Initialization Schemes for Kolmogorov–Arnold Networks: An Empirical Study

Spyros Rigas[1] · Dhruv Verma[2] · Georgios Alexandridis[1] · Yixuan Wang[2]

[1] Department of Digital Industry Technologies, National and Kapodistrian University of Athens

[2] Applied and Computational Mathematics, California Institute of Technology

ICLR 2026

Introduction

In a KAN layer, the layer's input, $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$, is related to its output, $\mathbf{y} \in \mathbb{R}^{n_{\text{out}}}$, via:

$$y_j = \sum_{i=1}^{n_{\text{in}}} \left(r_{ji}\, R\left(x_i\right) + c_{ji} \sum_{m=1}^{G+k} b_{jim}\, B_m\left(x_i\right) \right), \qquad (1)$$

where $B_m(x)$ are univariate spline basis functions of order k, defined on a grid with G intervals, and

$$R(x) = x \left(1 + e^{-x}\right)^{-1}$$

is the Sigmoid Linear Unit (SiLU) function operating as a residual function. The terms r_{ji}, c_{ji} and b_{jim} correspond to the layer's trainable parameters.

Motivation

Effective initialization is crucial in neural networks, as a good initial guess for the trainable parameters can significantly accelerate training [1] and prevent early saturation of hidden layers [2]. The original paper's [3] initialization scheme is:

- r_{ji} are initialized following Glorot initialization [2]
- $c_{ji} = 1$
- $b_{jim} \sim \mathcal{N}\left(0, \sigma^2\right)$, typically $\sigma = 0.1$

There is currently a gap in the study of initialization techniques for KANs.

Can we do better than this "default" initialization?

Proposed Initialization Schemes

We propose two theory-driven initializations, following the work of LeCun [4] and Glorot [2], as well as an empirical power-law scheme, for the residual-term and the spline basis-term coefficients. In all three cases, we assume:

$$r_{ji} \sim \mathcal{N}\left(0, \sigma_R^2\right), \qquad b_{jim} \sim \mathcal{N}\left(0, \sigma_B^2\right)$$

We further define:

$$\mu_R^{(0)} = \mathbb{E}\left[R(x)^2\right], \quad \mu_R^{(1)} = \mathbb{E}\left[R'(x)^2\right]$$

where the expectations are taken over the input distribution, and

$$\mu_B^{(0)} = \mathbb{E}\left[B_m(x)^2\right], \quad \mu_B^{(1)} = \mathbb{E}\left[B'_m(x)^2\right]$$

where the expectations are taken over both the input distribution and all spline basis indices.

Proposed Initialization Schemes · LeCun

Following LeCun et al. [4], we require that the variance of the output of a layer should match that of its input, for all layers within a KAN. This leads to:

$$\sigma_R \sim \frac{1}{\sqrt{n_{\text{in}}\,(G+k+1)\,\mu_R^{(0)}}}$$

for the residual-term coefficients and

$$\sigma_B \sim \frac{1}{\sqrt{n_{\text{in}}\,(G+k+1)\,\mu_B^{(0)}}}$$

for the basis-term coefficients. In these expressions we have assumed an equal contribution of the $G+k+1$ terms of Eq. (1) to the total variance.

Proposed Initialization Schemes · Glorot

If we attempt to balance the preservation of forward-pass variance (LeCun) and the preservation of backward-pass variance, we find the following Glorot-like initialization scheme:

$$\sigma_R \sim \sqrt{\frac{1}{G+k+1} \cdot \frac{2}{n_{\text{in}}\mu_R^{(0)} + n_{\text{out}}\mu_R^{(1)}}}$$

for the residual-term coefficients and

$$\sigma_B \sim \sqrt{\frac{1}{G+k+1} \cdot \frac{2}{n_{\text{in}}\mu_B^{(0)} + n_{\text{out}}\mu_B^{(1)}}}$$

for the basis-term coefficients.

Proposed Initialization Schemes · Power-law

Finally, we consider an initialization of the form:

$$\sigma_R \sim \left[\frac{1}{n_{\text{in}}\,(G+k+1)}\right]^{\alpha}$$

for the residual-term coefficients and

$$\sigma_B \sim \left[\frac{1}{n_{\text{in}}\,(G+k+1)}\right]^{\beta}$$

for the basis-term coefficients. The idea behind this initialization scheme is to perform a power-law scaling of the layer's architectural parameters, where the exponents α, β are intended to be tuned once per problem domain and used "out-of-the-box" for problems within each domain.

Power-law exponents tuning

For the purposes of this work, we solve two types of tasks: function fitting using a custom set of functions and forward PDEs within the PIML [5] framework. For each task we perform extensive grid-searches over (α, β) configurations, using different architectural settings (variations on grid size, hidden layer dimension and number of hidden layers). The identified optimal regions for the exponents are:

Function Fitting	**Forward PDE Problems**
$\alpha \in \{0.25, 0.5\}, \beta \geq 1.0$	$\alpha \in \{0.25, 0.5\}, 0.75 \leq \beta \leq 1.25$

For all subsequent experiments, we fix $\alpha = 0.25, \beta = 1.75$, which falls into the identified optimal region for function fitting, but outside the one for PDE problems, in order to demonstrate the robustness of the method.

Experimental Results - Training Stability

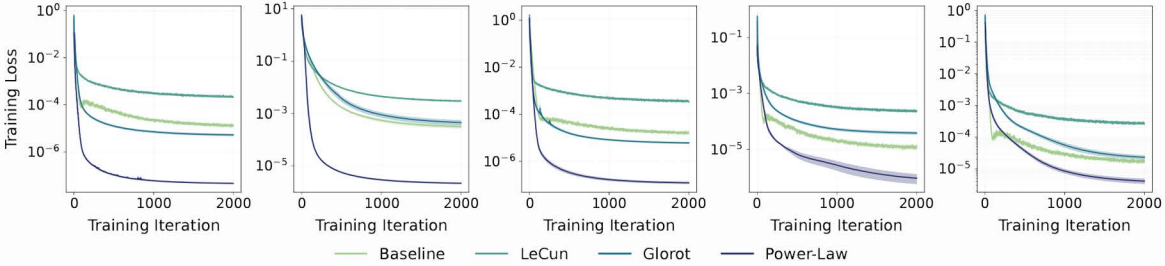

Figure 1: Training loss curves for different initializations on the task of custom function fitting.

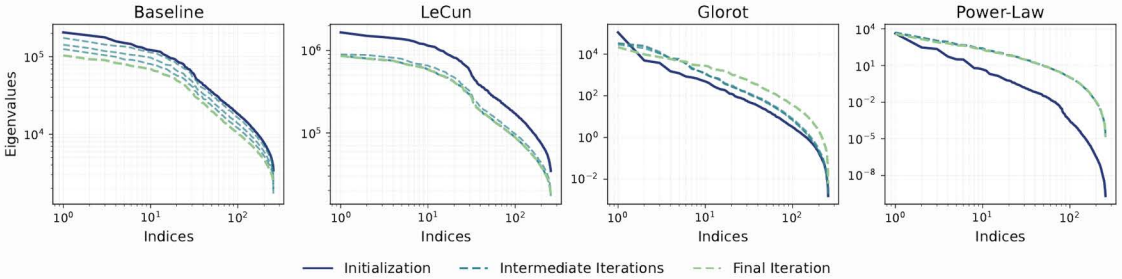

Figure 2: Neural Tangent Kernel dynamics for each initialization method.

Experimental Results - Training Stability

Key Takeaways:

- The LeCun-inspired initialization does not manage to reduce the training loss to the levels achieved by the other initialization schemes. Its NTK spectrum is dominated by large leading eigenvalues and collapses during training.
- The Glorot-inspired initialization appears to perform similarly to the default initialization scheme, but its NTK spectrum shows significant stability, whereas the NTK spectrum for the default scheme collapses during training.
- The power-law initialization scheme achieves the lowest training losses, while its NTK spectrum closely follows a power-law decay at initialization and remains perfectly stable during training.

Experimental Results - Accuracy

The previous results suggest that Glorot-based initialization is a strong alternative to the default method, while power-law initialization proves to be the most stable among those evaluated. Additional experiments on a subset of the Feynman dataset further reinforce these findings.

	Baseline		Glorot		Power-Law	
Function	Loss	L^2	Loss	L^2	Loss	L^2
I.6.2	$1.09 \cdot 10^{-3}$	$1.51 \cdot 10^{0}$	$4.80 \cdot 10^{-5}$	$4.19 \cdot 10^{-1}$	$\mathbf{5.20 \cdot 10^{-6}}$	$\mathbf{3.85 \cdot 10^{-1}}$
I.6.2b	$1.36 \cdot 10^{-3}$	$1.64 \cdot 10^{0}$	$7.60 \cdot 10^{-5}$	$5.80 \cdot 10^{-1}$	$\mathbf{2.18 \cdot 10^{-6}}$	$\mathbf{4.59 \cdot 10^{-1}}$
I.12.11	$1.64 \cdot 10^{-4}$	$3.77 \cdot 10^{-1}$	$3.00 \cdot 10^{-6}$	$1.47 \cdot 10^{-3}$	$\mathbf{2.16 \cdot 10^{-8}}$	$\mathbf{1.66 \cdot 10^{-4}}$
I.13.12	$2.70 \cdot 10^{3}$	$3.08 \cdot 10^{0}$	$2.81 \cdot 10^{3}$	$\mathbf{1.11 \cdot 10^{0}}$	$\mathbf{2.53 \cdot 10^{-1}}$	$5.49 \cdot 10^{0}$
I.16.6	$1.63 \cdot 10^{-4}$	$6.31 \cdot 10^{-1}$	$6.00 \cdot 10^{-6}$	$1.63 \cdot 10^{-2}$	$\mathbf{1.09 \cdot 10^{-6}}$	$\mathbf{1.48 \cdot 10^{-2}}$
I.18.4	$2.67 \cdot 10^{2}$	$\mathbf{1.00 \cdot 10^{0}}$	$1.53 \cdot 10^{3}$	$\mathbf{1.00 \cdot 10^{0}}$	$\mathbf{4.15 \cdot 10^{-2}}$	$\mathbf{1.00 \cdot 10^{0}}$
I.26.2	$1.01 \cdot 10^{-4}$	$1.10 \cdot 10^{0}$	$7.00 \cdot 10^{-6}$	$8.98 \cdot 10^{-3}$	$\mathbf{1.72 \cdot 10^{-7}}$	$\mathbf{1.25 \cdot 10^{-3}}$
I.27.6	$3.33 \cdot 10^{-3}$	$\mathbf{1.00 \cdot 10^{0}}$	$1.85 \cdot 10^{-4}$	$\mathbf{1.00 \cdot 10^{0}}$	$\mathbf{8.93 \cdot 10^{-5}}$	$\mathbf{1.00 \cdot 10^{0}}$
I.29.16	$2.01 \cdot 10^{-4}$	$4.45 \cdot 10^{-1}$	$1.20 \cdot 10^{-5}$	$6.28 \cdot 10^{-3}$	$\mathbf{2.06 \cdot 10^{-7}}$	$\mathbf{2.57 \cdot 10^{-3}}$
I.30.3	$1.18 \cdot 10^{-4}$	$7.72 \cdot 10^{-1}$	$1.00 \cdot 10^{-6}$	$2.92 \cdot 10^{-3}$	$\mathbf{2.17 \cdot 10^{-8}}$	$\mathbf{4.17 \cdot 10^{-4}}$
I.40.1	$2.26 \cdot 10^{-4}$	$6.70 \cdot 10^{-1}$	$5.00 \cdot 10^{-6}$	$3.39 \cdot 10^{-3}$	$\mathbf{1.41 \cdot 10^{-7}}$	$\mathbf{6.17 \cdot 10^{-4}}$
I.50.26	$2.03 \cdot 10^{-4}$	$4.38 \cdot 10^{-1}$	$2.00 \cdot 10^{-6}$	$1.50 \cdot 10^{-3}$	$\mathbf{3.70 \cdot 10^{-8}}$	$\mathbf{2.25 \cdot 10^{-4}}$
II.2.42	$1.52 \cdot 10^{-4}$	$6.86 \cdot 10^{-1}$	$4.00 \cdot 10^{-6}$	$2.62 \cdot 10^{-3}$	$\mathbf{8.54 \cdot 10^{-8}}$	$\mathbf{4.98 \cdot 10^{-4}}$
II.6.15a	$6.60 \cdot 10^{-5}$	$7.60 \cdot 10^{0}$	$2.00 \cdot 10^{-6}$	$5.47 \cdot 10^{-2}$	$\mathbf{8.13 \cdot 10^{-9}}$	$\mathbf{4.40 \cdot 10^{-3}}$
II.11.7	$1.75 \cdot 10^{-4}$	$9.78 \cdot 10^{-1}$	$1.10 \cdot 10^{-5}$	$1.01 \cdot 10^{-2}$	$\mathbf{1.80 \cdot 10^{-7}}$	$\mathbf{3.00 \cdot 10^{-3}}$
II.11.27	$8.80 \cdot 10^{-5}$	$1.04 \cdot 10^{0}$	$1.00 \cdot 10^{-6}$	$3.76 \cdot 10^{-3}$	$\mathbf{1.54 \cdot 10^{-7}}$	$\mathbf{1.95 \cdot 10^{-3}}$
II.35.18	$7.40 \cdot 10^{-5}$	$1.19 \cdot 10^{0}$	$6.00 \cdot 10^{-6}$	$1.18 \cdot 10^{-2}$	$\mathbf{2.95 \cdot 10^{-8}}$	$\mathbf{7.77 \cdot 10^{-4}}$
II.36.38	$1.93 \cdot 10^{-4}$	$9.48 \cdot 10^{-1}$	$8.00 \cdot 10^{-6}$	$1.11 \cdot 10^{-2}$	$\mathbf{3.05 \cdot 10^{-7}}$	$\mathbf{4.92 \cdot 10^{-3}}$
III.10.19	$1.81 \cdot 10^{-4}$	$2.74 \cdot 10^{-1}$	$1.00 \cdot 10^{-6}$	$9.89 \cdot 10^{-4}$	$\mathbf{9.87 \cdot 10^{-9}}$	$\mathbf{8.70 \cdot 10^{-5}}$
III.17.37	$1.45 \cdot 10^{-4}$	$9.10 \cdot 10^{-1}$	$4.90 \cdot 10^{-5}$	$1.31 \cdot 10^{-2}$	$\mathbf{6.45 \cdot 10^{-6}}$	$\mathbf{5.14 \cdot 10^{-3}}$

Table 1: Results on function fitting on a subset of the Feynman dataset.

Thank you!

References

[1] D. Mishkin and J. Matas, "All you need is a good init.", *ICLR*, 2016.

[2] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks.", *AISTATS*, pp. 249–256, 2010.

[3] Z. Liu et al., "KAN: Kolmogorov–Arnold networks.", *ICLR*, 2025.

[4] Y. LeCun, et al., "Efficient Backpop.", *Neural Networks: Tricks of the Trade*, pp. 9–50. Springer, 1998.

[5] M. Raissi, P. Perdikaris and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.", *J. Comput. Phys.*, 378:686–707, 2019.

Acknowledgement

S. R. and G. A. are supported by the Innovative Health Initiative Joint Undertaking (IHI JU) under grant agreement No 101253520. The JU receives support from the European Union's Horizon Europe research and innovation programme and COCIR, EFPIA, Europa Bío, MedTech Europe, and Vaccines Europe.