PIECEWISE BERNSTEIN NETWORKS: SCALABLE, COMPATIBLE AND UNIVERSAL LEARNABLE ACTIVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learnable activation functions (like Kolmogorov-Arnold networks KAN, Polynomials, and Rational) increase model expressivity by enabling individual neurons to adapt their non-linear shapes to data. However, three critical obstacles prevent them from replacing fixed activations: a lack of scalability to large, complex architectures such as Vision Transformers, a lack of architectural compatibility, and limited expressivity in existing flexible formulations. We introduce piecewise Bernstein networks (PBN), a family of learnable activations formed by $K{+}1$ Bernstein polynomials of degree n joined continuously at K interior knots. We prove that PBN neurons can exactly reproduce foundational activations like ReLU, LeakyReLU, and PReLU as special cases and can approximate any other continuous activation function, rendering them into universal neurons. Against ten other activations on 13 benchmarks and in six settings, spanning function fitting, tabular MLPs, ResNets on CIFAR and ImageNet, an FT-Transformer and a vision transformer, PBN ranks first on eight benchmarks and second on the other five, and it is never behind the best fixed activation beyond seed noise. It raises ImageNet top-1 over ReLU by 1.06 points with ResNet-34 and 0.73 with ResNet-50, and the accuracy of a vision transformer by 1.0 and 2.0 points on CIFAR-10 and CIFAR-100, at 1.4 to 3.1 times ReLU's epoch time. On every benchmark, its validation-selected configuration reaches ReLU's final validation value in fewer epochs than ReLU.

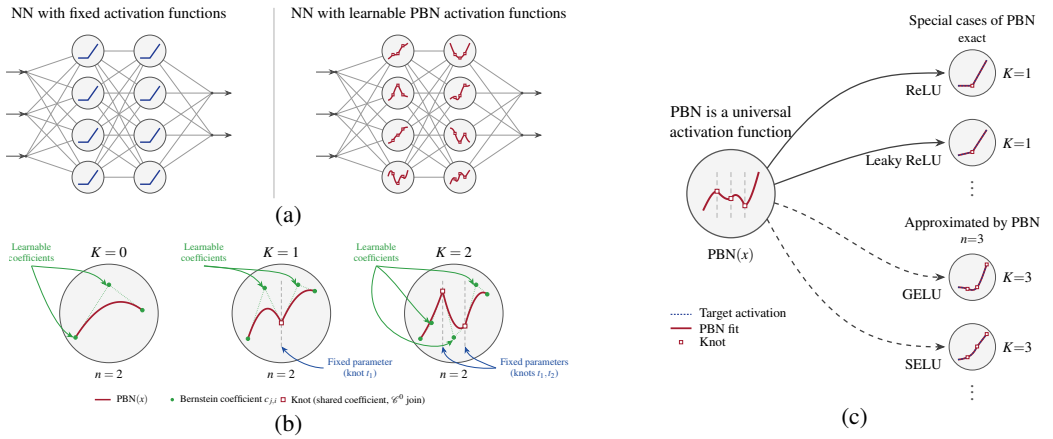

Figure 1: **PBN provides a scalable, compatible, and universal framework for learnable activation functions in deep learning. (a)** A PBN unit replaces a fixed activation in any network and learns its own shape (Section 2.2). **(b)** Each neuron joins $K{+}1$ Bernstein polynomials of degree n at K fixed knots and learns only their coefficients; $K = 0$ recovers DeepBern-Nets. **(c) Universal Expressivity:** By joining $K + 1$ local Bernstein polynomials continuously across K interior knots, PBN learns adaptive non-linear shapes while exactly recovering foundational activations (ReLU, LeakyReLU, PReLU) as special cases while able to approximate any other continuous activation (e.g., GELU and SELU) with an approximation error that shrinks as the number of knots K increases (Section 3). PBN ranks first on eight of 13 benchmarks and second on the rest (Section 5).

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.

1 INTRODUCTION

Networks usually fix their nonlinearity in advance and devote their capacity to the weights. A learnable activation moves part of that capacity into the nonlinearity: each unit carries a few parameters, trained alongside the weights, and learns the shape of its own curve (Apicella et al., 2021). Shape is then added where the network needs shape rather than width, and a unit can start as a standard activation and leave it only where the data require. Such units have repeatedly improved on fixed ones, from a learnable negative slope (He et al., 2015) and adaptive piecewise-linear units (Agostinelli et al., 2015) to a learnable convex-concave transition (Khan, 2026) and searched forms such as Swish (Ramachandran et al., 2017).

Learnable units now take several forms. Rational activations, ratios of learnable polynomials, train in deep image networks (Molina et al., 2020; Boullé et al., 2020) and in transformers (Fang et al., 2023). Kolmogorov-Arnold networks (KANs) go further and place a learnable univariate B-spline on every edge (Liu et al., 2024), and a line of variants keeps that structure while replacing the basis with Chebyshev polynomials (SS et al., 2024), Fourier series (GistNoesis, 2024), radial basis functions (Li, 2024) or rational functions (Yang & Wang, 2024). Polynomial networks raise the degree of the map itself (Chrysos et al., 2021), and Bernstein polynomials give both learnable units (Khedr & Shoukry, 2024) and smooth approximations of fixed ones (Zhang et al., 2026).

Three obstacles keep learnable activations from replacing fixed ones. The first is **scalability bottlenecks**. Most parameterized or adaptive activations fail to scale effectively when integrated into large, complex architectures like Transformers, Vision Transformers (ViTs), or deep tabular networks, often destabilizing optimization or causing prohibitive training slowdowns. The second is **compatibility**. A KAN (for example), as introduced, is a whole model, with its own optimiser and adaptive grid refinement (Liu et al., 2024), neither of which survives reduction to an activation slot. Each new backbone has required its own redesign, since the convolutional (Bodner et al., 2024) and transformer (Yang & Wang, 2024) versions restructure their layers around the unit. Nor does the spline edge itself carry over to transformer scale: KAT reports a spline ViT-T at 64.9 ImageNet top-1 against the plain ViT-Ti's 72.7 (Yang & Wang, 2024). The third is **expressivity**. KAN, Polynomial and rational units are smooth, so a kink (continuous non-smooth point), such as the one ReLU has at the origin or the many that piecewise-linear networks compose (Arora et al., 2016), can be approached but never reproduced, and a single global polynomial must raise its degree over the whole interval to sharpen its curve at any one point. At fixed depth, a network whose units are polynomials of a fixed degree computes a polynomial of bounded degree and is not a universal approximator (Leshno et al., 1993); learning the coefficients adds degree only through depth (Kileel et al., 2019), a restriction that matters most for targets with discontinuities or sharp local features.

We introduce *piecewise Bernstein networks* (PBN), a family of learnable activations indexed by the number of interior knots K: on a bounded interval, $K{+}1$ Bernstein polynomials of degree n, joined continuously at uniform knots, with $(K{+}1)n + 1$ learnable coefficients per unit. With $K = 0$ the family is exactly the global Bernstein activation of DeepBern-Nets (Khedr & Shoukry, 2024) (Lemma A.3), so PBN extends that activation where K becomes a hyperparameter like the degree. Against ten other activations on 13 benchmarks in six settings (Section 4), PBN is first or second on every benchmark and never falls behind the best fixed activation beyond seed noise (Section 5). The family addresses the three obstacles:

- **Universal.** By leveraging continuous knot partitions, a degree-1 PBN unit with a single interior knot exactly recovers ReLU, LeakyReLU, and PReLU. PBN inherits proven universal approximation capabilities: any continuous activation can be approximated to arbitrary precision, with an error bound that shrinks as the number of knots K grows (Corollary 3.2 and Figure 1c).
- **Compatible.** A PBN unit is a pointwise scalar map. One module takes an existing activation slot while topology, parameter shapes, optimiser and schedule stay as they were (Figure 1a), unchanged across MLPs, CIFAR and ImageNet ResNets (He et al., 2016), an FT-Transformer (Gorishniy et al., 2021) and a vision transformer (Dosovitskiy et al., 2021).
- **Scalable.** We implement custom fused Triton GPU kernels (Tillet et al., 2019) that compute the evaluation recurrence directly within registers. This avoids materializing intermediate basis tensors in memory, enabling PBN to train complex models like Vision Transformers and FT-Transformers while keeping epoch times within $1.4\times$ to $3.1\times$ of standard ReLU training.

2 BERNSTEIN-BASIS ACTIVATIONS

2.1 BERNSTEIN POLYNOMIALS

A polynomial of degree n admits a representation in the *Bernstein basis* on an interval $[l, u]$,

$$P_n^{[l,u]}(x) \;=\; \sum_{k=0}^{n} c_k \, b_{n,k}^{[l,u]}(x), \qquad b_{n,k}^{[l,u]}(x) \;=\; \frac{\binom{n}{k}}{(u-l)^n} \, (x-l)^k \, (u-x)^{n-k}, \tag{1}$$

where the $c_k \in \mathbb{R}$ are *control coefficients*. Unlike the power basis, this representation is intrinsic to $[l, u]$: the coefficients control the geometry of the curve directly.

A Bernstein network (Khedr & Shoukry, 2024) replaces every fixed activation in a feed-forward network with a learnable polynomial of the form equation 1. Writing $y^{(0)} = x$ and $y^{(\ell)}$ for the output of layer ℓ,

$$y^{(\ell)} \;=\; \sigma\Big(W^{(\ell)} y^{(\ell-1)} + b^{(\ell)};\, c^{(\ell)}\Big), \qquad \sigma(x;\, c^{(\ell)}) \;=\; \sum_{k=0}^{n} c_k^{(\ell)} \, b_{n,k}^{[l,u]}(x), \tag{2}$$

with σ applied element-wise and the control coefficients $c^{(\ell)} = \{c_k^{(\ell)}\}_{k=0}^{n}$ learned by backpropagation alongside $W^{(\ell)}$ and $b^{(\ell)}$. The degree n is a hyperparameter.

Three properties of Bernstein polynomials matter. The basis forms a *partition of unity*, $\sum_k b_{n,k}^{[l,u]}(x) = 1$ on $[l, u]$, so the output is a weighted average of the control coefficients. The *convex-hull* property then bounds the polynomial by $\min_k c_k \le P_n^{[l,u]}(x) \le \max_k c_k$. And the basis is *variation diminishing*: the curve crosses any line no more often than its control polygon does, so it cannot introduce oscillations that the coefficients do not already contain.

These three properties make the activation unit stable in training (Khedr & Shoukry, 2024): its output is bounded by its coefficients and oscillates no more than they do. Nevertheless, the same averaging is the source of its limitation, *global coupling*. Every basis function in equation 1 is nonzero on the interior of $[l, u]$, so every control coefficient influences the output at almost every input, and the classical approximation rate is correspondingly slow.

2.2 PIECEWISE BERNSTEIN NETWORKS (PBN)

PBN generalizes this activation by partitioning the domain $[l, u]$ of each activation neuron as $l = t_0 < t_1 < \cdots < t_K < t_{K+1} = u$ by placing K *interior knots* and splitting $[l, u]$ into $K + 1$ segments (Figure 1b). On segment j the activation is a degree-n Bernstein polynomial in its own local coordinates:

$$\sigma(x) \;=\; \sum_{k=0}^{n} c_{j,k} \, b_{n,k}^{[t_j,\, t_{j+1}]}(x), \qquad x \in [t_j,\, t_{j+1}]. \tag{3}$$

Continuity. The segments are tied together at the knots. Because a Bernstein polynomial evaluates to its endpoint coefficient at an endpoint, continuity at t_{j+1} is the single linear constraint $c_{j,n} = c_{j+1,0}$: the last coefficient of one segment and the first of the next are one shared value. Segment 0 therefore contributes $n+1$ free coefficients and each later segment n, giving $(K+1)n + 1$ free coefficients in place of $(K+1)(n+1)$. Unlike the cubic B-splines of KAN (Liu et al., 2024), which are C^2 at simple knots and therefore cannot represent a kink, a PBN unit is only C^0. Hence, a knot can hold a kink (Theorem 3.1), and continuity costs a single shared coefficient per knot.

With $K = 0$ there is a single segment and a PBN unit represents the special case of Bernstein network (Khedr & Shoukry, 2024). We write PBN (K=0) for it and PBN ($K \ge 1$) for neurons with interior knots, where each coefficient acts on its own segment; K is a hyperparameter of the family alongside the degree n.

Knot placement. Uniform knots place t_j at $\mathrm{linspace}(l, u, K+2)$, so the parity of K decides whether a kink falls on the origin: an odd K places one there, and an even K leaves the origin in the interior of a segment. This matters because exact representation of activations with a kink at zero requires a knot there (Section 3).

3 UNIVERSALITY OF APPROXIMATION

This section details the representation capacity of a single PBN activation function unit. All proofs (shown in AppendixA) construct the control coefficients explicitly, ensuring that our analysis hold independently of network depth, width, or optimization dynamics.

Before stating the main universality result, we establish that the representation capability of a PBN unit stems directly from the completeness of its underlying spline basis. Theorem 3.1 shows that PBN can exactly represent ReLU, LeakyReLU and PReLU, while Corollary 3.2 shows that any continuous function can be uniformly approximated to arbitrary precision by a single PBN unit, provided the polynomial degree n is sufficiently large. Crucially, this density guarantee requires no deep composition or wide parallel layers; the capacity is native to a single univariate mapping.

Theorem 3.1 (Exact piecewise-linear activations)**.** *Let $l < 0 < u$, place one interior knot at $t_1 = 0$ and extend the unit linearly outside $[l, u]$. A* PBN *unit of degree $n = 1$ with coefficients:*

$$c_{0,0} = \alpha l, \qquad c_{0,1} = c_{1,0} = 0, \qquad c_{1,1} = u \tag{4}$$

is exactly ReLU *for $\alpha = 0$,* $\mathrm{LeakyReLU}_\alpha$ *for a fixed $\alpha \in (0, 1)$, and* PReLU *for a learnable α.*

Corollary 3.2 (Every continuous activation is reachable)**.** *Let σ be continuous on $[l, u]$, and give a* PBN *unit with K uniform interior knots and degree n the coefficients*

$$c_{j,k} \;=\; \sigma\big(t_j + \tfrac{k}{n}\,(t_{j+1} - t_j)\big)\,, \qquad j = 0, \ldots, K, \;\; k = 0, \ldots, n, \tag{5}$$

which satisfy the continuity constraint of Section 2.2 by construction. Then $\|\sigma - \sigma_{\mathrm{PBN}}\|_\infty \le \frac{3}{2}\,\omega\big(\sigma;\, (u-l)/((K{+}1)\sqrt{n})\big)$, where ω is the modulus of continuity, so every $\varepsilon > 0$ is met by some K and n. This covers GELU, SiLU, SELU, $\tanh$ *and the logistic sigmoid.*

A few remarks follow directly. First, setting $K = 0$ gives the classical rate of a single Bernstein polynomial, with argument $(u - l)/\sqrt{n}$. That is, each added knot divides that argument, so accuracy can be bought by refining the mesh instead of raising the degree. Second, because a PBN ($K{\ge}1$) unit with a knot at the origin exact-recovers a ReLU (Theorem 3.1), networks composed of these units subsume standard ReLU networks and are universal approximators. By contrast, networks of degree-n PBN ($K{=}0$) units without interior knots can only represent fixed-degree polynomials at a given depth, failing the universality property (Leshno et al., 1993). Finally, since a degree-n Bernstein segment can express any polynomial of degree n, the best PBN approximation for a C^{n+1} function σ inherits the $O(h^{n+1})$ rate of piecewise degree-n interpolation (de Boor, 2001), since each segment contains all degree-n polynomials and the interpolant is continuous.

In practice, the formula in equation 5 serves as our initialization strategy; specifically, setting $\sigma = \mathrm{GELU}$ initializes all PBN ($K{\ge}1$) units in our scaled experiments (Section B.1). Indeed, training is not restricted to these sampled values, as the optimizer might find better shape for the activation function. Nonetheless, they provide initial shape for the activation function, from which gradient descent can keep adapting.

Scope. It is crucial to note that the theoretical statements above are unit-level and constructive: they give no network-level approximation rate, and we claim none. They describe what an activation function can represent, not what training reaches; Section 5 measures how much of this capacity gradient descent finds. And they count cost in coefficients, not in time: wall-clock is reported separately in Section 5, and the two need not agree.

4 EXPERIMENTAL SETUP

Our empirical comparison spans six distinct architectural configurations. These include two parameter-matched MLP baselines and four scale settings designed to isolate the impact of the activation function by keeping the host network static. Table 1 summarizes the experimental design, while Appendix B details the full setup.

Activations comparison. We evaluate both variants of PBN, PBN ($K{=}0$) and PBN ($K{\ge}1$). For both, we fix the domain for all neurons as $[-5, 5]$ and outputs are clamped by default outside the $[-5, 5]$ except for the FT-Transformer where units extend linearly according to their boundary slopes.

Table 1: **Summary of experimental settings and compute footprint.** Overview of the six evaluation settings. Total runs represent the product of hyperparameter configurations and random seeds evaluated per setting.

Setting	Host network	Data	Metric	Seeds	Runs
Function fitting	MLP, 10^3 to 10^5 params	53 analytic targets	test MSE	42, 123, 456	42,882
Tabular MLP	MLP, 10^3 to 10^5 params	HIGGS	test AUC	42, 123, 456	690
CIFAR	ResNet-18	CIFAR-10/100	test top-1	42, 123, 456	108
ImageNet	ResNet-34/50	ImageNet-1k	val. top-1	42, 43	28
FT-Transformer	FT-Transformer	HIGGS	test AUC	42, 123, 456	60
ViT	ViT-T/4	CIFAR-10/100	test top-1	42, 123, 456	102

This boundary treatment accommodates distinct input distributions: with batch normalization placed directly prior to the activation—as in our MLP and ResNet architectures—only 0.23% of inputs fall outside $[-5, 5]$. Conversely, in the FT-Transformer where a linear layer follows LayerNorm, 3% to 11% of inputs fall outside this domain (Section B.7). For settings lacking prior normalization where out-of-domain inputs are frequent, we recommend this linear tail extension to preserve non-zero gradients. Our evaluation compares these variants against two primary baseline categories:

- **Learnable-Basis Baselines:** Edge-based networks including KAN (Liu et al., 2024), Chebyshev (SS et al., 2024), and Fourier (GistNoesis, 2024) representations, as well as Pi-Net (Chrysos et al., 2021), GR-KAN (Yang & Wang, 2024) (our rational baseline), and BERLU (Zhang et al., 2026). BERLU—a learnable-slope leaky ReLU variant—serves as our PReLU-style baseline.
- **Fixed Activations:** Standard non-linearities including ReLU, GELU (Hendrycks, 2016), tanh, and ReGLU (Shazeer, 2020) (the default gated block for the FT-Transformer).

Formal definitions for all activations appear in Table 2, and task-specific hyperparameter sweep details are provided in Appendix B.

Parameter-matched protocol (MLP settings). To ensure fair comparison, all networks are constructed across three distinct capacity regimes: 10^3, 10^4, and 10^5 trainable parameters, inclusive of activation function coefficients. Architecturally, the PBN, ReLU, and tanh networks share a residual scaffold consisting of $D \in \{2, 4, 8\}$ hidden layers with batch normalization. The remaining baseline architectures maintain their native network structures across three corresponding shapes. For each model, layer widths are solved analytically to match the target parameter count precisely at each tier (Appendix B.1-B.2). All models are trained using AdamW (Loshchilov & Hutter, 2019) with a weight decay of 10^{-4} evaluated across two base learning rates; the Bernstein coefficients are updated at the base learning rate with weight decay excluded.

Substitution protocol (scale settings). In the scale settings, each host network operates under a fixed architecture, training recipe, and learning-rate grid. Rather than parameter-matching, activation coefficients are added directly to the host network, introducing marginal parameter overhead (at most 1.5% on CIFAR, 2.4% on ImageNet, 0.02% for the ViT, and 10.4% for the FT-Transformer; Section B.2). For transformer architectures, activation substitutions are restricted exclusively to the feed-forward blocks. In ResNet architectures, both PBN variants and GELU replace the internal non-linearities within each residual block, while maintaining the standard post-addition ReLU; this placement strategy is applied uniformly across all ImageNet experiments. Bernstein coefficients are updated at the base learning rate without weight decay. Finally, to stabilize training across ImageNet runs, control-point gradients for each PBN ($K \geq 1$) piece are rescaled linearly based on distance from the domain center $[-5, 5]$, scaling smoothly from 0.5 at the origin to 1.5 at the outermost boundaries.

Reporting. For all experimental settings (Appendices D to I), we report performance using the median and range across random seeds, evaluated on the test set using the checkpoint selected via validation. Because hyperparameter grid sizes vary across activations—ranging from 6 configurations per parameter tier for standard baselines up to 36 for PBN ($K \geq 1$) in function-fitting tasks—activations are evaluated configuration by configuration; any single representative metric per activation is selected strictly based on validation performance to prevent test-set contamination. For ImageNet, which lacks public test labels, we report the peak top-1 validation accuracy observed during training. Precision and hardware configurations follow standard training practices: MLP experiments execute in single precision (fp32), whereas scale settings utilize mixed precision (bfloat16 for ImageNet and float16 across all other benchmarks), with each individual run mapped to a single GPU.

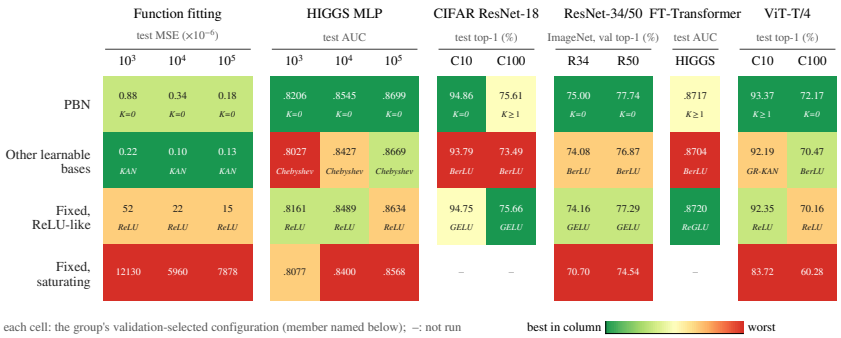

Figure 2: Each cell displays the top-performing configuration selected via validation from the group's pooled hyperparameter grid, alongside its reported test metric (or peak top-1 validation accuracy for ImageNet, which also serves as the selection criterion) and the specific activation that produced it. Cell color indicates the group's rank within each benchmark column. The PBN group pools both PBN ($K{=}0$) and PBN ($K{\geq}1$) variants; learnable bases include KAN, Chebyshev, Fourier, Pi-Net, GR-KAN, and BERLU; fixed ReLU-like activations comprise ReLU, GELU, and ReGLU. Function-fitting tasks report test MSE ($\times 10^{-6}$) across three parameter tiers (10^3, 10^4, and 10^5), with dashes denoting unevaluated groups. Individual activation results are detailed in Figure 9.

5 RESULTS

Figure 2 summarizes performance across the 13 benchmark columns spanning our six architectural settings. To enable direct comparison, activations are categorized into four distinct groups. Within each benchmark column, validation performance determines the optimal configuration from each group's pooled hyperparameter grid according to the setting-specific selection criteria (Section 4). Pooling standardizes comparisons across groups with differing grid sizes and treats knot count as a standard hyperparameter by evaluating PBN ($K{=}0$) and PBN ($K{\geq}1$) within a unified PBN group.

PBN achieves top rank in 8 of the 13 benchmark columns, including all three HIGGS parameter tiers, CIFAR-10, both ImageNet model depths, and both ViT datasets. In the remaining 5 columns, PBN ranks second, trailing KAN in function fitting, GELU on CIFAR-100 (75.61% vs. 75.66%), and ReGLU on the FT-Transformer (0.8717 vs. 0.8720)—with the latter two margins falling well within the standard deviation across random seeds. Notably, PBN is the only group to rank among the top two across all 13 settings. Other learnable-basis baselines perform best solely in function fitting while placing last on CIFAR and the FT-Transformer; fixed ReLU-like non-linearities exhibit $59\times$ to $82\times$ higher error than PBN in function fitting; and tanh ranks last across all evaluated configurations. Regarding knot selection, validation chooses $K{=}0$ in 7 columns (as well as 34 to 44 of the 52 function-fitting targets) and $K{\geq}1$ in the remaining 3 columns (CIFAR-100, FT-Transformer, and ViT CIFAR-10). These results demonstrate that performance is not driven by a single variant or an artifact of grid pooling: even when evaluated independently, PBN ($K{=}0$) alone maintains top rank in 8 of the 13 columns (Figure 9).

5.1 PREDICTIVE PERFORMANCE

Grid-level results and sensitivity analysis. To account for differences in hyperparameter grid sizes across methods (Section 4), the figures 3-4 depict all evaluated configurations, displaying the median and range across random seeds, highlighting the validation-selected configuration with a ring, and marking ReLU's selected baseline as a horizontal reference line. Corresponding plots evaluated on validation data are provided in Appendices D to I. Furthermore, to control for search space volume, we conduct a control experiment where every activation is restricted to the same number of randomly sampled configurations as ReLU's grid (Appendices D to I). Under this equalized budget, PBN consistently outperforms ReLU across all settings, retains its top rank on HIGGS and ViT architectures, while GELU achieves higher accuracy on CIFAR and ReGLU on the FT-Transformer.

MLP settings. In function fitting (Figure 3, top), all 18 PBN ($K{=}0$) configurations have a lower error than ReLU's selected configuration at 10^3 and 10^5 parameters and 17 at 10^4, as do 32 or 33

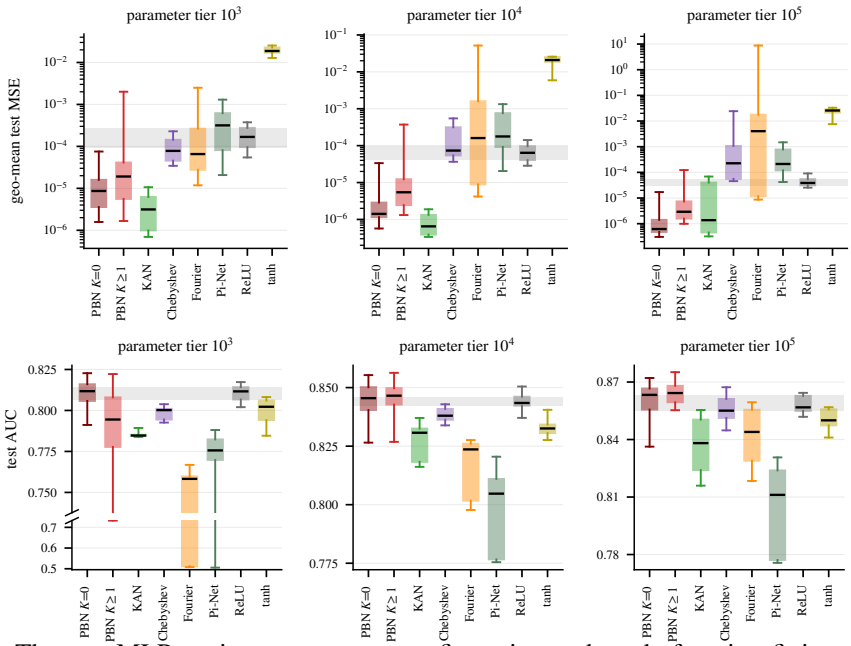

Figure 3: The two MLP settings over every configuration and seed: function fitting, test MSE averaged geometrically over the 52 targets (top, log scale, lower is better), and HIGGS, test AUC (bottom, higher is better). Each box pools an activation's runs at a parameter tier; it covers the middle half, the line is the median, the whiskers reach the best and the worst run, and the grey band spans ReLU's box. The individual configurations are shown per tier in Figures 10 and 34 and the figures that follow them.

of the 36 PBN ($K \geq 1$) configurations, against at most half of the Chebyshev, Fourier and Pi-Net configurations. Over every configuration and seed, KAN has the lowest median error at 10^3 and 10^4 (3.1 and 0.65×10^{-6}) and PBN ($K=0$) at 10^5 (0.62×10^{-6}, against 1.4 for KAN and 39 for ReLU). On HIGGS (Figure 3, bottom), both PBN forms have the highest median AUC at 10^4 and 10^5 (0.845 and 0.847, then 0.863 and 0.864, against 0.843 and 0.857 for ReLU) and hold the two best runs at every tier, up to 0.875 for PBN ($K \geq 1$) at 10^5. At 10^3, PBN ($K=0$) matches ReLU (median 0.812), while the depth-8 configurations of PBN ($K \geq 1$) reach only 0.74 to 0.78. As the tier grows, the runs of PBN ($K \geq 1$) concentrate: from 10^3 to 10^5 parameters their range narrows from three orders of magnitude of error to two in function fitting and from 0.090 to 0.020 AUC on HIGGS, where it is then narrower than for PBN ($K=0$) (0.036) and KAN (0.040). On HIGGS at 10^5, half of each PBN grid exceeds ReLU's selected configuration, against one of six Chebyshev configurations and none of KAN, Fourier or Pi-Net.

Convolutional networks. On CIFAR (Figure 4, top), four of the six PBN ($K=0$) and five of the six PBN ($K \geq 1$) configurations exceed ReLU's selected configuration on CIFAR-10, and five and six on CIFAR-100, as do both GELU configurations and neither BERLU configuration. The selected PBN ($K=0$) is the most accurate network on CIFAR-10 (94.86, against 94.75 for GELU and 94.55 for ReLU), and on CIFAR-100 the selected PBN ($K \geq 1$) (75.61) is level with GELU (75.66) and 0.73 above ReLU. On ImageNet (Figure 4, bottom), the two PBN forms are the two most accurate networks at both depths. At ResNet-34, PBN ($K=0$) ($n=9$) is 1.06 points above ReLU and PBN ($K \geq 1$) ($n=9$, $K=8$) 0.87, with every one of their seeds at least 0.77 above ReLU's better seed; GELU is 0.22 points above ReLU, BERLU 0.14 above and tanh 3.2 below. At ResNet-50, PBN ($K=0$) with $n=3$ is 0.73 points above ReLU and PBN ($K \geq 1$) 0.60, against 0.29 for GELU and -0.13 for BERLU, while PBN ($K=0$) with $n=9$ remains at ReLU's level (-0.03). These gains do not come from added parameters: PBN ($K=0$) adds 0.18% to ResNet-34 and 0.12% to ResNet-50 (Appendix B.2). They also hold against GELU, which uses the same ResNet implementation as PBN: PBN leads it by 0.84 and 0.44 points at the two depths.

Transformers. In the ViT (Figure 68, Appendix I), each PBN block learns one curve shared by its 768 channels, so a PBN network has at most 876 more parameters (0.02%) than the ReLU one. Every PBN ($K=0$) configuration and 16 of the 18 PBN ($K \geq 1$) configurations exceed ReLU's, while among

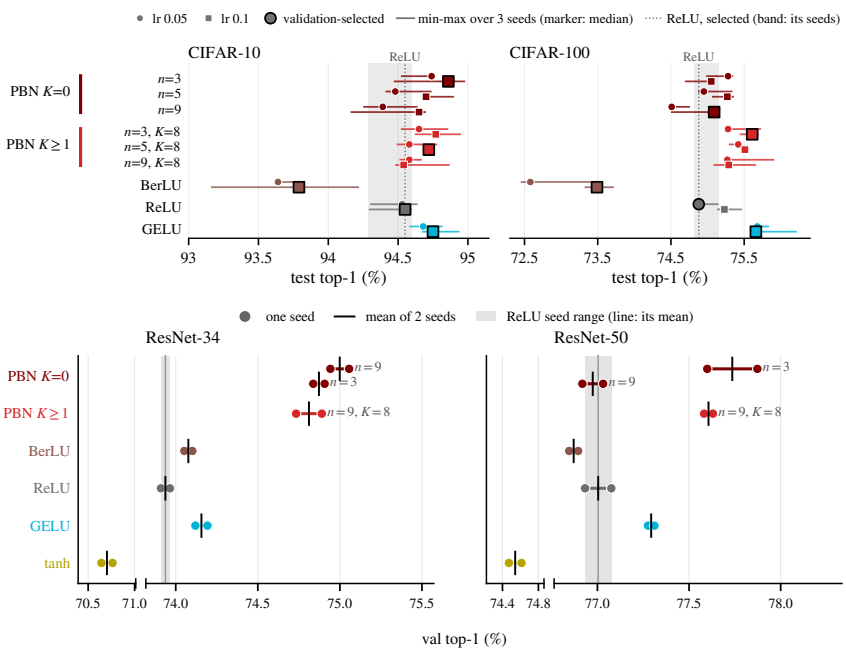

Figure 4: Convolutional networks. Top: every configuration of ResNet-18 on CIFAR-10 and CIFAR-100, test top-1; the marker is the median of three seeds and the bar their range, circles and squares are the two base learning rates, the ringed marker is the validation-selected configuration, and the dotted line and the grey band are ReLU's selected configuration and its seed range. Bottom: ImageNet validation top-1 of ResNet-34 and ResNet-50; the dots are the two seeds, the bar their range and the tick their mean, the grey band is ReLU's seed range, and for PBN ($K{=}0$) the configuration selected at each depth is drawn above the other one. tanh lies beyond a break in the axis.

the other activations only BERLU does, on CIFAR-100. With its selected configuration, PBN ($K{=}0$) is the most accurate activation on both datasets (93.86 and 72.17, against 92.35 and 70.16 for ReLU). In the FT-Transformer (Figure 58), all eight PBN ($K{\geq}1$) configurations exceed ReLU's selected configuration (medians 0.8714 to 0.8718, against 0.8704), and over every configuration and seed PBN ($K{\geq}1$) has the highest median AUC (0.8717, against 0.8712 for ReGLU); with each activation's selected configuration, ReGLU is first (0.8720) and PBN ($K{\geq}1$) second. All configurations lie within 0.003 AUC of one another, so the choice of activation has little effect in this setting, and PBN ($K{\geq}1$) has the best AUC of any run (0.8737, against 0.8729 for ReGLU).

5.2 CONVERGENCE

To compare how quickly the activations learn, we take ReLU's final validation value as a common target. A configuration's curve is the median over seeds of the best validation value reached so far; the target is the end of the curve of ReLU's selected configuration, and a configuration's *crossing* is the first epoch at which its curve reaches the target. Figure 5 shows the crossings of every configuration in function fitting, and Appendices D to I give the curves behind them and the other five settings.

In function fitting the comparison turns in favor of PBN as the parameter tier grows (Figure 5). ReLU improves until the cap of 3,000 epochs, while every PBN ($K{=}0$) configuration and 33 to 35 of the 36 PBN ($K{\geq}1$) configurations reach its final error much earlier. At 10^3 parameters KAN crosses first (median 596 epochs, against 863 for PBN ($K{=}0$) and 800 for PBN ($K{\geq}1$)); at 10^4 the three are close (340, 384 and 436); at 10^5 both PBN forms cross at a median of 213 to 221 epochs, ahead of KAN (365), of which only four of six configurations still cross. In the ViT (Figure 75), every PBN ($K{=}0$) configuration reaches ReLU's final accuracy, at a median of epoch 203 on CIFAR-10 and 197 on CIFAR-100, and PBN ($K{\geq}1$) at 218 (nine of nine configurations) and 248 (six of nine), against 284 and 288 for ReLU; GELU, GR-KAN and tanh never reach it, and BERLU does only on CIFAR-100 (245). The other settings show the same ordering, and in every one of the 13 columns the configuration that validation selects for PBN crosses before ReLU's (Appendices D to I). On CIFAR-10, five of the six configurations of each PBN form cross, at median epochs 169 and 175

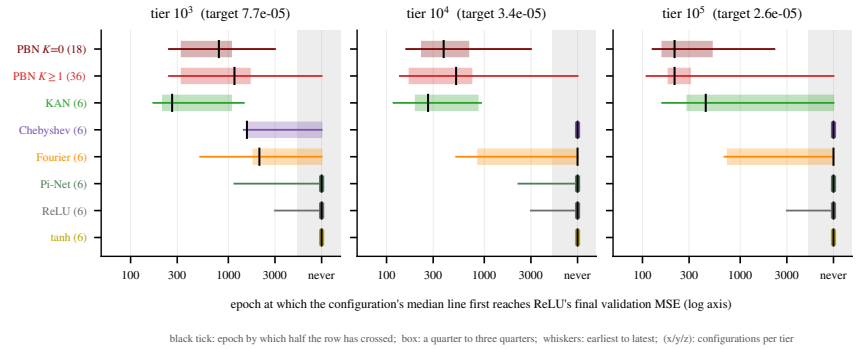

Figure 5: Function fitting: the epoch at which each configuration first reaches ReLU's final validation error (geometric mean over the 52 targets; log axis), per parameter tier. A box summarises an activation's configurations (their number in parentheses): the middle half, the median (black tick) and the earliest to latest crossing; configurations that never reach the target are collected at the right.

against 192 for ReLU; on CIFAR-100, four PBN ($K \geq 1$) configurations do, at a median of 166.5 against 181. On ImageNet the selected configurations of both forms cross at epochs 84 to 86 of 90, and in the FT-Transformer all eight PBN ($K \geq 1$) configurations cross at a median of 19 epochs, against 25. On HIGGS the effect grows with the tier: at 10^5 half of the PBN ($K \geq 1$) configurations cross, at a median of 18.5 epochs against 75.

Cost per epoch. With each run alone on one GPU, an epoch with PBN ($K=0$) costs 1.4 to 2.2 times one with ReLU, and with PBN ($K \geq 1$) 1.5 to 3.1 times, except in function fitting (5.6 to 7.9 times); KAN costs 4.3 to 43 times ReLU in the MLP settings and GR-KAN 2.7 to 2.8 times in the ViT (Appendices D to I). An earlier crossing therefore saves training time only where the lead is large: at 10^5 parameters, PBN ($K=0$) reaches ReLU's final error in function fitting in about a seventh of ReLU's training time, and PBN ($K \geq 1$) reaches ReLU's AUC on HIGGS in about half. Appendix C.2 relates each activation's accuracy to its cost and memory, and Appendix C.1 turns the results into guidance on when PBN is worth it.

6 Conclusions and Limitations

We introduced Polynomial Bernstein Networks (PBN), demonstrating that parameterized piecewise-Bernstein activation functions offer strong expressive capacity and competitive performance across diverse architectures. Empirically, PBN achieves top-tier performance in 8 of 13 benchmark columns and ranks among the top two across all evaluated settings, proving to be a versatile alternative to both fixed non-linearities and existing learnable-basis baselines.

Four primary limitations qualify these findings. First, in scale settings, a training epoch with PBN incurs a 1.4x to 3.1x wall-clock overhead relative to ReLU (benchmarked on A30 and A100 GPUs); thus, while PBN reaches target validation accuracy in fewer epochs, it does not reduce overall wall-clock training time. Nevertheless, we argue that PBN could amortize the total computational expenditure by removing the need for trial-and-error activation search. Standard activation functions exhibit strong domain preferences (e.g., ReGLU on tabular data vs. GELU on vision transformers). Achieving optimal performance with standard activations requires sweeping multiple candidate functions across full training runs. Instead, PBN serves as a universal, plug-and-play non-linearity that autonomously converges to task-optimal shapes, which would lead to a net reduction in end-to-end compute compared to standard hyperparameter search.

Second, with three random seeds per configuration (two on ImageNet), our reported medians and ranges demonstrate clear trends without asserting formal statistical significance, and ImageNet evaluations for PBN ($K \geq 1$) relied on a single configuration (n=9, K=8) selected and reported on the validation split. Third, the two variants differ beyond knot count K: PBN ($K=0$) uses random coefficient initialization while PBN ($K \geq 1$) uses random walks or GELU-shaped sampling curves, per-piece gradient scaling was applied only to PBN ($K \geq 1$) on ImageNet, and ViT training hyperparameters were selected during early pilots on PBN variants. Fourth, the domain and uniform knot placements were fixed for computational efficiency without explicit ablation, boundary extension rules were compared only within the FT-Transformer, learnable-basis baselines (KAN variants and Pi-Net) were restricted to MLP settings, and host network scales were capped at ResNet-50.

AI USE STATEMENT

We used generative AI tools (Claude models by Anthropic, through Claude Code) in this work. Among the tasks that require disclosure, they helped implement the experiment code (training scripts, job generation and result extraction), gave feedback on the experimental methodology and on the selection and reporting protocol, helped interpret results, and helped state and write up the lemma, theorems and proofs of Section 3 and Section A from the authors' derivations. They were not used to generate data sets, to translate, or for qualitative or thematic data analysis, and every reported number comes from our own training runs. Among the tasks with recommended disclosure, they wrote and edited analysis and plotting code, created the figures from our results, edited parts of the paper, and helped identify and format references. We reviewed all AI-assisted work, checked the reported numbers against the run data, and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies activation functions for neural networks. Its experiments use analytic functions generated by code and public benchmark datasets (HIGGS, CIFAR-10, CIFAR-100 and ImageNet-1k), used under their terms of use; it involves no human subjects and collects no personal data. We are not aware of a use of this method that raises ethical concerns beyond those of neural networks in general. Its computational cost is reported in Section 5.2 and Appendices C to I.

REPRODUCIBILITY STATEMENT

Section B gives every setting behind the reported numbers: data and splits, host networks, recipes, sweeps, parameter counts and the rule that selects each reported configuration. Section A gives complete proofs of the statements of Section 3, and Appendices D to I report every configuration and seed rather than a single selected number, so each result can be checked against its full grid. We will release the code, including the Triton kernels, the training scripts and the scripts that produce every figure and table.

REFERENCES

Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. In *International Conference on Learning Representations (ICLR), Workshop Track*, 2015.

Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, 2021.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1):4308, 2014.

Alexander Dylan Bodner, Antonio Santiago Tepsich, Jack Natan Spolski, and Santiago Pourteau. Convolutional Kolmogorov–Arnold networks. *arXiv preprint arXiv:2406.13155*, 2024.

Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 14243–14253, 2020.

Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. Deep polynomial neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 44(8):4021–4034, 2021.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer, New York, revised edition, 2001.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

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 *International Conference on Learning Representations*, 2021.

Haishuo Fang, Ji-Ung Lee, Nafise Sadat Moosavi, and Iryna Gurevych. Transformers with learnable activation functions. In *Findings of the Association for Computational Linguistics: EACL 2023*, 2023.

GistNoesis. FourierKAN. https://github.com/GistNoesis/FourierKAN, 2024. GitHub repository.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, 2021.

Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

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 (ICCV)*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, volume 9908 of *Lecture Notes in Computer Science*, pp. 646–661. Springer, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, 2015.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository. https://archive.ics.uci.edu, n.d.

Asif R. Khan. CAP: A unified framework for learnable activation functions in artificial neural networks. SSRN preprint 6359188, 2026.

Haitham Khedr and Yasser Shoukry. Deepbern-nets: Taming the complexity of certifying neural networks using bernstein polynomial activations and precise bound propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 21232–21240, 2024.

Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32, 2019.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

Ziyao Li. Kolmogorov-arnold networks are radial basis function networks. *arXiv preprint arXiv:2405.06721*, 2024.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations*, 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*, 32, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Noam Shazeer. GLU variants improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Sidharth SS, Keerthana R, Gokul AR, and Anas KP. Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation. *arXiv preprint arXiv:2405.07200*, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357, 2021.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

Xingyi Yang and Xinchao Wang. Kolmogorov-arnold transformer. *arXiv preprint arXiv:2409.10594*, 2024.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Wentao Zhang, Yutong Zhang, Yifan Zhu, and Wentao Mo. Universal smoothness via Bernstein polynomials: A constructive approximation approach for activation functions. *arXiv preprint arXiv:2605.02591*, 2026.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.