c_2 \in \mathbb{R}$.

Let

$$\boldsymbol{v}_1 = \begin{bmatrix} w_1 \\ 0 \\ \mathbf{0}_{n_s-3} \end{bmatrix}, \boldsymbol{c}_1 = \begin{bmatrix} b_1 \\ 1 \\ \mathbf{0}_{n_s-3} \end{bmatrix}, \boldsymbol{v}_2 = \begin{bmatrix} w_2/\sqrt{n_s - 1} \\ 0 \\ \mathbf{0}_{n_s-3} \end{bmatrix}, c_2 = b_2. \tag{48}$$

We can identify that the first element of $\mathrm{LS}(\boldsymbol{v}_1x + \boldsymbol{c}_1)$ is

$$\frac{w_1x + b_1}{\sqrt{[(w_1x + b_1)^2 + 1]/(n_s - 1)}} = \sqrt{n_s - 1}\frac{w_1x + b_1}{\sqrt{(w_1x + b_1)^2 + 1}} = \sqrt{n_s - 1}\phi(w_1x + b_1). \tag{49}$$

Therefore, we obtain

$$\begin{aligned} f(x) &= \boldsymbol{v}_2^\top \mathrm{LS}(\boldsymbol{v}_1x + \boldsymbol{c}_1) + c_2 \\ &= w_2/\sqrt{n_s - 1} \cdot \sqrt{n_s - 1}\phi(w_1x + b_1) + c_2 \\ &= w_2\phi(w_1x + b_1) + b_2 = \hat{f}(x). \end{aligned} \tag{50}$$

This completes the proof. □

We can also extend it to high dimensions.

Corollary A.2. *Given any $\hat{f} \in \mathcal{F}(\phi; 1, 1)$ and $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, there exists some $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ with $n_s \geq 3$, such that $f = \hat{f}$. In other words, we have*

$$\mathcal{F}(\phi; 1, 1) \subseteq \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s), \tag{51}$$

where $\phi(x) = x/\sqrt{x^2 + 1}$ is an activation function.

Proof Sketch. Follow the scalar construction above, replacing w_1x by $\boldsymbol{w}_1^\top \mathbf{x}$ and the scalar output weight by a vector in $\mathbb{R}^m$.

A.5 Proof of Theorem 4.1

We first characterize the possible shapes of a scalar function represented by one shallow LN unit.

Lemma A.3 (Structure of scalar shallow LN-Nets)**.** *Let $n_s \geq 2$ and let $f \in \mathcal{F}(\mathrm{LN}[n_s]; 1, n_s)$ map $\mathbb{R}$ to $\mathbb{R}$. Then either $f \in \mathcal{F}(\mathrm{sign}; 1, 1)$, or $f \in C^\infty(\mathbb{R})$ is nonconstant and f' has at most one zero.*

Proof. Let $d = n_s - 1$. By Lemma 3.2, every such function can be written as

$$f(x) = \frac{\boldsymbol{v}^\top(\boldsymbol{w}x + \boldsymbol{b})}{\|\boldsymbol{w}x + \boldsymbol{b}\|_2} + b_0, \tag{52}$$

where $\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{v} \in \mathbb{R}^d$ and $b_0 \in \mathbb{R}$; the factor $\sqrt{d}$ in $\mathrm{LS}(\mathbf{h}) = \sqrt{d}\,\mathbf{h}/\|\mathbf{h}\|_2$ has been absorbed into the free output vector $\boldsymbol{v}$.

If $\boldsymbol{w} = \mathbf{0}$, then f is constant and hence belongs to $\mathcal{F}(\mathrm{sign}; 1, 1)$. If $\boldsymbol{w} \neq \mathbf{0}$ and $\boldsymbol{b} = k\boldsymbol{w}$ for some $k \in \mathbb{R}$, then, using the zero-output convention at $x = -k$,

$$f(x) = \frac{\boldsymbol{v}^\top \boldsymbol{w}}{\|\boldsymbol{w}\|_2}\,\mathrm{sign}(x + k) + b_0, \tag{53}$$

so again $f \in \mathcal{F}(\mathrm{sign}; 1, 1)$.

It remains to consider $\boldsymbol{w} \neq \mathbf{0}$ and $\boldsymbol{b} \notin \mathrm{span}\{\boldsymbol{w}\}$. Then $\boldsymbol{w}x + \boldsymbol{b} \neq \mathbf{0}$ for every x, and strict Cauchy–Schwarz gives

$$\inf_{x \in \mathbb{R}} \|\boldsymbol{w}x + \boldsymbol{b}\|_2^2 = \|\boldsymbol{b}\|_2^2 - \frac{(\boldsymbol{w}^\top \boldsymbol{b})^2}{\|\boldsymbol{w}\|_2^2} > 0. \tag{54}$$

Thus $f \in C^\infty(\mathbb{R})$. Set $\alpha = \boldsymbol{v}^\top \boldsymbol{w}$, $\beta = \boldsymbol{v}^\top \boldsymbol{b}$, $p = \|\boldsymbol{w}\|_2^2$, $q = \boldsymbol{w}^\top \boldsymbol{b}$, and $r = \|\boldsymbol{b}\|_2^2$. Direct differentiation yields

$$f'(x) = \frac{(\alpha q - \beta p)x + (\alpha r - \beta q)}{\|\boldsymbol{w}x + \boldsymbol{b}\|_2^3}. \tag{55}$$

If the two numerator coefficients are not both zero, this derivative has at most one zero. Otherwise, $\alpha q = \beta p$ and $\alpha r = \beta q$. Since $pr - q^2 > 0$, these identities imply $\alpha = \beta = 0$, and hence $f \equiv b_0$ is constant. Such a function belongs to $\mathcal{F}(\text{sign}; 1, 1)$, completing the dichotomy. □

Now we prove Theorem 4.1 based on Lemma A.3.

Theorem 4.1 (Weak representation capacity of LN-Nets). There exists a continuous function $\hat{f}$ on $\Omega \subset \mathbb{R}$, such that for any shallow LN-Net $f \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ with $n_s \geq 2$,

$$|f - \hat{f}|_{L^\infty(\Omega)} \geq 1. \tag{56}$$

Proof. Take $\Omega = [-2, 2]$ and $\hat{f}(x) = \cos(\pi x)$. Suppose for contradiction that $|f(x) - \hat{f}(x)| < 1$ on $[-2, 2]$.

First suppose $f \in \mathcal{F}(\text{sign}; 1, 1)$ and write $f(x) = w_2 \,\text{sign}(w_1 x + b_1) + b_2$. If $w_1 = 0$, then f is constant, so $f(1) = f(2)$. If $w_1 \neq 0$, let $t_0 = -b_1/w_1$. When $t_0 \leq 0$, the function has the same value at 1 and 2; when $t_0 > 0$, it has the same value at -2 and -1. Thus, in every case, there are x_1, x_2 equal to either $1, 2$ or $-2, -1$ such that $f(x_1) = f(x_2)$ and $|\hat{f}(x_2) - \hat{f}(x_1)| = 2$. Consequently,

$$2 \leq |\hat{f}(x_2) - f(x_2)| + |f(x_1) - \hat{f}(x_1)| < 2, \tag{57}$$

a contradiction.

Now suppose $f \notin \mathcal{F}(\text{sign}; 1, 1)$. By Lemma A.3, $f \in C^1(\mathbb{R})$ and has at most one stationary point. The assumed error bound and the values of $\hat{f}$ imply

$$f(-2) > 0, \qquad f(-1) < 0, \qquad f(0) > 0, \qquad f(1) < 0. \tag{58}$$

The mean value theorem gives points $\eta_1 \in (-2, -1)$, $\eta_2 \in (-1, 0)$, and $\eta_3 \in (0, 1)$ such that $f'(\eta_1) < 0$, $f'(\eta_2) > 0$, and $f'(\eta_3) < 0$. Since f' is continuous and $\eta_1 < \eta_2 < \eta_3$, the intermediate value theorem gives distinct points $\xi_1 \in (\eta_1, \eta_2)$ and $\xi_2 \in (\eta_2, \eta_3)$ satisfying $f'(\xi_1) = f'(\xi_2) = 0$. This contradicts Lemma A.3. Hence the uniform error is at least one. □

A.6 Proof of Corollary 4.1

Corollary 4.1 Suppose there exists $g \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ that can represent any $\hat{g} \in \mathcal{F}(\phi; 1, 1)$. Then for any $\hat{f} \in \mathcal{F}(\phi; 1, N)$ with $\hat{f} : \mathbb{R} \to \mathbb{R}$, there exists a shallow PLN-Net $f \in \mathcal{F}(\text{PLN}[n_s]; 1, n_s N)$ such that $f = \hat{f}$. In short,

$$\mathcal{F}(\phi; 1, N) \subseteq \mathcal{F}(\text{PLN}[n_s]; 1, n_s N). \tag{59}$$

Proof. Let $\hat{f} \in \mathcal{F}(\phi; 1, N)$ and $f : \mathbb{R} \to \mathbb{R}$. Set $\hat{f}(x) = \boldsymbol{w}_2^\top \phi(\boldsymbol{w}_1 x + \boldsymbol{b}_1) + b_2$. Note that ϕ is an element-wise activation function, we then have

$$\hat{f}(x) = \boldsymbol{w}_2^\top \phi(\boldsymbol{w}_1 x + \boldsymbol{b}_1) + b_2 = \sum_{i=1}^{N} w_{2i} \phi(w_{1i} x + b_{1i}) + b_{2i}, \tag{60}$$

where

$$\boldsymbol{w}_1 = \begin{bmatrix} w_{11} \\ \vdots \\ w_{1N} \end{bmatrix}, \boldsymbol{w}_2 = \begin{bmatrix} w_{21} \\ \vdots \\ w_{2N} \end{bmatrix}, \boldsymbol{b}_1 = \begin{bmatrix} b_{11} \\ \vdots \\ b_{1N} \end{bmatrix}, b_2 = \sum_{i=1}^{N} b_{2i}. \tag{61}$$

That means $\hat{f}$ can be represented by the sum of N shallow ϕ-Nets $\hat{f}_i \in \mathcal{F}(\phi; 1, 1)$. By Lemma 4.2, each f_i can be represented by a shallow LN-Net $f_i \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$. By Lemma 3.1, the sum of N shallow LN-Nets $f_i \in \mathcal{F}(\text{LN}[n_s]; 1, n_s)$ can be represented by a shallow PLN-Net $f \in \mathcal{F}(\text{PLN}[n_s]; 1, n_s N)$. Therefore, f can represent $\hat{f}$.

Here we complete the proof. □

Similarly, we can extend the corollary to the case $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$.

Corollary A.3. *Suppose every width-one shallow ϕ-Net can be represented by an $\mathrm{LN}[n_s]$-Net. Then for any $\hat{f} \in \mathcal{F}(\phi; 1, N)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ such that $f = \hat{f}$. In short,*

$$\mathcal{F}(\phi; 1, N) \subseteq \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N). \tag{62}$$

Proof. Let $\hat{f} \in \mathcal{F}(\phi; 1, N)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$. Write $\hat{f}(\mathbf{x}) = \boldsymbol{W}_2 \phi(\boldsymbol{W}_1 \mathbf{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$. Since ϕ acts element-wise,

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \left[\boldsymbol{w}_{2i} \phi(\boldsymbol{w}_{1i}^\top \mathbf{x} + b_{1i}) + \boldsymbol{b}_{2i} \right], \tag{63}$$

where

$$\boldsymbol{W}_1 = \begin{bmatrix} \boldsymbol{w}_{11}^\top \\ \vdots \\ \boldsymbol{w}_{1N}^\top \end{bmatrix}, \quad \boldsymbol{W}_2 = [\boldsymbol{w}_{21} \quad \cdots \quad \boldsymbol{w}_{2N}], \quad \boldsymbol{b}_1 = \begin{bmatrix} b_{11} \\ \vdots \\ b_{1N} \end{bmatrix}, \quad \boldsymbol{b}_{21} = \boldsymbol{b}_2, \quad \boldsymbol{b}_{2i} = \mathbf{0} \ (i \geq 2). \tag{64}$$

Thus $\hat{f}$ is a sum of N width-one shallow ϕ-Nets from $\mathbb{R}^d$ to $\mathbb{R}^m$. Corollary A.2 represents each summand by an $\mathrm{LN}[n_s]$-Net, and Lemma 3.1 combines them into one shallow $\mathrm{PLN}[n_s]$-Net of width $n_s N$.

Here we complete the proof. □

A.7 Proof of Theorem 4.2

Theorem 4.2 (Universal approximations of univariate L-Lipschitz continuous functions by shallow PLN-Nets). Given any L-Lipschitz continuous function $\hat{f} \in C([0, 1])$ and every $\epsilon > 0$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ with $n_s \geq 2$ and $N = \lfloor L/(2\epsilon) \rfloor + 1$, such that

$$|f - \hat{f}|_{L^\infty} < \epsilon \tag{65}$$

Proof. Step 1: construct a proper $\bar{f} \in \mathcal{F}(\mathrm{sign}; 1, N)$ to approximate $\hat{f}$, namely $|\bar{f} - \hat{f}|_{L^\infty} < \epsilon$.

We construct a function

$$\bar{f}(x) = \sum_{j=1}^{N} \alpha_j \, \mathrm{sign}(w_j x + b_j), \tag{66}$$

where $N = \lfloor L/(2\epsilon) \rfloor + 1$. For $1 \leq j \leq N - 1$, we set

$$\alpha_j = \frac{1}{2} \left[\hat{f}\left(\frac{2j+1}{2N} \right) - \hat{f}\left(\frac{2j-1}{2N} \right) \right]; \tag{67}$$

and for $j = N$,

$$\alpha_N = \frac{1}{2} \left[\hat{f}\left(\frac{1}{2N} \right) + \hat{f}\left(\frac{2N-1}{2N} \right) \right]. \tag{68}$$

Moreover, we take $w_j = 1$ for all $1 \leq j \leq N$, $b_j = -\dfrac{j}{N}$ for $1 \leq j \leq N - 1$, and $b_N = 1$.

With this construction, we can verify the following:

1. For $\dfrac{j-1}{N} < x < \dfrac{j}{N}$ with $1 \leq j \leq N$,

$$\bar{f}(x) = \hat{f}\left(\frac{2j-1}{2N} \right). \tag{69}$$

2. For $x = \dfrac{j}{N}$ with $1 \le j \le N-1$,

$$\bar{f}(x) = \frac{1}{2}\left[\hat{f}\left(\frac{2j-1}{2N}\right) + \hat{f}\left(\frac{2j+1}{2N}\right)\right]. \tag{70}$$

3. Additionally, $\bar{f}(0) = \hat{f}\left(\dfrac{1}{2N}\right)$ and $\bar{f}(1) = \hat{f}\left(\dfrac{2N-1}{2N}\right)$.

Since $N = \lfloor L/(2\epsilon) \rfloor + 1$, we have $N > L/(2\epsilon)$. Consequently, we obtain the following error estimates:

1. For $x = 0$,

$$|\bar{f}(0) - \hat{f}(0)| = \left|\hat{f}(0) - \hat{f}\left(\frac{1}{2N}\right)\right| \quad \le \frac{L}{2N} < \epsilon. \tag{71}$$

2. For $x = 1$,

$$|\bar{f}(1) - \hat{f}(1)| = \left|\hat{f}(1) - \hat{f}\left(\frac{2N-1}{2N}\right)\right| \quad \le \frac{L}{2N} < \epsilon. \tag{72}$$

3. For $x = \dfrac{j}{N}$ with $1 \le j \le N-1$,

$$\begin{aligned}\left|\bar{f}\left(\frac{j}{N}\right) - \hat{f}\left(\frac{j}{N}\right)\right| &= \left|\frac{1}{2}\left[\hat{f}\left(\frac{2j-1}{2N}\right) + \hat{f}\left(\frac{2j+1}{2N}\right)\right] - \hat{f}\left(\frac{j}{N}\right)\right| \\ &\le \frac{1}{2}\left|\hat{f}\left(\frac{2j-1}{2N}\right) - \hat{f}\left(\frac{j}{N}\right)\right| + \frac{1}{2}\left|\hat{f}\left(\frac{2j+1}{2N}\right) - \hat{f}\left(\frac{j}{N}\right)\right| \\ &\le \frac{L}{4N} + \frac{L}{4N} = \frac{L}{2N} < \epsilon.\end{aligned} \tag{73}$$

4. For $\dfrac{j-1}{N} < x < \dfrac{j}{N}$ with $1 \le j \le N$,

$$|\bar{f}(x) - \hat{f}(x)| = \left|\hat{f}\left(\frac{2j-1}{2N}\right) - \hat{f}(x)\right| \quad \le L\left|\frac{2j-1}{2N} - x\right| < \frac{L}{2N} < \epsilon. \tag{74}$$

In summary, for every $x \in [0,1]$, one of the above four cases applies, and we have

$$|\bar{f}(x) - \hat{f}(x)|_{L^\infty([0,1])} < \epsilon. \tag{75}$$

Thus, the constructed function $\bar{f}(x) = \displaystyle\sum_{j=1}^{N} \alpha_j \operatorname{sign}(x + b_j)$ uniformly approximates $\hat{f}$ on $[0,1]$ with error less than ϵ.

Step 2: by Corollary 4.1 and Lemma 4.1, since $\bar{f} \in \mathcal{F}(\mathrm{sign}; 1, N)$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$, such that $f = \bar{f}$.

Step 3: Since $f = \bar{f}$ and $|\bar{f}(x) - \hat{f}(x)|_{L^\infty([0,1])} < \epsilon$, we obtain $|f - \hat{f}|_{L^\infty} < \epsilon$.

Here we complete the proof. □

A.8 Stable Layer Normalization with a Numerical Constant

The explicit width bound in Theorem 4.2 uses the idealized zero-variance convention, because it represents the sign function exactly. We now consider the standard stabilized operator

$$[\mathrm{LN}_\varepsilon(\mathbf{h})]_j = \frac{h_j - \mu}{\sqrt{\sigma^2 + \varepsilon}}, \qquad \varepsilon > 0, \tag{76}$$

and show that it can still represent the smooth activation $\phi(z) = z/\sqrt{1+z^2}$ exactly. We use $\mathrm{PLN}_\varepsilon[n_s]$ to denote the direct sum of $\mathrm{LN}_\varepsilon[n_s]$ blocks.

Lemma A.4 (Exact smooth activation under stabilized LN)**.** *Let $\phi(z) = z/\sqrt{1+z^2}$ and fix $\varepsilon > 0$. For every $n_s \geq 3$, any width-one shallow ϕ-Net can be represented exactly by a width-n_s shallow $\mathrm{LN}_\varepsilon[n_s]$-Net.*

Proof. Let a width-one ϕ-Net be

$$\hat{f}(\mathbf{x}) = \alpha\phi(\boldsymbol{w}^\top \mathbf{x} + b) + \beta. \tag{77}$$

We construct the corresponding stabilized LN-Net in three steps.

Step 1: Construct the first affine layer. For $n_s \geq 3$, take

$$\mathbf{a} = \sqrt{\frac{n_s}{2}}(1, -1, 0, \ldots, 0)^\top, \qquad \mathbf{c} = \sqrt{\frac{n_s}{6}}(1, 1, -2, 0, \ldots, 0)^\top. \tag{78}$$

They satisfy

$$\mathbf{1}^\top \mathbf{a} = \mathbf{1}^\top \mathbf{c} = 0, \qquad \mathbf{a}^\top \mathbf{c} = 0, \qquad \|\mathbf{a}\|_2^2 = \|\mathbf{c}\|_2^2 = n_s. \tag{79}$$

Set $A = \sqrt{1+\varepsilon}$ and $z = \boldsymbol{w}^\top \mathbf{x} + b$. The first affine layer outputs

$$\mathbf{h}(z) = A\mathbf{a}z + \mathbf{c}. \tag{80}$$

Step 2: Apply stabilized LN. Since both $\mathbf{a}$ and $\mathbf{c}$ are centered, the mean of $\mathbf{h}(z)$ is zero. Moreover, its variance is

$$\sigma^2 = \frac{1}{n_s}\|A\mathbf{a}z + \mathbf{c}\|_2^2 = A^2 z^2 + 1. \tag{81}$$

Let $\mathbf{u} = \mathbf{a}/n_s$. By the orthogonality of $\mathbf{a}$ and $\mathbf{c}$, we obtain

$$\mathbf{u}^\top \mathrm{LN}_\varepsilon(\mathbf{h}(z)) = \frac{Az}{\sqrt{A^2 z^2 + 1 + \varepsilon}} = \frac{Az}{A\sqrt{z^2+1}} = \phi(z). \tag{82}$$

Step 3: Construct the output affine layer. The output layer applies $\alpha\mathbf{u}^\top$ and adds β. Therefore,

$$\alpha\mathbf{u}^\top \mathrm{LN}_\varepsilon(\mathbf{h}(\boldsymbol{w}^\top \mathbf{x} + b)) + \beta = \alpha\phi(\boldsymbol{w}^\top \mathbf{x} + b) + \beta = \hat{f}(\mathbf{x}). \tag{83}$$

Thus, the stabilized LN-Net represents the width-one ϕ-Net exactly. This completes the proof. □

We next extend the construction from one activation unit to shallow and deep networks. Applying Lemma A.4 independently to N activation units and arranging the resulting LN blocks in parallel gives

$$\mathcal{F}(\phi; 1, N) \subseteq \mathcal{F}(\mathrm{PLN}_\varepsilon[n_s]; 1, n_s N), \qquad n_s \geq 3. \tag{84}$$

For a deep ϕ-Net, we apply the same construction to every activation layer and combine the adjacent affine layers, as in the proof of Lemma 4.3. Hence the corresponding inclusion also holds for arbitrary depth.

Since ϕ is smooth and non-polynomial, standard shallow-network universal-approximation results (Hornik, 1991; Pinkus, 1999) imply that stabilized PLN-Nets are dense on compact domains. The multivariate approximation rate, the deep-network approximation result, and the position-wise sequence and convolutional representation results are all transferred through this exact ϕ-representation. Therefore, these results continue to hold under the stabilized convention.

The stabilized operator is continuous and does not represent the discontinuous sign function exactly. Consequently, this argument does not assert the explicit width bound $N = \lfloor L/(2\epsilon) \rfloor + 1$ from Theorem 4.2 for stabilized LN. Learnable scale and shift parameters do not alter the conclusion because they can be absorbed into the following affine layer.

B Additional Mathematical Proofs in Section 4

B.1 Proof of Lemma 4.3

Lemma 4.3 Given any $\hat{f} \in \mathcal{F}(\phi; L, N)$ with $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$, there exists a PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; L, n_s N)$ with $n_s \geq 3$ such that $f = \hat{f}$. In other words, we have

$$\mathcal{F}(\phi; L, N) \subseteq \mathcal{F}(\mathrm{PLN}[n_s]; L, n_s N), \tag{85}$$

where $\phi(x) = x/\sqrt{x^2+1}$ is an activation function.

Proof. Assume $\hat{f} : \mathbb{R}^d \to \mathbb{R}^m$. By Corollary A.3, we obtain that

$$\mathcal{F}(\phi; 1, N) \subseteq \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N). \tag{86}$$

Now we consider to extend to the deep networks.

Given a $\hat{f} \in \mathcal{F}(\phi; L, N)$, which can be unrolled as

$$\begin{aligned} \hat{f} &= W_{L+1} \circ \phi_L \circ W_L \circ \cdots \circ W_2 \circ \phi_1 \circ W_1 \\ &= W_{L+1} \circ \phi_L \circ I_L \circ W_L \circ \cdots \circ W_2 \circ \phi_1 \circ I_1 \circ W_1 \\ &= (W_{L+1} \circ \phi_L \circ I_L) \circ W_L \circ \cdots \circ (W_2 \circ \phi_1 \circ I_1 \circ W_1). \end{aligned} \tag{87}$$

where $W_1 : \mathbb{R}^{d_x} \to \mathbb{R}^{d_1}$, $W_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ for $\ell = 2, \ldots, L$, and $W_{L+1} : \mathbb{R}^{d_L} \to \mathbb{R}^{d_y}$ are affine layers. $I_\ell : \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_\ell}$ for $\ell = 1, \ldots, L$ are identity maps.

Now we construct that $\hat{f}_\ell = W_{\ell+1} \circ \phi_\ell \circ I_\ell$ for $\ell = 2, \ldots, L$ and $\hat{f}_1 = W_2 \circ \phi_1 \circ I_1 \circ W_1$. We find that each $\hat{f}_\ell \in \mathcal{F}(\phi; 1, d_\ell)$ and

$$\hat{f} = \hat{f}_L \circ \hat{f}_{L-1} \circ \cdots \circ \hat{f}_1. \tag{88}$$

By Eqn.86, we can find $f_\ell \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s d_\ell)$ such that $f_\ell = \hat{f}_\ell$. Set $f_\ell = V_\ell \circ \mathrm{PLN} \circ V'_\ell$, then we construct

$$\begin{aligned} f &= f_L \circ f_{L-1} \circ \cdots \circ f_1 \\ &= (V_L \circ \mathrm{PLN} \circ V'_L) \circ (V_{L-1} \circ \mathrm{PLN} \circ V'_{L-1}) \circ \cdots \circ (V_1 \circ \mathrm{PLN} \circ V'_1) \\ &= V_L \circ \mathrm{PLN} \circ V'_L \circ V_{L-1} \circ \mathrm{PLN} \circ V'_{L-1} \circ \cdots \circ V_1 \circ \mathrm{PLN} \circ V'_1 \\ &= \bar{V}_{L+1} \circ \mathrm{PLN} \circ \bar{V}_L \circ \cdots \circ \mathrm{PLN} \circ \bar{V}_1 \end{aligned} \tag{89}$$

where $\bar{V}_1 = V'_1 : \mathbb{R}^{d_x} \to \mathbb{R}^{n_s d_1}$, $\bar{V}_\ell = V'_\ell \circ V_{\ell-1} : \mathbb{R}^{n_s d_{\ell-1}} \to \mathbb{R}^{n_s d_\ell}$ for $\ell = 2, \ldots, L$, and $\bar{V}_{L+1} = V_L : \mathbb{R}^{n_s d_L} \to \mathbb{R}^{d_y}$ are affine layers.

Thus f is a PLN-Net of depth L. Let $D = \max\{d_1, \ldots, d_L\} \le N$ be the actual width of $\hat{f}$. The width of f is $\max\{n_s d_1, \ldots, n_s d_L\} = n_s D \le n_s N$, so $f \in \mathcal{F}(\mathrm{PLN}[n_s]; L, n_s N)$. Since each $f_\ell = \hat{f}_\ell$, we also have $f = \hat{f}$, completing the proof for the stated vector input and output dimensions. □

B.2 Proof of Theorem 4.3

We first recall the definition of Sobolev space.

Sobolev space $\mathcal{W}^{s,\infty}(\Omega)$. Let $\Omega \subset \mathbb{R}^n$ be a bounded Lipschitz domain, $s \in \mathbb{N}$. For a multi-index $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$, let $|\alpha| = \sum_{i=1}^{n} \alpha_i$ and $D^\alpha u = \partial^{|\alpha|} u / \partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}$ denote weak derivatives.

The Sobolev space is defined as

$$\mathcal{W}^{s,\infty}(\Omega) = \big\{ u \in L^\infty(\Omega) \mid D^\alpha u \in L^\infty(\Omega) \; \forall \, |\alpha| \le s \big\}, \tag{90}$$

with norm

$$\|u\|_{\mathcal{W}^{s,\infty}(\Omega)} = \max_{|\alpha| \le s} \|D^\alpha u\|_{L^\infty(\Omega)}. \tag{91}$$

Theorem 4.3 Let $d \ge 2$, $s \ge 1$, and let $\hat{f} : B^d \to \mathbb{R}$ belong to $\mathcal{W}^{s,\infty}(B^d)$ with $\|\hat{f}\|_{\mathcal{W}^{s,\infty}} \le 1$, where $B^d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \le 1\}$. For every $N \ge 1$, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ with $n_s \ge 3$ such that

$$\|f - \hat{f}\|_{L^\infty(B^d)} \le C N^{-s/d}, \tag{92}$$

for some C independent of N.

Theorem 4.3 follows from Pinkus (1999, Theorem 6.8). Here we fix $p = \infty$ and rewrite the result as follows.

Theorem B.1 (Pinkus, 1999, Theorem 6.8)**.** *Assume $\sigma : \mathbb{R} \to \mathbb{R}$ is such that $\sigma \in C^\infty(\Theta)$ on some open interval Θ, and σ is not a polynomial on Θ. Then for every $m \geq 1$ and $n \geq 2$,*

$$\sup_{\hat{f} \in \mathcal{B}^m_\infty} \inf_{f \in \mathcal{F}(\sigma;1,r)} |f - \hat{f}|_{L^\infty(B^n)} \leq C r^{-m/n} \tag{93}$$

for some constant C independent of r, where

$$\mathcal{B}^m_\infty = \{f \in W^{m,\infty}(B^n) : \|f\|_{W^{m,\infty}} \leq 1\}, B^n = \{\mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\| \leq 1\}. \tag{94}$$

Here we give the proof of Theorem 4.3.

Proof. Note that $\phi(x) = x/\sqrt{x^2+1}$ is such that $\phi \in C^\infty(-1,1)$ on some open interval Θ, and ϕ is not a polynomial on $(-1,1)$.

Given $\hat{f} \in \mathcal{W}^{s,\infty}(B^d)$ and $\|\hat{f}\|_{\mathcal{W}^{s,\infty}} \leq 1$, by Theorem B.1, there exists some $\bar{f} \in \mathcal{F}(\phi; 1, N)$ such that

$$\|\bar{f} - \hat{f}\|_{L^\infty(B^d)} \leq C N^{-s/d}. \tag{95}$$

By Corollary 4.1, there exists a shallow PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 1, n_s N)$ such that $f = \bar{f}$.

Therefore, $\|f - \hat{f}\|_{L^\infty(B^d)} \leq C N^{-s/d}$, which completes the proof. □

B.3 Proof of Theorem 4.4

Theorem 4.4 (Approximate ReLU networks by PLN-Nets.) Let $\hat{f} \in \mathcal{F}(\mathrm{ReLU}; L, N)$ with $\hat{f} : [0,1]^d \to \mathbb{R}^n$, and let $\epsilon > 0$. Then there exists a PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 2L, 3n_s N)$ with $n_s \geq 3$ such that

$$\|f - \hat{f}\|_{L^\infty([0,1]^d)} < \epsilon. \tag{96}$$

Theorem 4.4 follows from Zhang et al. (2024, Theorem 1).

Theorem B.2 (Zhang et al., 2024, Theorem 1)**.** *Suppose $\rho \in \mathscr{A}$ and $\hat{f}_{\mathrm{ReLU}} \in \mathcal{F}(\mathrm{ReLU}; L, N) : \mathbb{R}^d \to \mathbb{R}^n$. Then for any $\epsilon > 0$ and $A > 0$, there exists $f_\rho \in \mathcal{F}(\rho; 2L, 3N) : \mathbb{R}^d \to \mathbb{R}^n$ such that*

$$\|f_\rho - \hat{f}_{\mathrm{ReLU}}\|_{\sup([-A,A]^d)} < \epsilon. \tag{97}$$

In the notation of Zhang et al. (2024), $\mathscr{A}_3 \subset \mathscr{A}$, where

$$\mathscr{A}_3 = \{\rho : \mathbb{R} \to \mathbb{R} | \rho \in \mathscr{S}, \exists x_0 \in \mathbb{R}, \rho''(x_0) \neq 0\}, \tag{98}$$

where

$$\mathscr{S} = \{h : \mathbb{R} \to \mathbb{R} \mid \sup_{x \in \mathbb{R}} |h(x)| < \infty,\ h(+\infty), h(-\infty) \in \mathbb{R},\ h(+\infty) \neq h(-\infty)\}. \tag{99}$$

Now we prove Theorem 4.4.

Proof. Let $\hat{f} \in \mathcal{F}(\mathrm{ReLU}; L, N)$. For $\phi(x) = x/\sqrt{x^2+1}$, we have $\phi(-\infty) = -1 \neq \phi(+\infty) = 1$ and $\sup_{x \in \mathbb{R}} |\phi(x)| = 1$, so $\phi \in \mathscr{S}$. Moreover, $\phi''(1) \neq 0$, hence $\phi \in \mathscr{A}_3 \subset \mathscr{A}$.

Therefore, by Theorem B.2, there exists $\bar{f} \in \mathcal{F}(\phi; 2L, 3N)$ such that $\|\bar{f} - \hat{f}\|_{\sup([-1,1]^d)} < \epsilon$. By Lemma 4.3, there exists a PLN-Net $f \in \mathcal{F}(\mathrm{PLN}[n_s]; 2L, 3n_s N)$ with $n_s \geq 3$ such that $f = \bar{f}$. Furthermore, we obtain

$$\|f - \hat{f}\|_{L^\infty([0,1]^d)} \leq \|f - \hat{f}\|_{L^\infty([-1,1]^d)} \leq \|f - \hat{f}\|_{\sup([-1,1]^d)} < \epsilon. \tag{100}$$

Here we complete the proof. □

C MATHEMATICAL PROOFS IN SECTION 5

C.1 PROOF OF THEOREM 5.1

Theorem 5.1 (Exact representation of position-wise FFNs by PLN). Let $\hat{\varphi}_1$ and $\hat{\varphi}_2$ be two sequence linear layers mapping token dimensions $d_x \to N$ and $N \to d_y$, respectively, and let $\phi(x) = x/\sqrt{x^2+1}$. Define the position-wise FFN mapping $\hat{f} = \hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1$. For every $n_s \geq 3$, there exist sequence linear layers φ_1 and φ_2 mapping $d_x \to n_s N$ and $n_s N \to d_y$, respectively, such that

$$f = \varphi_2 \circ \mathrm{PLN}[n_s] \circ \varphi_1 \tag{101}$$

exactly represents $\hat{f}$ for all sequence inputs.

Proof. Assume that $\hat{\varphi}_1(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{W} + \mathbf{1}_m \boldsymbol{b}^\top$.

Note that for $\boldsymbol{X}$, we separate it into $\boldsymbol{X}^\top = [\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_s]^\top$ and obtain that

$$\begin{aligned}
\hat{\varphi}_1(\boldsymbol{X}) &= \boldsymbol{X}\boldsymbol{W} + \mathbf{1}_m \boldsymbol{b}^\top \\
&= \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_s^\top \end{bmatrix} \boldsymbol{W} + \begin{bmatrix} \boldsymbol{b}^\top \\ \vdots \\ \boldsymbol{b}^\top \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{x}_1^\top \boldsymbol{W} + \boldsymbol{b}^\top \\ \vdots \\ \mathbf{x}_s^\top \boldsymbol{W} + \boldsymbol{b}^\top \end{bmatrix} \\
&= \begin{bmatrix} \hat{\varphi}_1(\mathbf{x}_1^\top) \\ \vdots \\ \hat{\varphi}_1(\mathbf{x}_s^\top) \end{bmatrix}.
\end{aligned} \tag{102}$$

Such calculation is also suitable for $\hat{\varphi}_2$ Since ϕ activates the neurons element-wisely, we can obtain that

$$[\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\boldsymbol{X}) = \begin{bmatrix} [\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\mathbf{x}_1^\top) \\ \vdots \\ [\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\mathbf{x}_s^\top) \end{bmatrix} \tag{103}$$

By Corollary 4.1, we can find some $\varphi_1 : \mathbf{x}^\top \to \mathbf{x}^\top \boldsymbol{W}_1 + \boldsymbol{b}_1^\top$ and $\varphi_2 : \mathbf{x}^\top \to \mathbf{x}^\top \boldsymbol{W}_2 + \boldsymbol{b}_2^\top$, such that

$$[\varphi_2 \circ \mathrm{PLN} \circ \varphi_1](\mathbf{x}^\top) = [\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\mathbf{x}^\top), \forall \mathbf{x} \in \mathbb{R}^d. \tag{104}$$

Applying Corollary 4.1 is suitable, for we can identify it by transposing the matrices in the calculation process.

Similar to Eqn.103, we can identify that

$$\begin{bmatrix} \varphi_1(\mathbf{x}_1^\top) \\ \vdots \\ \varphi_1(\mathbf{x}_s^\top) \end{bmatrix} = \varphi_1(\boldsymbol{X}) \tag{105}$$

holds for any $\boldsymbol{X} \in \mathbb{R}^{s \times d}$, and the similar conclusion holds for φ_2.

Set $\boldsymbol{H} = \varphi_1(\boldsymbol{X})$, we figure out that

$$\mathrm{PLN}(\boldsymbol{H}) = \begin{bmatrix} \mathrm{PLN}(\mathbf{h}_1^\top) \\ \vdots \\ \mathrm{PLN}(\mathbf{h}_s^\top) \end{bmatrix} = \begin{bmatrix} \mathrm{PLN}(\varphi_1(\mathbf{x}_1^\top)) \\ \vdots \\ \mathrm{PLN}(\varphi_1(\mathbf{x}_s^\top)) \end{bmatrix}. \tag{106}$$

Like activation functions, PLN can also act on different dimensionalities—PLN out of the matrix acts on $\mathbb{R}^{s \times n_s N}$, while the one in the matrix acts on $\mathbb{R}^{n_s N}$. Both of them have the same norm size n_s.

Furthermore, we figure out that

$$[\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\boldsymbol{X}) = \begin{bmatrix} [\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\mathbf{x}_1^\top) \\ \vdots \\ [\hat{\varphi}_2 \circ \phi \circ \hat{\varphi}_1](\mathbf{x}_s^\top) \end{bmatrix} = \begin{bmatrix} [\varphi_2 \circ \mathrm{PLN} \circ \varphi_1](\mathbf{x}_1^\top) \\ \vdots \\ [\varphi_2 \circ \mathrm{PLN} \circ \varphi_1](\mathbf{x}_s^\top) \end{bmatrix} = [\varphi_2 \circ \mathrm{PLN} \circ \varphi_1](\boldsymbol{X}). \tag{107}$$

Here we complete the proof. □

C.2 Proof of Theorem 5.2

Auxiliary pointwise convolutions. Let $\mathbf{X} \in \mathbb{R}^{C_{\mathrm{in}} \times H \times W}$ denote an image feature map. A pointwise convolution φ^{pos} mapping the channel dimension $C_{\mathrm{in}} \to C_{\mathrm{out}}$ is defined as

$$[\varphi^{\mathrm{pos}}(\mathbf{X})]_{:,i,j} = \boldsymbol{W}\mathbf{X}_{:,i,j} + \boldsymbol{b}, \tag{108}$$

for every spatial position (i, j), where $\boldsymbol{W} \in \mathbb{R}^{C_{\mathrm{out}} \times C_{\mathrm{in}}}$ and $\boldsymbol{b} \in \mathbb{R}^{C_{\mathrm{out}}}$ are shared across all spatial positions. This is the standard 1×1 convolution: it mixes channels independently at each position without changing the spatial receptive field.

Lemma C.1 (Folding pointwise convolutions into general convolutions)**.** *Let $\mathcal{C}$ be the affine convolutional layer in Eqn.14. Let $\varphi_1^{\mathrm{pos}}(\mathbf{z}) = \boldsymbol{P}\mathbf{z} + \mathbf{c}$ and $\varphi_2^{\mathrm{pos}}(\mathbf{z}) = \boldsymbol{Q}\mathbf{z}$ be pointwise convolutions, where the latter is bias-free. Then there exists an affine convolutional layer $\mathcal{C}'$ with the same kernel support, stride, dilation, and padding as $\mathcal{C}$ such that*

$$\mathcal{C}' = \varphi_1^{\mathrm{pos}} \circ \mathcal{C} \circ \varphi_2^{\mathrm{pos}}. \tag{109}$$

In particular, its kernel slices and bias are

$$K'_\delta = \boldsymbol{P}K_\delta\boldsymbol{Q}, \qquad \mathbf{b}' = \boldsymbol{P}\mathbf{b} + \mathbf{c}. \tag{110}$$

Proof of Lemma C.1.

Proof. Because the two pointwise convolutions share their parameters across spatial positions, for every input tensor $\mathbf{X}$ and output position (i, j),

$$\begin{aligned} &[\varphi_1^{\mathrm{pos}} \circ \mathcal{C} \circ \varphi_2^{\mathrm{pos}}(\mathbf{X})]_{:,i,j} \\ &\quad = \boldsymbol{P}\left[\mathbf{b} + \sum_{(u,v)\in S} K_{u,v}\boldsymbol{Q}\widetilde{\mathbf{X}}_{:,\, s_h i + d_h u - p_h,\, s_w j + d_w v - p_w}\right] + \mathbf{c} \\ &\quad = (\boldsymbol{P}\mathbf{b} + \mathbf{c}) + \sum_{(u,v)\in S} (\boldsymbol{P}K_{u,v}\boldsymbol{Q})\widetilde{\mathbf{X}}_{:,\, s_h i + d_h u - p_h,\, s_w j + d_w v - p_w}. \end{aligned} \tag{111}$$

This is an affine convolution with kernel slices $K'_{u,v} = \boldsymbol{P}K_{u,v}\boldsymbol{Q}$ and bias $\mathbf{b}' = \boldsymbol{P}\mathbf{b} + \mathbf{c}$. Zero padding is preserved because the pre-composed pointwise convolution is bias-free; the stride and dilation indices are unchanged. □

A scalar identity. Let $n = n_s \geq 3$. Choose $\mathbf{a}, \mathbf{c} \in \mathbb{R}^n$ as

$$\mathbf{a} = \sqrt{\frac{n}{2}}(1, -1, 0, \ldots, 0)^\top, \qquad \mathbf{c} = \sqrt{\frac{n}{6}}(1, 1, -2, 0, \ldots, 0)^\top, \tag{112}$$

and set $\mathbf{u} = \mathbf{a}/n$. These vectors satisfy

$$\mathbf{1}^\top\mathbf{a} = \mathbf{1}^\top\mathbf{c} = 0, \qquad \mathbf{a}^\top\mathbf{c} = 0, \qquad \|\mathbf{a}\|_2^2 = \|\mathbf{c}\|_2^2 = n. \tag{113}$$

For every $t \in \mathbb{R}$, the vector $\mathbf{a}t + \mathbf{c}$ has mean zero and variance $t^2 + 1$. Consequently,

$$\mathbf{u}^\top \mathrm{LN}(\mathbf{a}t + \mathbf{c}) = \frac{t}{\sqrt{1 + t^2}} = \phi(t). \tag{114}$$

A pointwise-convolutional representation of ϕ. For any channel dimension C, define two pointwise (1×1) convolutions $\varphi^{\mathrm{pos}}_{1,C}$ and $\varphi^{\mathrm{pos}}_{2,C}$ by

$$[\varphi^{\mathrm{pos}}_{1,C}(\mathbf{Z})]_{r,g}(p) = a_g \mathbf{Z}_r(p) + c_g, \tag{115}$$

$$[\varphi^{\mathrm{pos}}_{2,C}(\mathbf{G})]_r(p) = \sum_{g=1}^{n} u_g \mathbf{G}_{r,g}(p), \tag{116}$$

where $1 \le r \le C$ and $1 \le g \le n$. The first layer maps $C \to nC$ channels and the second maps $nC \to C$ channels. By Eqn.114, for every tensor $\mathbf{Z}$, channel r, and spatial position p,

$$\begin{aligned}&[\varphi^{\mathrm{pos}}_{2,C} \circ \mathrm{CPLN}_n \circ \varphi^{\mathrm{pos}}_{1,C}(\mathbf{Z})]_r(p) \\ &= \sum_{g=1}^{n} u_g \big[\mathrm{LN}(\mathbf{a}\mathbf{Z}_r(p) + \mathbf{c})\big]_g = \phi(\mathbf{Z}_r(p)).\end{aligned} \tag{117}$$

Hence, as position-wise tensor maps,

$$\phi = \varphi^{\mathrm{pos}}_{2,C} \circ \mathrm{CPLN}_n \circ \varphi^{\mathrm{pos}}_{1,C}. \tag{118}$$

Theorem 5.2 (Exact representation of convolutional blocks by CPLN). Let $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$ be arbitrary affine convolutional layers mapping channel dimensions $C_x \to N$ and $N \to C_y$, respectively, and let $\phi(t) = t/\sqrt{1+t^2}$ act element-wise. For every norm size $n_s \ge 3$, there exist affine convolutional layers $\mathcal{C}_1$ and $\mathcal{C}_2$ with the same kernel supports, strides, dilations, and padding as $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$, and intermediate width $n_s N$, such that

$$\mathcal{C}_2 \circ \mathrm{CPLN}_{n_s} \circ \mathcal{C}_1 = \hat{\mathcal{C}}_2 \circ \phi \circ \hat{\mathcal{C}}_1 \tag{119}$$

for every input tensor.

Proof. Apply Eqn.118 with $C = N$:

$$\hat{\mathcal{C}}_2 \circ \phi \circ \hat{\mathcal{C}}_1 = \hat{\mathcal{C}}_2 \circ \varphi^{\mathrm{pos}}_{2,N} \circ \mathrm{CPLN}_n \circ \varphi^{\mathrm{pos}}_{1,N} \circ \hat{\mathcal{C}}_1. \tag{120}$$

Define

$$\mathcal{C}_1 = \varphi^{\mathrm{pos}}_{1,N} \circ \hat{\mathcal{C}}_1, \qquad \mathcal{C}_2 = \hat{\mathcal{C}}_2 \circ \varphi^{\mathrm{pos}}_{2,N}. \tag{121}$$

The layer $\varphi^{\mathrm{pos}}_{2,N}$ is bias-free. Therefore, Lemma C.1 shows that $\mathcal{C}_1$ and $\mathcal{C}_2$ are affine convolutions with the same spatial specifications as $\hat{\mathcal{C}}_1$ and $\hat{\mathcal{C}}_2$, respectively. Thus

$$\mathcal{C}_2 \circ \mathrm{CPLN}_n \circ \mathcal{C}_1 = \hat{\mathcal{C}}_2 \circ \phi \circ \hat{\mathcal{C}}_1 \tag{122}$$

for every input tensor. □

C.3 Extension to Deep Convolutional Networks

Deep convolutional representation. For every $n_s \ge 3$, replacing each element-wise ϕ by its CPLN realization and folding the intervening pointwise 1×1 convolutions into the adjacent convolutional layers converts any depth-L ϕ-activated convolutional network with hidden channel dimensions $C_1, \dots, C_L$ into an exactly equivalent CPLN network with dimensions $n_s C_1, \dots, n_s C_L$, without changing its depth or spatial specifications.

Proof. Write the original network as

$$\hat{F} = \hat{\mathcal{C}}_{L+1} \circ \phi \circ \hat{\mathcal{C}}_L \circ \cdots \circ \phi \circ \hat{\mathcal{C}}_1. \tag{123}$$

For each hidden channel dimension C_ℓ, let $\varphi^{\mathrm{pos}}_{1,C_\ell}$ and $\varphi^{\mathrm{pos}}_{2,C_\ell}$ be the two layers in Equations (115) and (116). Define

$$\mathcal{C}_1 = \varphi^{\mathrm{pos}}_{1,C_1} \circ \hat{\mathcal{C}}_1, \tag{124}$$

$$\mathcal{C}_\ell = \varphi^{\mathrm{pos}}_{1,C_\ell} \circ \hat{\mathcal{C}}_\ell \circ \varphi^{\mathrm{pos}}_{2,C_{\ell-1}}, \qquad 2 \le \ell \le L, \tag{125}$$

$$\mathcal{C}_{L+1} = \hat{\mathcal{C}}_{L+1} \circ \varphi^{\mathrm{pos}}_{2,C_L}. \tag{126}$$

By Lemma C.1, every $\mathcal{C}_\ell$ is an affine convolution with the same spatial specifications as $\hat{\mathcal{C}}_\ell$. Using associativity of composition and Eqn.118,

$$\begin{aligned} F &= \mathcal{C}_{L+1} \circ \mathrm{CPLN}_n \circ \mathcal{C}_L \circ \cdots \circ \mathrm{CPLN}_n \circ \mathcal{C}_1 \\ &= \hat{\mathcal{C}}_{L+1} \circ \left(\varphi^{\mathrm{pos}}_{2,C_L} \circ \mathrm{CPLN}_n \circ \varphi^{\mathrm{pos}}_{1,C_L}\right) \circ \hat{\mathcal{C}}_L \circ \cdots \\ &\qquad \circ \left(\varphi^{\mathrm{pos}}_{2,C_1} \circ \mathrm{CPLN}_n \circ \varphi^{\mathrm{pos}}_{1,C_1}\right) \circ \hat{\mathcal{C}}_1 \\ &= \hat{\mathcal{C}}_{L+1} \circ \phi \circ \hat{\mathcal{C}}_L \circ \cdots \circ \phi \circ \hat{\mathcal{C}}_1 = \hat{F}. \end{aligned} \tag{127}$$

The equality holds for every input tensor, and each hidden channel C_ℓ is replaced by $n_s C_\ell$ channels. □

D Additional Experimental Details

Unless stated otherwise, all experiments are implemented in PyTorch (Paszke et al., 2019). The normalization-domain operators and experiment grids are our implementations; below we cite the external datasets, architectures, training recipes, and evaluation tools on which individual protocols rely. The sections follow the order of Section 6: MLP function approximation and PINNs, followed by CNNs and ViTs. The VGG-16 initialization study is included with the CNN experiments as an auxiliary comparison.

D.1 MLP Experiments

D.1.1 Dense Fourier Function Approximation

For each target seed s and dimension $d \in \{4, 8\}$, we generate

$$f_{s,d}(\mathbf{x}) = 8^{-1/2} \sum_{j=1}^{8} a_{s,j} \sin(\pi \boldsymbol{\omega}_{s,j}^\top \mathbf{x} + \phi_{s,j}), \qquad \mathbf{x} \in [-1, 1]^d, \tag{128}$$

where $a_{s,j} \in \{-1, 1\}$, $\boldsymbol{\omega}_{s,j} \in \{-1, 1\}^d$, and $\phi_{s,j}$ are independently sampled and then fixed. We use five target seeds and standardize each target using its training values. We independently sample 10,000, 2,000, and 10,000 inputs for training, validation, and testing. Every Fourier mode involves all d coordinates, making the $d = 8$ case a denser-interaction stress test rather than merely adding irrelevant input coordinates.

All MLPs have width 1024 and either one or four hidden layers. We compare Identity, LN, PLN[4], PLN[8], ReLU, SiLU, Tanh, and $\mathrm{SAT}(x) = x/\sqrt{1 + x^2}$. LN and PLN use learnable affine parameters, and no model contains an additional normalization layer. Each run uses Adam (Kingma & Ba, 2014) for 10,000 updates with batch size 256 and zero weight decay. The learning rate is linearly warmed up for 500 updates and then cosine-decayed to zero. We evaluate $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ and select one shared learning rate for each dimension–method–depth configuration by its mean validation MSE over all five targets and three model seeds. We first average the three model seeds within each target, then report the mean and sample standard deviation across the five targets in Table I.

Table I: Dense Fourier test MSE (mean ± standard deviation across five target functions, after averaging three model seeds). Lower is better. PLN gives the lowest error among all shallow networks in both dimensions. Identity denotes no nonlinearity.

	$d = 4$		$d = 8$	
Nonlinearity	Depth 1	Depth 4	Depth 1	Depth 4
Identity	0.9982 ± 0.0071	0.9979 ± 0.0072	0.9989 ± 0.0159	0.9972 ± 0.0144
LN	0.9979 ± 0.0071	0.8793 ± 0.0885	0.9990 ± 0.0158	0.9808 ± 0.0099
PLN[4]	$\mathbf{0.0048 \pm 0.0013}$	0.0018 ± 0.0006	0.1171 ± 0.0197	0.1034 ± 0.0244
PLN[8]	0.0079 ± 0.0030	0.00075 ± 0.00027	$\mathbf{0.0803 \pm 0.0068}$	0.0660 ± 0.0215
ReLU	0.0118 ± 0.0043	0.00081 ± 0.00025	0.1196 ± 0.0082	0.0282 ± 0.0036
SiLU	0.9985 ± 0.0072	$\mathbf{0.00037 \pm 0.00026}$	0.9986 ± 0.0156	$\mathbf{0.0269 \pm 0.0143}$
Tanh	0.3574 ± 0.0425	0.0014 ± 0.0006	0.9384 ± 0.0242	0.0345 ± 0.0075
SAT	0.2302 ± 0.0127	0.0011 ± 0.0003	0.7131 ± 0.0359	0.0335 ± 0.0079

For one hidden layer, PLN is not only trainable but is the best-performing nonlinearity in both dimensions: PLN[4] is best for $d = 4$, while PLN[8] is best for $d = 8$. In contrast, increasing depth substantially improves the element-wise activations, with SiLU giving the lowest four-layer error. These results isolate the intended empirical point: PLN supplies useful multivariate nonlinearity within a shallow block; they do not claim that PLN must outperform deep compositions of scalar activations.

D.1.2 Helmholtz PINN

Following the physics-informed neural-network framework of Raissi et al. (2019), let $u_\theta(x)$ approximate the solution to the one-dimensional Helmholtz equation. Using automatic differentiation, we compute $u''_\theta(x)$ and form the residual

$$r_\theta(x) = -u''_\theta(x) + \pi^2 u_\theta(x) - 2\pi^2 \sin(\pi x). \tag{129}$$

The exact solution is $u(x) = \sin(\pi x)$. On $\Omega = [-1, 1]$, we sample $5{,}000$ interior points and 500 boundary points and jointly minimize the mean squared PDE residual and Dirichlet boundary error. The MLPs have depth 6, width 128 or 256, and no additional normalization. We compare ReLU, Tanh, SAT, and PLN with group sizes $4, 8, 16, 32$ over five model seeds. We evaluate loss weights $(\lambda_{\text{PDE}}, \lambda_{\text{BC}}) \in \{(1, 10), (1, 1), (3, 1)\}$ and use learning rate 3×10^{-4} for both Adam (Kingma & Ba, 2014) and SGD (Amari, 1993), with a step schedule for SGD. Relative $\mathcal{L}_2$ error against the exact solution measures global fitting quality. Table II reports the complete loss-weight breakdown; the best result in each row is bold and the second best is underlined. Because the objective contains a second-order differential operator, this is a practical PINN comparison, not evidence for a derivative-approximation theorem.

Table II: We record the best relative $\mathcal{L}_2$ error of different activation layers under different experiment settings on solving Helmholtz equation.

Width	Loss Weight	ReLU	PLN4	PLN8	PLN16	PLN32	Tanh	SAT
128	[1, 10]	0.999601	**0.000035**	0.001120	0.001176	0.007878	0.000250	<u>0.000104</u>
	[1, 1]	0.995612	**0.000941**	0.002284	<u>0.000970</u>	0.004231	0.003733	0.000979
	[3, 1]	0.993593	0.009553	0.003912	0.004856	0.007465	**0.000440**	<u>0.000525</u>
256	[1,10]	0.998639	**0.000036**	0.004040	0.001631	0.000567	0.000188	<u>0.000052</u>
	[1, 1]	0.998485	0.001416	0.005552	0.001839	0.002643	0.002653	**0.001219**
	[3, 1]	0.999727	0.018451	0.002966	0.002875	0.005476	0.009267	**0.000410**

D.2 CNN Experiments

D.2.1 Mixed-Axis Normalization

We use the full CIFAR-10 training split (Krizhevsky et al., 2009) and evaluate both fixed random labels, following the established random-label fitting protocol (Zhang et al., 2017), and the original labels. The network processes feature maps in $\mathbb{R}^{B\times 128\times 32\times 32}$ using $[\text{Conv}(3\times3)\text{–Norm}]^4$, followed

by 4×4 average pooling and a linear classifier. There is no separate element-wise activation. If G_C is the number of channel groups and each spatial axis is split into K parts, each normalization domain has shape $(128/G_C) \times (32/K) \times (32/K)$.

We evaluate all $8 \times 6 = 48$ choices $G_C \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and $K \in \{1, 2, 4, 8, 16, 32\}$. Every configuration is run with three model seeds for 200 epochs using SGD (Amari, 1993), learning rate 0.01, momentum 0.9, and batch size 128. Weight decay is zero and normalization affine parameters are disabled. For the clean-label experiment, final training and test accuracy are reported; for the random-label experiment, final fitting accuracy is measured on the same fixed 50,000-example training set.

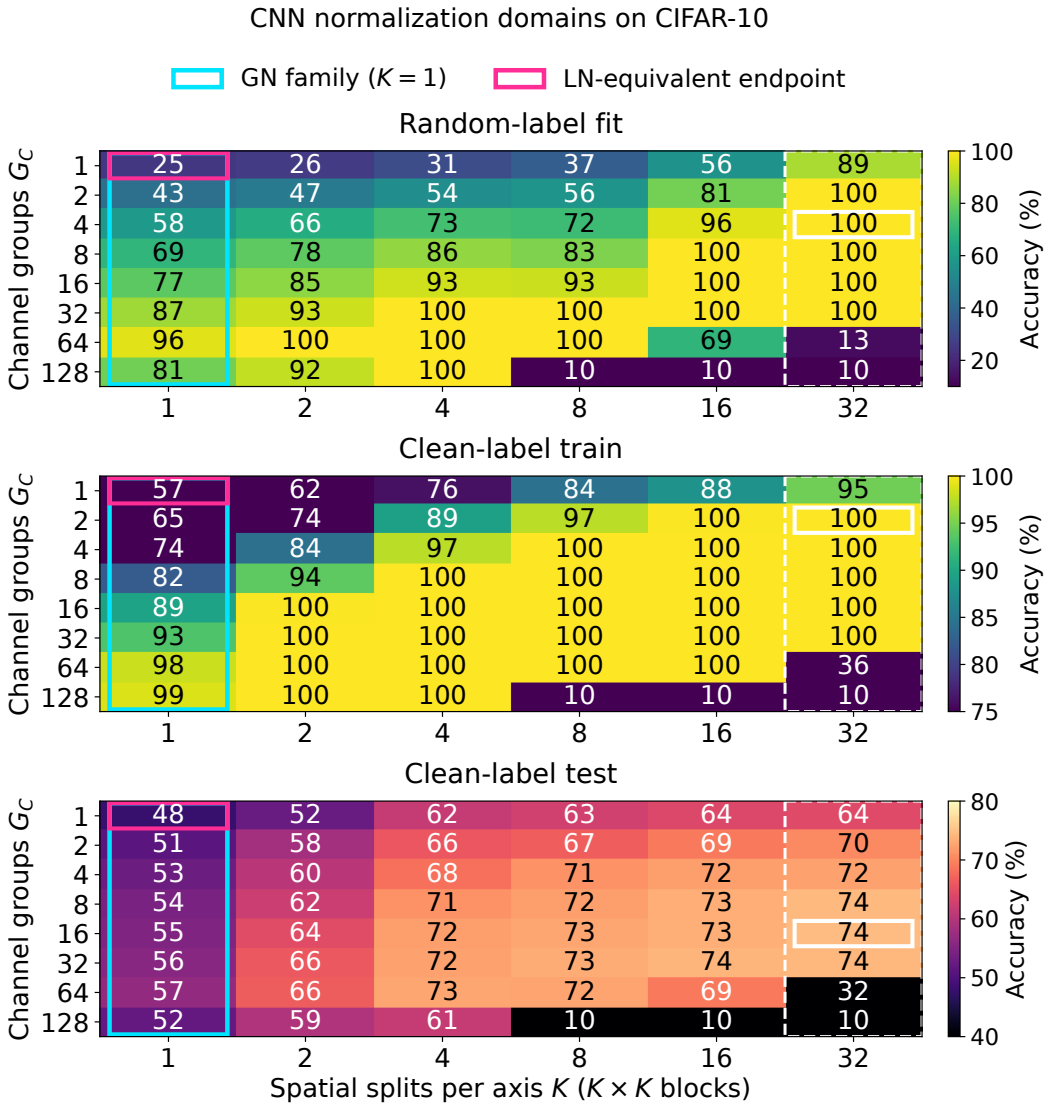


Figure A1: CIFAR-10 accuracy across mixed-axis domains in the depth-4, width-128 CNN (three-seed means): random-label fit (top), clean-label train (middle), and clean-label test (bottom). Cyan marks GN ($K = 1$), magenta marks LN, dashed $K = 32$ marks position-wise PLN, and white boxes mark panel maxima. Each cell shows its three-seed mean accuracy, rounded to the nearest percentage point.

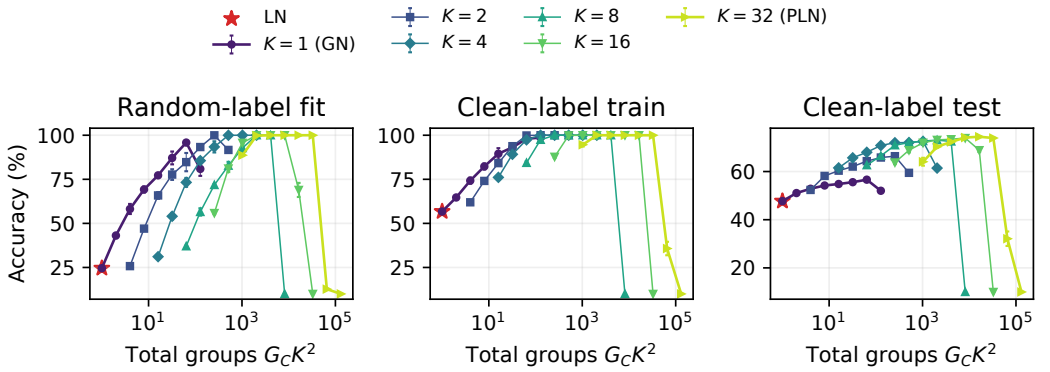


Figure A2: CNN accuracy versus total group count on CIFAR-10: (a) fitting 50,000 fixed random labels, (b) clean-label training, and (c) clean-label testing. K is the number of spatial patches per axis, so the total number of groups is $G_C K^2$. The GN ($K = 1$) and position-wise PLN ($K = 32$) directions are marked explicitly. Curves show three-seed means with sample standard deviations.

Figure A2 shows that the best attainable accuracy generally rises as the total group count increases, until overly fine partitions cause a sharp decline. Within a given grouping direction, accuracy did not improve monotonically with group count, and the three settings favor different grouping directions. At matched totals of 8–64 groups, GN ($K = 1$) gives the highest random-label fitting accuracy among the evaluated directions but the lowest clean-label test accuracy. For clean-label training, GN leads at 8–16 groups but is overtaken at 32–64 groups. Thus group count alone does not determine fitting or generalization; the grouped axes matter.

D.2.2 VGG-16 Initialization Sweep

Purpose and protocol. This auxiliary experiment studies whether the choice of nonlinear layer changes the sensitivity of a deep convolutional network to its learning rate and weight initialization. We train VGG-16 (Simonyan & Zisserman, 2014) on CIFAR-10 (Krizhevsky et al., 2009). We use the standard 50,000-image training split for optimization and the 10,000-image test split only for held-out evaluation. The VGG-16 backbone contains no additional BN or other normalization layer: each hidden nonlinear layer is replaced by one of PLN[8], ReLU, SiLU, Tanh, and SAT, while the rest of the architecture is held fixed. For a feature map in $\mathbb{R}^{B\times C\times H\times W}$, PLN[8] independently normalizes each consecutive group of eight channels at every spatial position. It uses $\epsilon = 10^{-5}$ and learnable channel-wise scale and bias, initialized to one and zero, respectively. The same data split, training augmentation, and optimization budget are used for every method, and no method-specific tuning is performed outside the common grid described below.

Hyperparameter grid. For every nonlinear layer, we take the Cartesian product of the initial learning rates

$$\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$$

and initialization multipliers

$$\{0.25, 0.35, 0.5, 0.7, 1.0, 1.4, 2.0, 2.8, 4.0\}.$$

The multiplier rescales the He initialization (He et al., 2015); hence, 1.0 is the unmodified He initialization, whereas, for example, 4.0 multiplies the initialized weights by four. This gives 63 hyperparameter configurations per nonlinear layer and 315 configurations in total. Each configuration is trained independently with three random seeds, for a total of 945 runs.

Optimization and evaluation. All models are trained for 200 epochs with mini-batch SGD (Amari, 1993), batch size 256, momentum 0.9, and weight decay 10^{-4}. The selected initial learning rate is linearly warmed up during the first 10 epochs and is subsequently divided by 2.5 at epochs 50, 90, 130, and 170. The same CIFAR-10 data-augmentation pipeline is used for all configurations. Weight decay is applied uniformly to all trainable parameters, including the affine parameters of PLN[8]. For

each fixed triple of nonlinear layer, learning rate, and initialization multiplier, Figure A3 reports the mean test accuracy over the three seeds. Thus, every cell in the figure reports a fixed configuration; no best-of-grid selection is applied.

Interpretation. PLN[8] is comparatively less sensitive to initialization scale than the element-wise activation baselines in this setting. We report this result as an auxiliary optimization study.

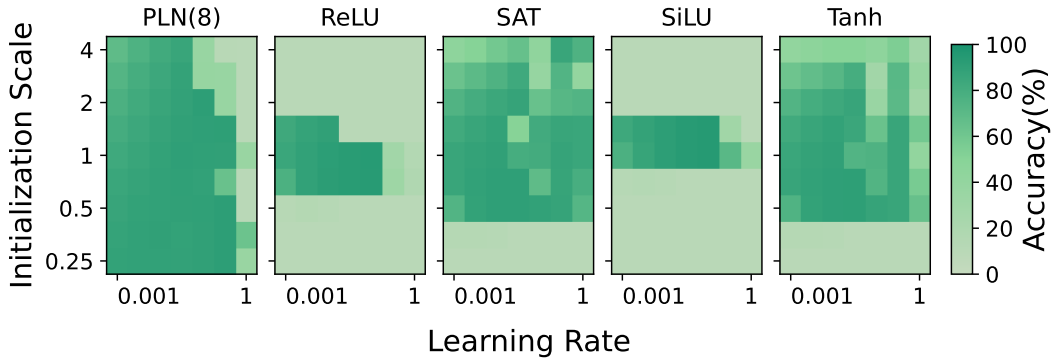


Figure A3: Mean CIFAR-10 test accuracy of VGG-16 models over three random seeds for each learning-rate–initialization pair. Each panel corresponds to one nonlinear layer, and darker green denotes higher accuracy. The learning rate increases from 0.001 (left) to 1 (right). The middle row uses the unmodified He initialization (He et al., 2015); the remaining rows scale its weights from $1/4$ (bottom) to 4 (top).

D.3 ViT Normalization-Domain Experiments

We train a depth-4, width-192 Vision Transformer (Dosovitskiy et al., 2021) on CIFAR-10 (Krizhevsky et al., 2009). Images are resized to 224×224 and divided into 16×16 patches, giving $T = 197$ tokens; the model has three attention heads and FFN width 768. We partition the token and channel axes into balanced groups. The all-normalization grid uses $G_T \in \{1, 2, 4, 8, 16, 32, 64, 128, 197\}$ and $G_C \in \{1, 2, 4, 8, 12, 24, 48, 192\}$; the activation-replacement grid uses the same token group counts and $G_C \in \{1, 2, 4, 8, 12, 24, 48, 96, 192\}$. In all-normalization replacement every LN is replaced and GELU is retained; in activation replacement every LN is retained and all FFN GELUs are replaced.

The primary controlled fitting study uses a fixed $6{,}000$-example subset with 100% random labels, following the finite-sample random-label fitting protocol (Zhang et al., 2017), shared across model seeds $40, 41, 42$. Models are trained for 100 epochs using AdamW (Loshchilov & Hutter, 2019), batch size 256, constant learning rate 10^{-4}, and weight decay 5×10^{-6}. Affine parameters are enabled for every normalization, while normalization weights and biases are excluded from weight decay. We report the best accuracy evaluated on the same fixed random-label set. The $20{,}000$-example study uses the same architecture and optimization protocol, and extends the $6{,}000$ fixed examples and labels with $14{,}000$ additional fixed examples and labels.

Figure A4 shows that high random-label fitting accuracy is concentrated at larger channel-group counts (the right side of each heatmap). The effect of grouping across tokens depends on the replacement site: under activation replacement, cross-token groupings ($G_T < 197$) offer a broader high-accuracy region than position-wise PLN, whereas under all-normalization replacement the position-wise row is stronger at most matched channel-group counts. Thus the bright right-hand region alone does not establish a general advantage for cross-token grouping.

$20{,}000$-sample random-label results. Figure A5 reports the complete three-seed grids. The best all-normalization point reaches 89.49% at $(G_T, G_C) = (32, 192)$, while the best activation-replacement point reaches 91.66% at $(64, 192)$. In both modes, the best cross-token point substantially outperforms the best point on the position-wise row.

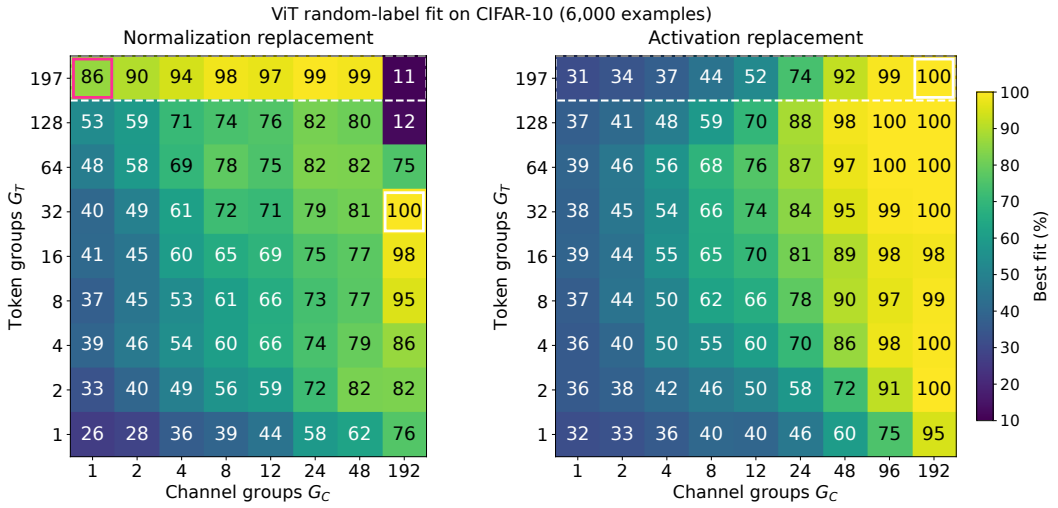


Figure A4: Random-label fitting over token–channel grouping domains in the depth-4, width-192 ViT. Each cell shows a three-seed mean rounded to the nearest percentage point. The dashed top row ($G_T = 197$) is the position-wise PLN direction. Left: every LN is replaced while GELU is retained. Right: LN is retained while all FFN activations are replaced. The magenta box marks standard LN in the all-normalization panel, and white boxes mark panel maxima.

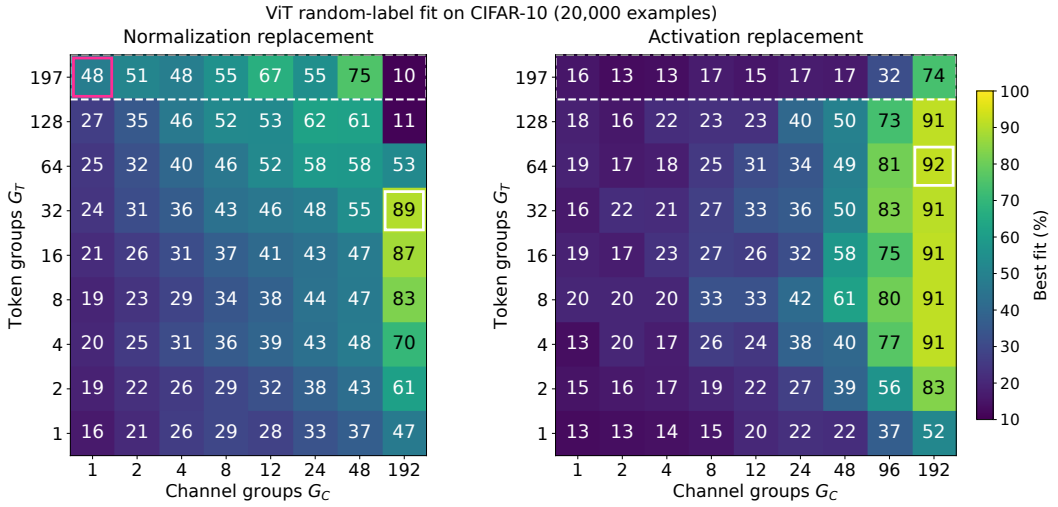


Figure A5: Three-seed random-20,000 ViT fitting results (cell labels rounded to the nearest percentage point). The magenta box marks standard LN in the all-normalization panel, the dashed row is the position-wise PLN direction, and white boxes mark maxima.

Clean-label comparison. For both replacement modes, we additionally train the complete grouping grid on all 50,000 clean CIFAR-10 training examples and evaluate on the standard 10,000-example test set. We use model seeds $40, 41, 42$ and otherwise retain the architecture, 100-epoch optimizer schedule, batch size, affine setting, and weight-decay exclusions described above. Figure A6 reports final training and test accuracy.

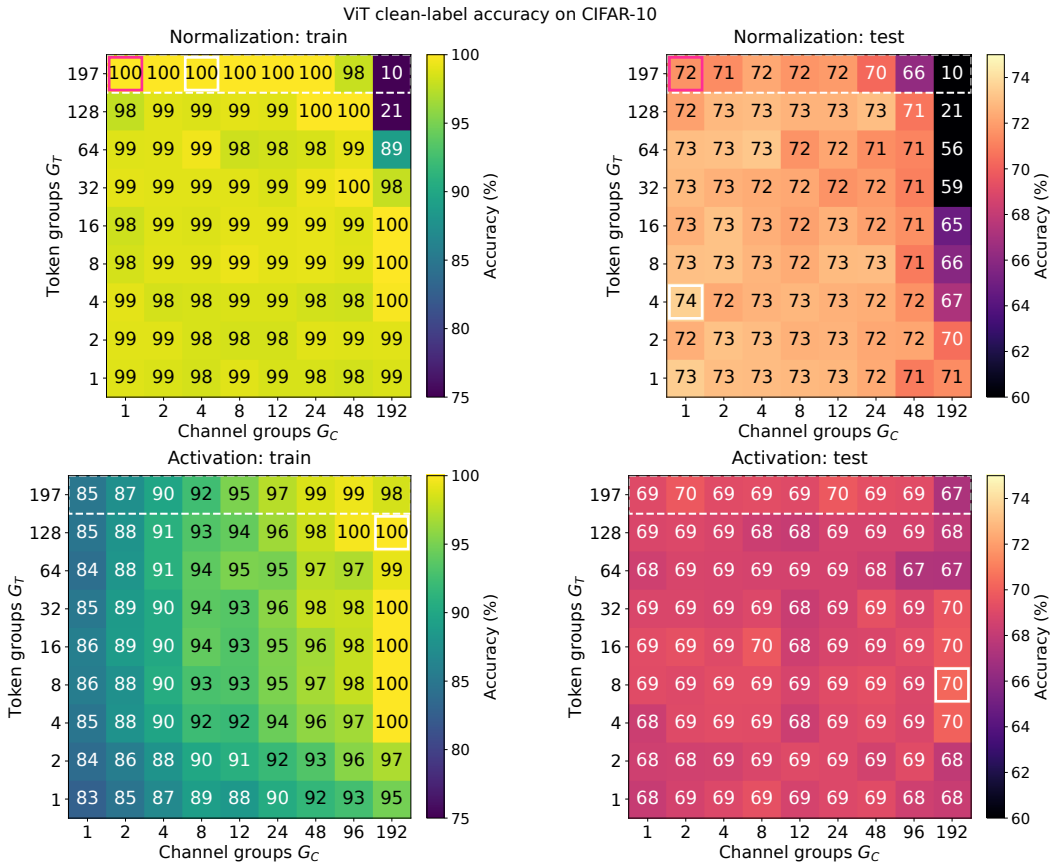


Figure A6: Three-seed clean-label ViT results (cell labels rounded to the nearest percentage point). Training accuracy varies strongly with domain geometry, while test accuracy occupies a much narrower range except at degenerate, near-scalar endpoints. The magenta box marks standard LN and white boxes mark panel maxima.

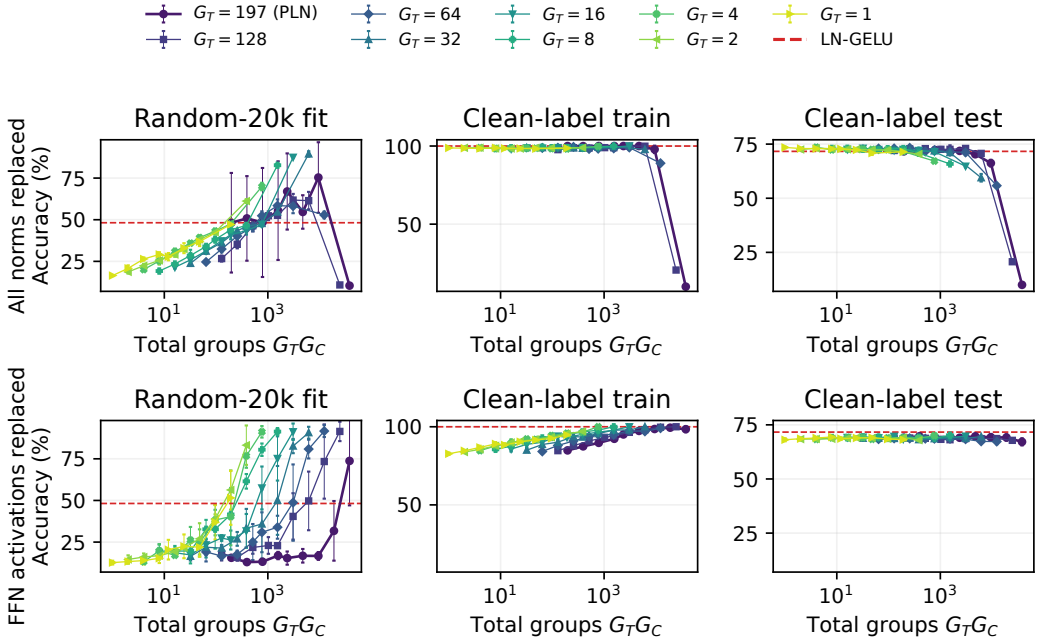


Figure A7: ViT accuracy versus total group count on CIFAR-10. Columns show fitting 20,000 fixed random labels, clean-label training, and clean-label testing; rows show all-normalization and FFN-activation replacement. Each curve fixes the number of token groups G_T; $G_T = 197$ is position-wise PLN, while $G_T < 197$ allows cross-token statistics. The dashed line is the original LN–GELU ViT. Curves show three-seed means with sample standard deviations.