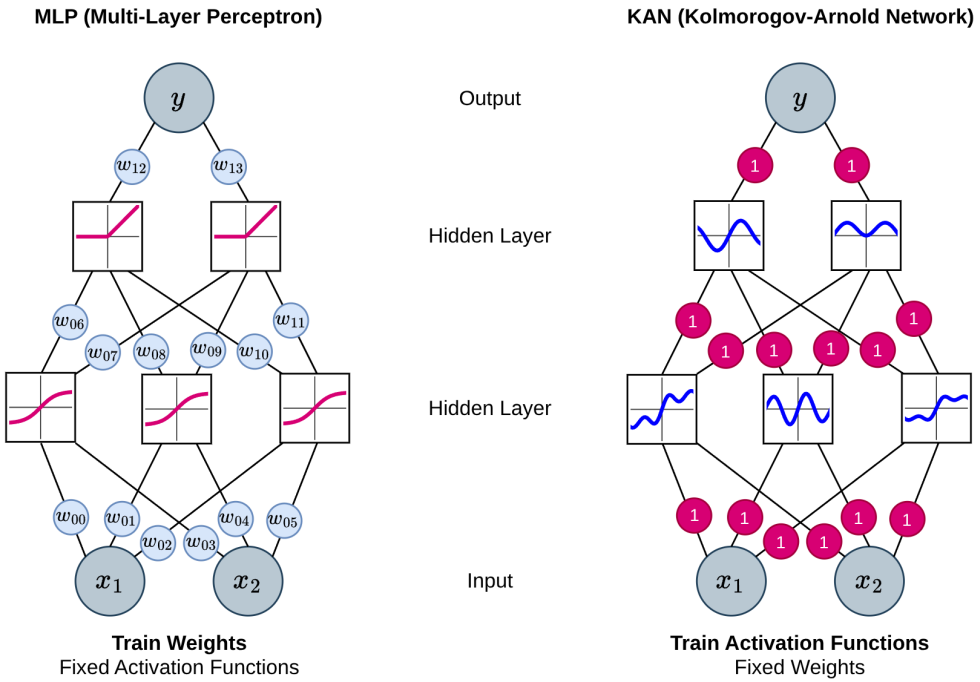
Table of Contents

Introduction to Kolmogorov-Arnold Networks

From fixed activations to learnable 1D edge functions



MLP vs KAN overview.

Motivation

Why KANs?

- Interpretability: learned **1D edge functions** can be inspected and simplified.
- Parameter efficiency on scientific tasks. [Liu+25]
- Better inductive bias when the target is smooth + compositional (common in physics/biology). [Liu+25]

From MLPs to KANs

Same layout, different nonlinearity

- MLP: scalar weights $w_{j,i}$ on edges; fixed activation σ on nodes.
- KAN: each edge is a learnable 1D function $\varphi_{j,i}(x)$ (e.g., a spline).
- Nodes just add inputs $\longrightarrow$ interpret learned functions directly.

From MLPs to Kolmogorov-Arnold Networks (KANs)

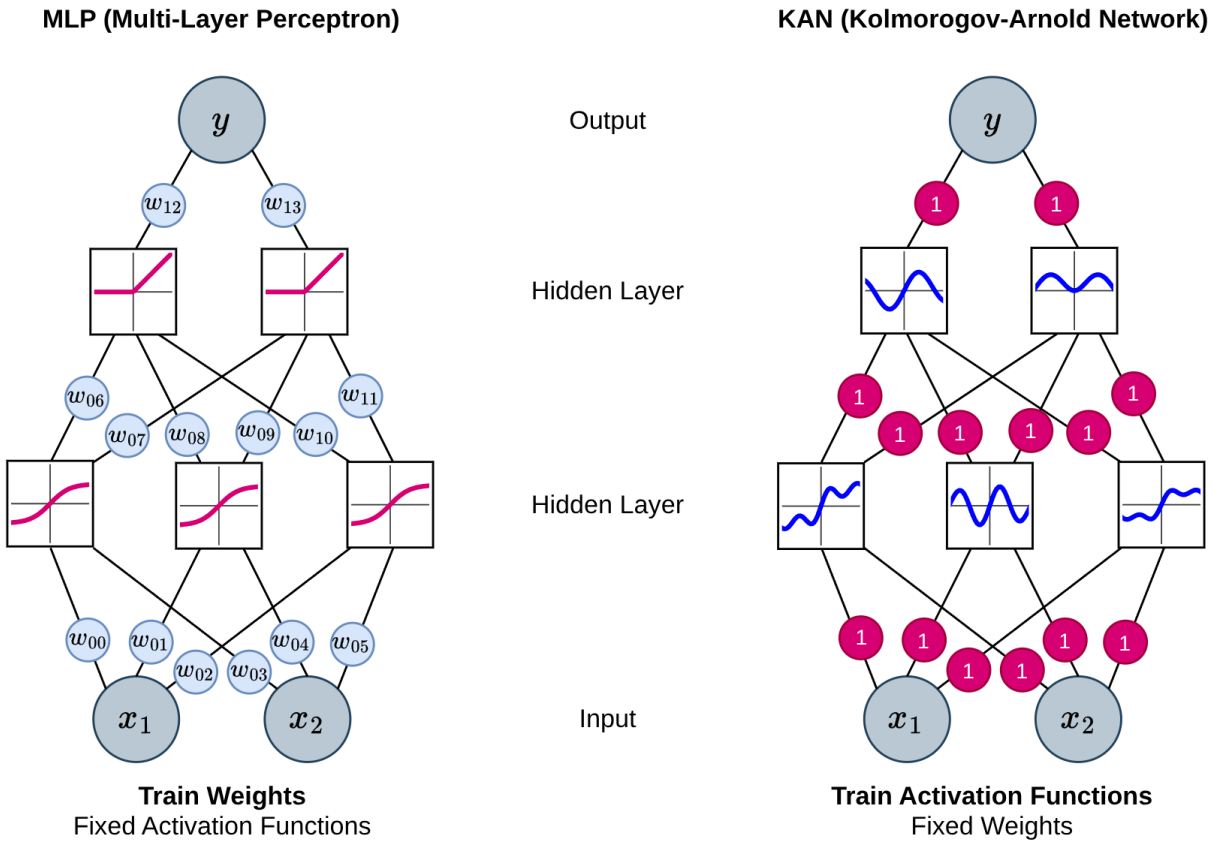
HM

Same wiring, different learnable object

Same basic layout as MLPs (fully connected layers):

- MLP: scalar weight $w_{i,j}$ on each edge, fixed activation σ on neurons.
- KAN: each edge has a learnable 1D function $\Phi_{i,j}(x)$ (e.g., spline).
  - Neurons just add inputs.

Introduction to Kolmogorov-Arnold Networks



MLP vs. KAN overview [Ser24]

The Kolmogorov-Arnold theorem (intuition)

Key statement

- Any continuous $f(x_1, ..., x_n)$ on $[0, 1]^n$ can be represented using **1D functions + addition**.
- The only true multivariate operation is **sum**; everything else can be composed from univariate transforms + additions.
- 1D functions can be approximated very well (e.g., with splines).

Takeaway: reduce multivariate learning to learning many 1D building blocks. [Kol57, Liu+25]

$$f(\boldsymbol{x}) = f(x_1, ..., x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$$

Kolmogorov-Arnold representation theorem. [Kol57, Liu+25]

The Kolmogorov-Arnold Theorem

Simple splines can approximate complex high-dimensional functions once the representation is compositional.

Key points

- Any continuous $f(x_1, ..., x_n)$ can be built from **1D functions + addition**.
- 1D functions can be approximated very well (e.g., with splines).

$$f(\boldsymbol{x}) = f(x_1, ..., x_n) = \sum_{q=1}^{2n+1} \Phi_{q\left(\sum_{p=1}^{n} \varphi_{q,p}(x_p)\right)}$$

[Kol57, Liu+25]

Can every function be learned in practice?

Caveat

- Classical KAT guarantees existence, but inner 1D functions can be highly non-smooth/fractal.
- Mitigation: go beyond the rigid depth-2, width $(2n+1)$ form $\longrightarrow$ use deeper/wider KANs.
- In many real tasks we expect smooth, compositionally sparse structure, making KAT-like forms learnable.

[Kol57, Liu+25]

Can any high-dimensional function be represented by KANs?

HM

Caveat + mitigation

- Classical KAT is elegant, but the required 1D inner functions can be non-smooth/fractal → hard to learn in practice.

Mitigation:

- Don't stick to the rigid depth-2, width $(2n + 1)$ form → use deeper/wider KANs to admit smoother representations.
- In real tasks we often expect smooth + compositionally sparse structure, so "typical cases" may admit smooth KA-like representations.

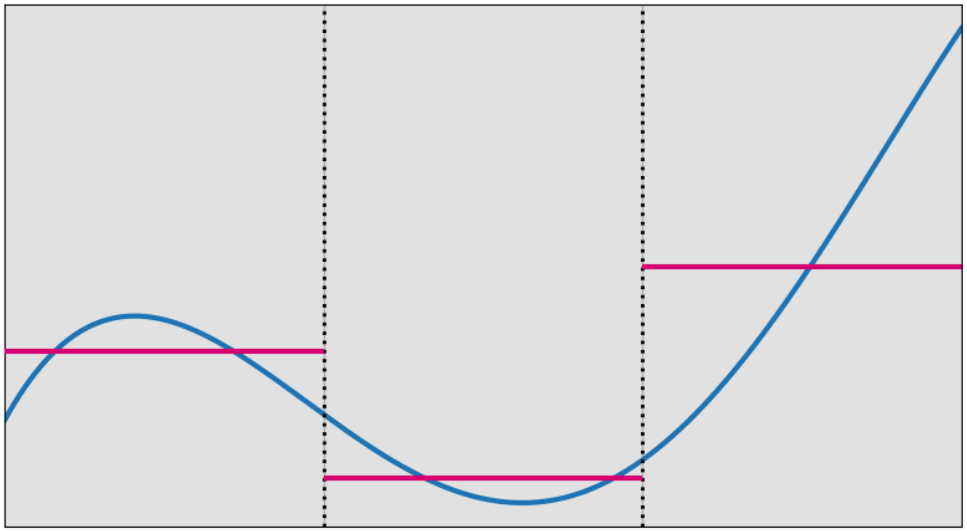
[Kol57, Liu+25]

Splines

Why splines?

- Smooth **piecewise-polynomial** functions of one variable x.
- Controlled by knots + coefficients $\rightarrow$ flexible local curves.
- Approximate 1D functions well with few parameters.

In KANs, each edge learns its own spline $\varphi_{j,i}(x)$. [Liu+25]

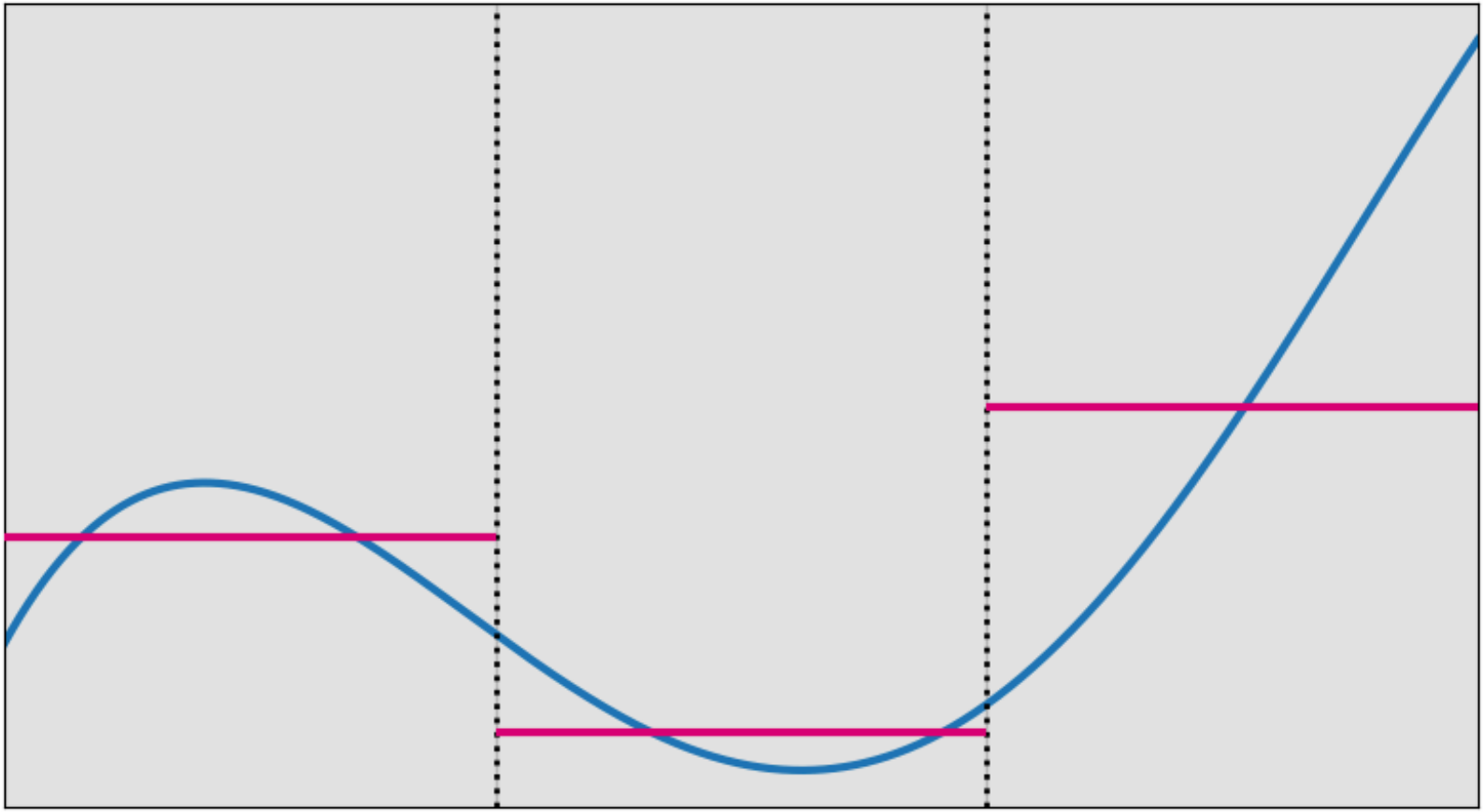


Spline example.

[De 78, Liu+25]

Splines (example)

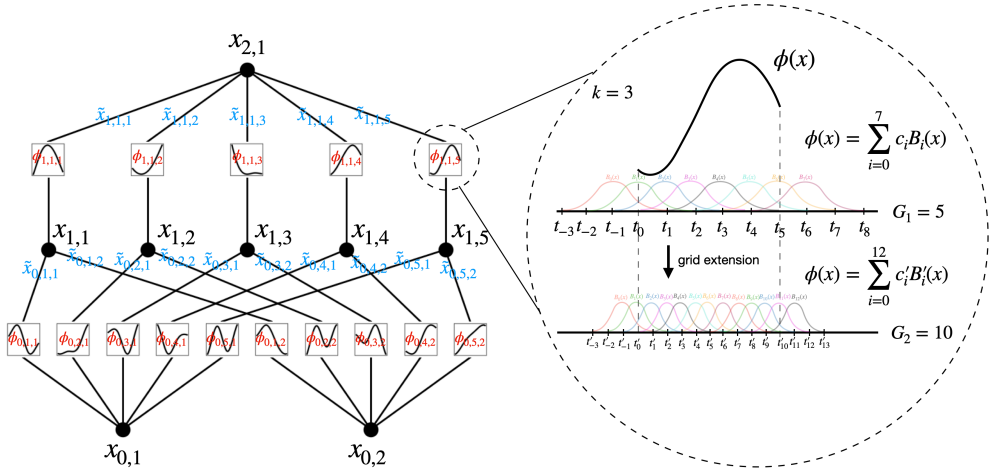
0.5
0.2
0.7



Splines (in KANs)

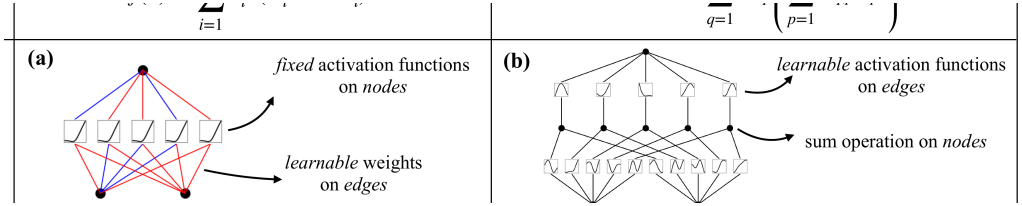
- Smooth **piecewise-polynomial** functions of one variable x
- Controlled by knots + coefficients → flexible local curves
- Approximate 1D functions very well with few parameters

In KANs, each edge learns its own 1D spline $\varphi_{i,j}(x)$. [Liu+25]



Spline notation and grid refinement. [Liu+25]

MLP vs KAN (shallow): where does nonlinearity live?



Shallow MLP vs shallow KAN (Fig. 0.1a,b). [Liu+25]

Shallow formulas

- MLP / UAT-style:

$$f(\boldsymbol{x}) \approx \sum_{j=1}^{m} a_j \sigma\left(\boldsymbol{w}_j^T \boldsymbol{x} + b_j\right)$$

- KAN / KAT-style:

$$f(\boldsymbol{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}\left(x_p\right) \right)$$

[Cyb89, HSW89, Kol57, Liu+25]

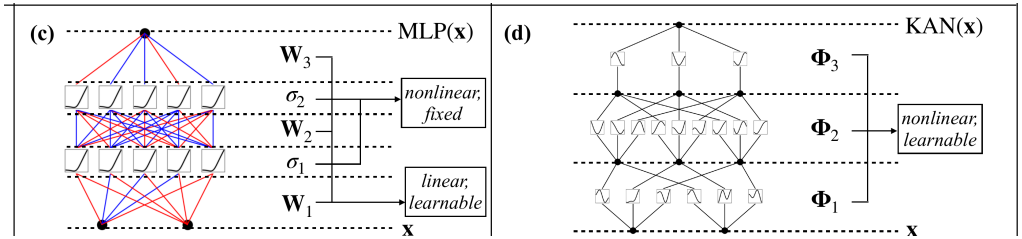
Connections: fixed vs learnable nonlinearity; inductive bias towards symbolic/compositional structure.

MLP vs KAN (shallow): where does nonlinearity live?

Key idea

- MLP: fixed activation σ on nodes; learn weights $w_{j,i}$ on edges.
- KAN: learn 1D edge functions $\varphi_{q,p}$; nodes only sum inputs.
- Intuition: learn expressive 1D building blocks, then compose across layers.
- Sum is the only multivariate operation (no explicit products).
- Example trick: $x \cdot y = \exp(\log x + \log y)$ shows how products can be expressed by univariate maps + addition.

MLP vs KAN (deep): what gets learned?



Deep MLP vs deep KAN (Fig. 0.1c,d). [Liu+25]

Deep takeaway

- Deep MLPs: learn linear maps $\boldsymbol{W}_l$; nonlinearity stays fixed.
- Deep KANs: learn function matrices $\boldsymbol{\Phi}_l$ (one 1D function per edge).
- Practical upside: plot/inspect learned edge functions directly.

Deep composition

- Deep MLP:

$$\mathrm{MLP}(\boldsymbol{x}) = (\boldsymbol{W}_{L-1} \circ \sigma \circ \boldsymbol{W}_{L-2} \circ \ldots \circ \sigma \circ \boldsymbol{W}_0)(\boldsymbol{x})$$

- Deep KAN:

$$\mathrm{KAN}(\boldsymbol{x}) = (\boldsymbol{\Phi}_{L-1} \circ \ldots \circ \boldsymbol{\Phi}_0)(\boldsymbol{x})$$

[Cyb89, HSW89, Liu+25]

Interpretation

- MLP: learn linear maps $\boldsymbol{W}$; nonlinearity is fixed.
- KAN: learn edge functions $\varphi_{l,j,i}$; nodes are sums.

Curse of dimensionality: where the pain shows up

MLPs (UAT): existence → not efficiency

- UAT is an **existence** guarantee: a shallow MLP can approximate any continuous f on a compact set.
- It does **not** guarantee dimension-free efficiency: in worst cases, required width / samples can grow exponentially with dimension.
- Deep nets often help by exploiting **structure** (compositionality / low intrinsic dimension), but this is not automatic.

KANs (KART): shift the difficulty

- KART suggests reducing multivariate learning to learning many 1D functions.
- But the classical 2-layer representation can require **highly non-smooth / fractal** inner functions → hard to learn.
- Deep/wide KANs assume a **smooth compositional** representation exists; then each edge is a learnable 1D problem.

> KANs do not delete CoD; they try to **negotiate** with it by betting on smooth, compositional structure.

Curse of dimensionality: where the pain shows up

> CoD for MLPs typically bites via **sample/ width blow-up** when f has no exploitable structure.

Curse of dimensionality: where the pain shows up

UAT (MLPs)

- Statement: 2-layer nets can approximate any continuous f on a compact domain.
- Learnable parts: $\boldsymbol{W}, \boldsymbol{b}$ (activations fixed).
- Caveat: existence result; rates can still suffer from dimensionality.

$$f(\boldsymbol{x}) \approx \sum_{j=1}^{m} a_j \sigma\left(\boldsymbol{w}_j^T \boldsymbol{x} + b_j\right)$$

[Cyb89, HSW89]

KAT (Kolmogorov-Arnold)

- Statement: represent $f : [0,1]^n \to \mathbb{R}$ via sums of 1D functions + addition.
- Promise: reduce multivariate learning to learning many 1D functions.
- Caveat: worst-case representations can be highly non-smooth/fractal.

$$f(\boldsymbol{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}\left(x_p\right) \right)$$

[BG09, GP89, Kol57, Liu+25, Sch21]

KAN viewpoint: assume smooth/compositional structure; learn φ with splines and add depth to avoid pathological 2-layer forms. [Liu+25, PBL20]

KAN layer mechanics

Each layer is a matrix of learnable 1D functions:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \varphi_{l,j,i}(x_{l,i})$$

$$\boldsymbol{x}_{l+1} = \boldsymbol{\Phi}_l \boldsymbol{x}_l$$

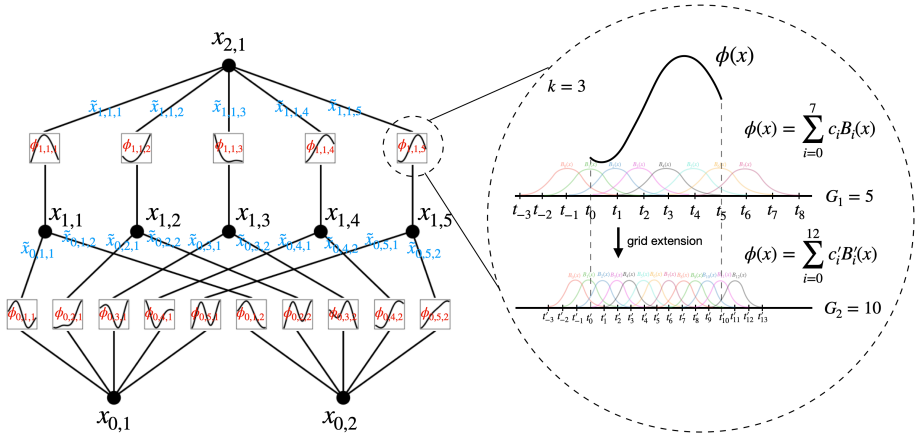
[Liu+25]

Each edge function is a residual spline:

$$\varphi(x) = w_b b(x) + w_s \sum_i c_i B_{i(x)}$$

[De 78, Liu+25]

Residual $b(x)$ defaults to SiLU; spline coefficients are trainable. Local B-spline basis → localized updates → potentially less catastrophic forgetting (continual learning intuition).



B-spline parametrization and grid refinement. [Liu+25]

Training and optimization tricks

Optimization details

- Residual activation: $\varphi(x) = w_b b(x) + w_s \ \mathrm{spline}(x)$ (keeps outputs well-defined even if inputs drift outside the spline grid).
- Grid update: periodically estimate the activation distribution and **move knot/grid points** to maintain good coverage.
- Grid updates are **non-differentiable** (a data-driven reparameterization step).
- B-splines are a practical choice (locality), but KAN ≠ splines: other bases / global activations are possible.

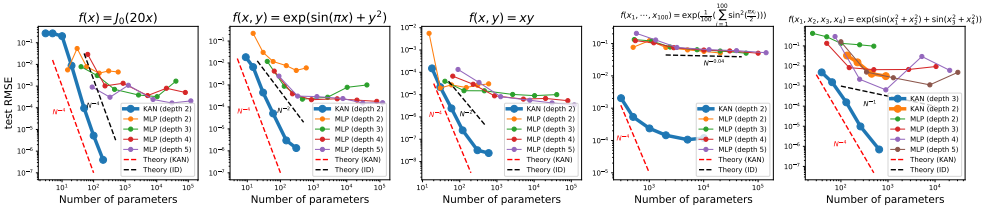
Connections

- Initialization and stability in deep networks
- Loss landscapes and optimization schedules
- Bias-variance trade-off when increasing capacity

[Liu+25]

Accuracy & Scaling

How KANs generalize and grow

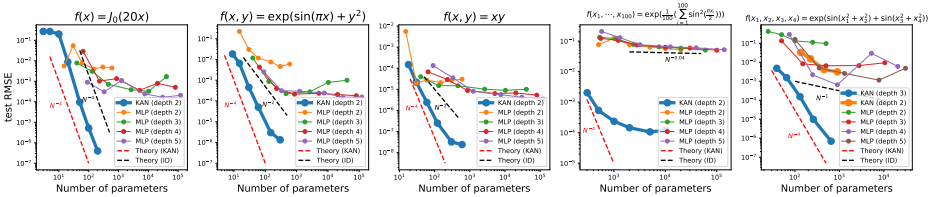


Fast scaling trends on structured function classes. [Liu+25]

Scaling laws

Theory + observation

- Smooth-KAT bound: $|f - \mathrm{KAN}_G| \leq CG^{-(k+1)}$ (cubic: $k = 3 \to \alpha \approx 4$).
- Comparison: manifold view ($\alpha \approx \frac{k+1}{d}$) vs arity view ($\alpha \approx \frac{k+1}{2}$).
- Empirically: KANs reach steeper scaling than MLPs on compositional data.
- Caveat: this advantage assumes the target admits a **smooth compositional** KAN/KAR; we usually do not know this structure a priori.



Scaling vs MLP baselines. [Liu+25]

[De 78, Liu+25, MLT23, SK20]

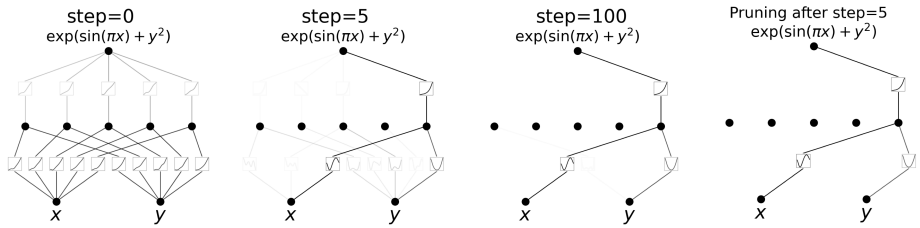
Connections: scaling laws; approximation theory; bias-variance trade-off.

Why symbolic extraction matters (beyond “post-hoc”)

From fit → formula

- Goal: not only predict, but **compress** knowledge into a symbolic law.
- This turns supervised learning into a form of **scientific discovery**: we obtain an explicit equation that can be checked, generalized, and reused.
- KANs help because intermediate artifacts are readable: 1D edge functions $\varphi(c\,\cdot)$.

> Interpretability is not an afterthought here; it is an explicit objective.



From dense model → sparse graph → symbolic form (schematic).
[Liu+25]

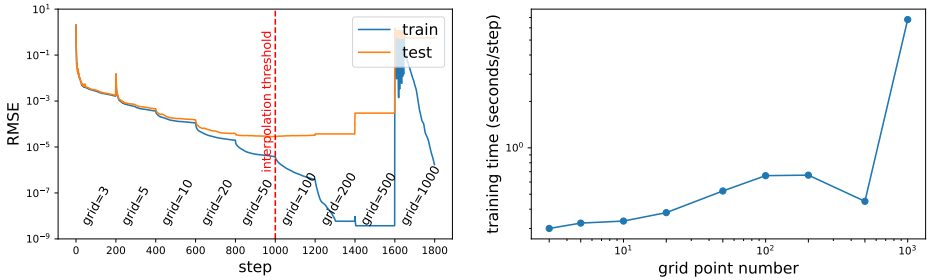
Grid extension: fine-grain without retraining

Key idea

- Start with coarse grids, then refine spline knots.
- Initialize finer grids by least-squares fit to the coarse spline.
- Produces staircase-like drops in loss after each extension.
- Improves accuracy without retraining a larger model from scratch.

[Liu+25]

Connections: model scaling vs training schedules; KAN adds explicit fine-graining.



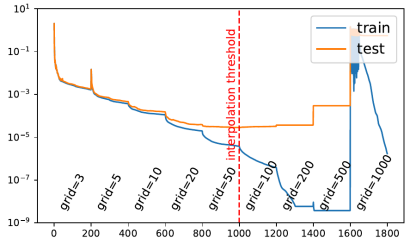
Grid extension illustration. [Liu+25]

Grid extension: why it works

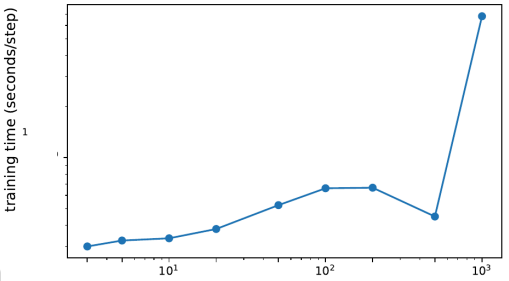
Why grid extension works

- External dofs: graph structure (width/depth) learns compositional structure.
- Internal dofs: spline grid points learn 1D functions precisely.
- Grid extension: increase internal dofs without re-initializing.
- Warm-start: least-squares fit a finer spline to the coarse spline (per edge).
- Effect: staircase-like loss drops after each refinement; cost grows with grid size.

[Liu+25]



Staircase loss drops after each refinement. [Liu+25]



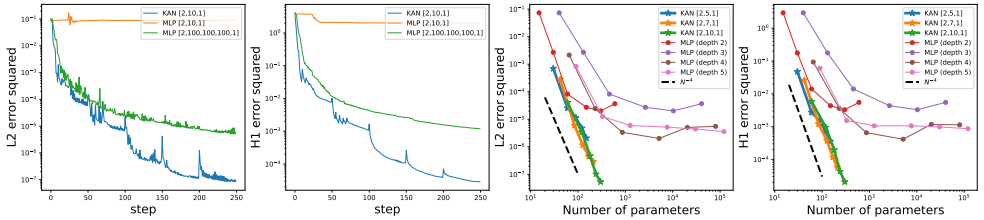
Training time vs grid size. [Liu+25]

Accuracy results: PDEs and scientific fitting

Results

- PDE solving: Poisson equation solved with smaller KANs at higher accuracy.
- Special functions + Feynman datasets show strong sample efficiency.
- Suggests KANs as compact, high-precision function approximators.

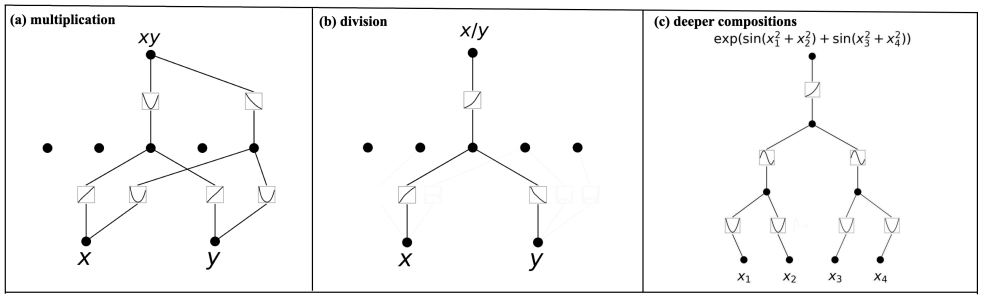
[Liu+25]



PDE benchmark results. [Liu+25]

Interpretability & Science

From pruning to symbolic laws



Symbolic recovery examples from pruned/simplified KANs. [Liu+25]

Interpretability toolkit: sparsify, prune, symbolify

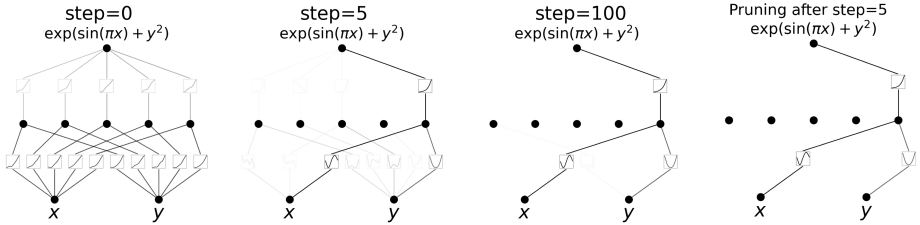
Four steps to a formula

- Sparsify: encourage few active edges (L1 + entropy).
- Visualize: inspect learned 1D edge functions $\varphi_{l,j,i}$.
- Prune: drop inactive nodes to a minimal shape $[n_0, ..., n_L]$.
- Symbolify: snap splines to analytic forms with an affine wrapper

$$y \approx cf(ax + b) + d$$

(grid search for a, b; linear regression for c, d).

[Liu+25]



Sparsification + pruning yields simpler, more interpretable KANs.
[Liu+25]

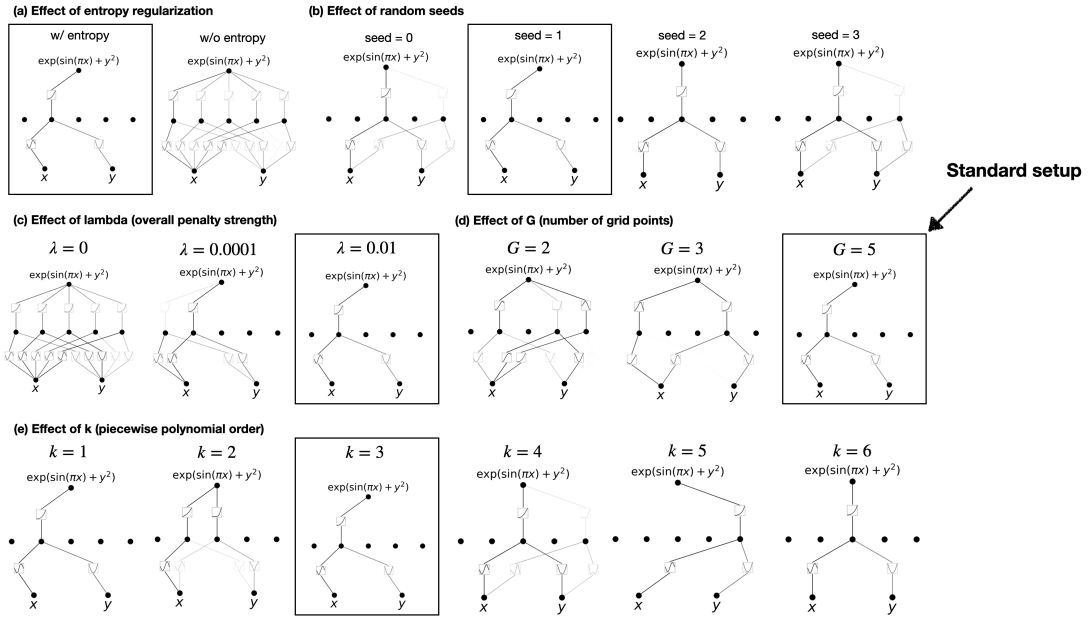
Interpretability: hyperparameters matter

What changes (and why)

- Entropy regularization: encourages sparse, readable graphs.
- λ: sparsity-accuracy trade-off; too small $\rightarrow$ dense, too large $\rightarrow$ underfit.
- Grid size G + spline order k: resolution vs compute (larger G is slower).
- Random seeds can reveal different relations in unsupervised discovery.

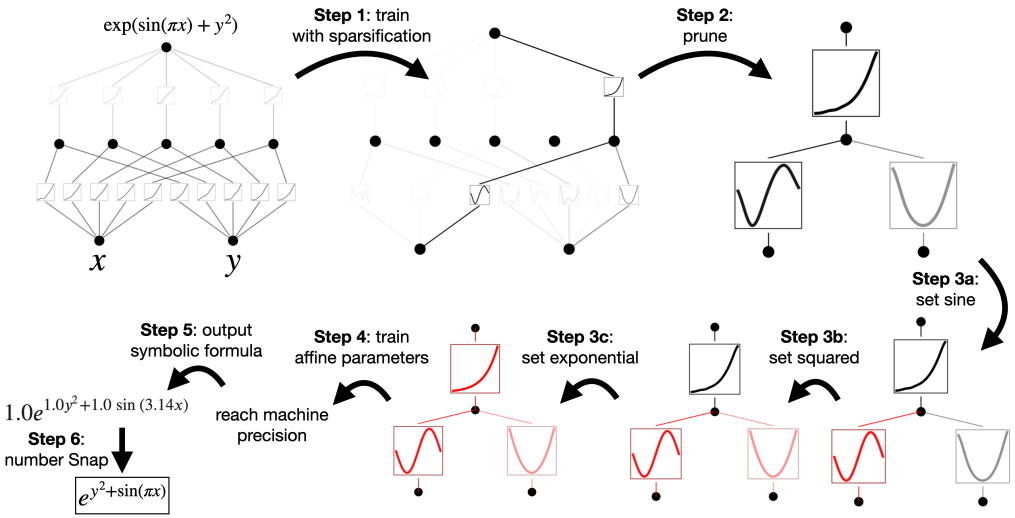
[Liu+25]

Takeaway: interpretability is an objective + design choice, not a byproduct.



Dependence on regularization, seeds, and spline resolution. [Liu+25]

Interactive symbolification (toy example)



Interactive workflow for symbolic regression with KANs. [Liu+25]

What the user does

- Train with sparsification.
- Prune to a minimal graph.
- Set/suggest symbolic forms (manual or assisted).
- Retrain only affine parameters and export the symbolic formula.

[Liu+25]

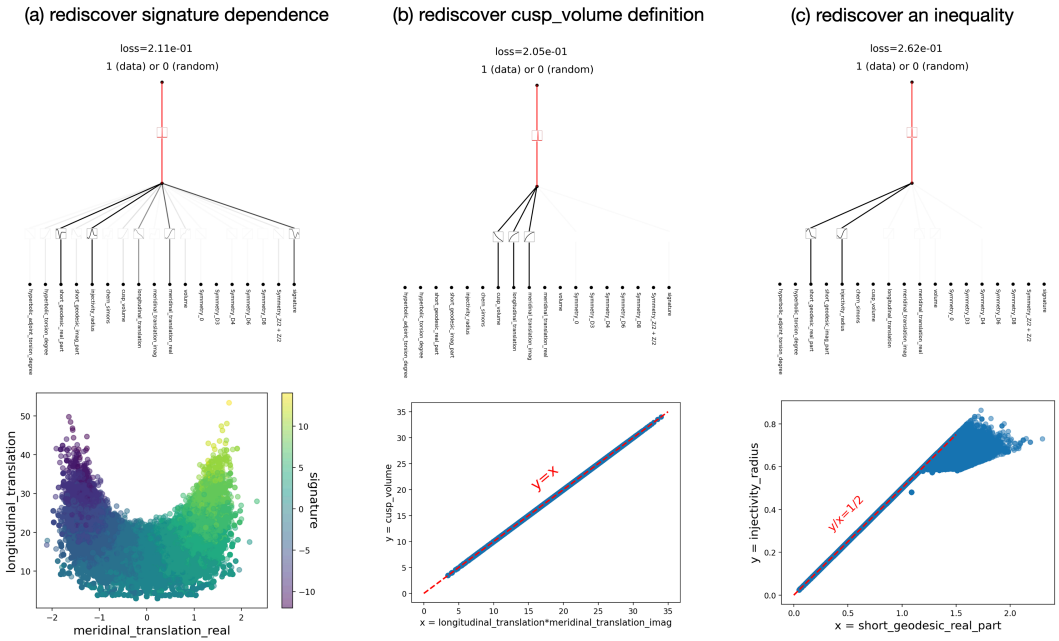
Case study (math): knot invariants (unsupervised)

Unsupervised discovery idea

- Goal: discover sparse relations among many invariants (not just predict one target).
- Train a sparse classifier KAN (shape $[18, 1, 1]$).
- Fix the last activation to a Gaussian peak at 0 $\Longrightarrow$ positives satisfy

$$\sum_{i=1}^{18} g_{i(x_i)} \approx 0$$

(read g_i off learned edges).
- Sweep seeds + λ and cluster multiple discovered relations.



Knot dataset (unsupervised): rediscovered relations. [Dav+21, Guk+23, Liu+25]

Case study (physics): mobility edges via KANs

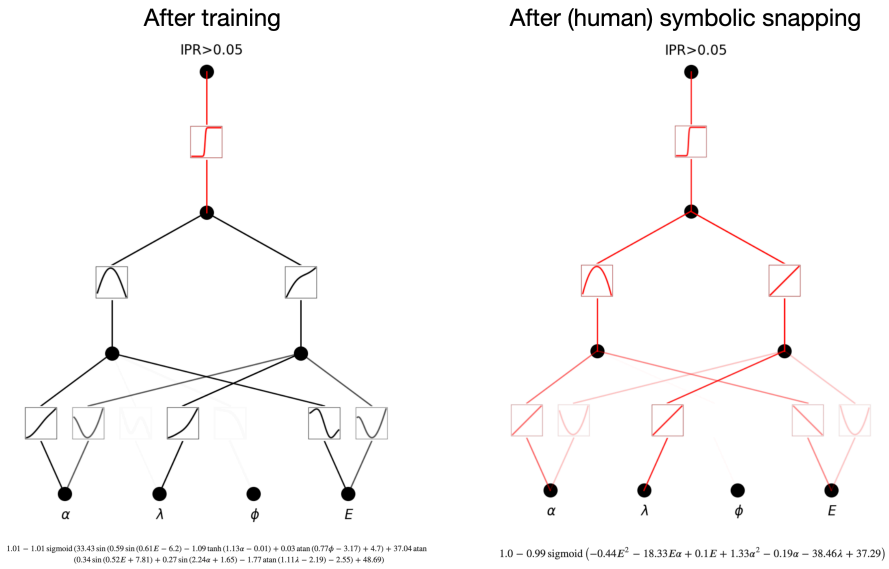
From data to an order parameter

- Goal: learn the mobility edge separating localized vs extended phases.
- Localization metric (eigenstate ψ^k):

$$\mathrm{IPR}_k = \frac{\sum_n |\psi_n^k|^4}{\left(\sum_n |\psi_n^k|^2\right)^2}$$

$$D_k = -\frac{\log(\mathrm{IPR}_k)}{\log(N)}$$

- Train $\longrightarrow$ sparsify/prune $\longrightarrow$ symbolify to recover a compact boundary $g(\cdot) = 0$ (human-in-the-loop: constrain the symbol library).



Mobility-edge discovery before/after symbolic snapping. [And58, Liu+25]

Symbolic regression: KANs vs classic SR

Why KANs help

- Continuous search in function space (gradients) before snapping to symbols.
- Debuggable intermediate artifacts: plots of $\varphi_{l,j,i}$.
- Works even when the target is not exactly symbolic (splines as fallback).

[Liu+25]

Related SR methods

- Genetic / heuristic: Eureqa [Dub11]
- Physics-inspired: AI Feynman [Udr+20, UT20]
- NN-based: EQL [ML16], OccamNet [Dug+20]
- Program search: PySR [Cra23]

[Liu+25]

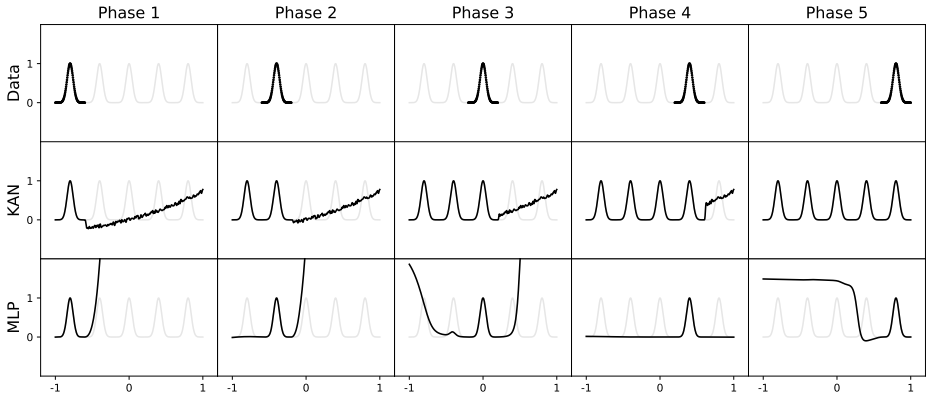
Continual learning and locality

Local plasticity

- B-spline activations are local in input space.
- Updates can be localized, reducing catastrophic forgetting.
- Promising for continual or lifelong learning regimes.
- Trade-off: locality can be computationally expensive; global bases may be faster but lose locality.

[Liu+25]

Connections: catastrophic forgetting; local adaptation; compute reuse in continual settings.



Continual learning experiments. [Liu+25]

Limitations and open questions

Practical limits

- Training is slower (poor batching; no optimized spline kernels). [Liu+25]
- Scaling claims are strongest on structured, low-data scientific tasks.
- Choosing minimal KAN shapes is still an open design problem (we usually don't know the target's compositional structure).
- Can KANs replace MLP blocks in CNNs/Transformers without hardware regressions?

Connections: throughput vs parameter count; hardware efficiency vs expressivity.

Summary + discussion prompts

Takeaways

- KANs move nonlinearity to edges, learning 1D functions directly.
- Grid extension + spline parametrization yield strong accuracy/scaling.
- Sparsification enables symbolic interpretability (white-box ML).
- Trade-off: better accuracy/interpretability vs slower training.

Questions for the track

- Where would KANs beat MLPs (e.g., scientific regression, PDEs, transformer MLP blocks)?
- What hardware/software advances would make KANs practical?
- How should we evaluate interpretability vs performance for science?

References

[Liu+25] Z. Liu et al., “KAN: Kolmogorov-Arnold Networks.” [Online]. Available: https://arxiv.org/abs/2404.19756

[Ser24] Serrano.Academy, “Kolmogorov-Arnold Networks (KANs) - What are they and how do they work?.” Accessed: Dec. 21, 2025. [Online]. Available: https://www.youtube.com/watch?v=myFtp5zMv8U

[Kol57] A. N. Kolmogorov, “On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition,” in Doklady Akademii Nauk, 1957, pp. 953–956.

[De 78] C. De Boor, A practical guide to splines, vol. 27. springer-verlag New York, 1978.

[Cyb89] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of control, signals and systems, vol. 2, no. 4, pp. 303–314, 1989.

[HSW89] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal

References

approximators," Neural networks, vol. 2, no. 5, pp. 359–366, 1989.

[BG09] J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," Constructive approximation, vol. 30, pp. 653–675, 2009.

[Sch21] J. Schmidt-Hieber, "The Kolmogorov–Arnold representation theorem revisited," Neural networks, vol. 137, pp. 119–126, 2021.

[GP89] F. Girosi and T. Poggio, "Representation properties of networks: Kolmogorov's theorem is irrelevant," Neural Computation, vol. 1, no. 4, pp. 465–469, 1989.

[PBL20] T. Poggio, A. Banburski, and Q. Liao, "Theoretical issues in deep networks," Proceedings of the National Academy of Sciences, vol. 117, no. 48, pp. 30039–30045, 2020.

[SK20] U. Sharma and J. Kaplan, "A neural scaling law from the dimension of the data manifold," arXiv preprint arXiv:2004.10802, 2020.

[MLT23] E. J. Michaud, Z. Liu, and M. Tegmark, "Precision machine

References

learning," Entropy, vol. 25, no. 1, p. 175, 2023.

[Dav+21] A. Davies et al., "Advancing mathematics by guiding human intuition with AI," Nature, vol. 600, no. 7887, pp. 70–74, 2021.

[Guk+23] S. Gukov, J. Halverson, C. Manolescu, and F. Ruehle, "Searching for ribbons with machine learning." 2023.

[And58] P. W. Anderson, "Absence of diffusion in certain random lattices," Physical review, vol. 109, no. 5, p. 1492, 1958.

[Dub11] R. Dubcáková, "Eureqa: software review," Genetic Programming and Evolvable Machines, vol. 12, pp. 173–178, 2011, [Online]. Available: https://api.semanticscholar.org/CorpusID:36698573

[UT20] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," Science Advances, vol. 6, no. 16, p. eaay2631, 2020.

[Udr+20] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark, "AI Feynman 2.0: Pareto-optimal

References

symbolic regression exploiting graph modularity," Advances in Neural Information Processing Systems, vol. 33, pp. 4860–4871, 2020.

[ML16] G. Martius and C. H. Lampert, "Extrapolation and learning equations," arXiv preprint arXiv:1610.02995, 2016.

[Dug+20] O. Dugan et al., "OccamNet: A Fast Neural Model for Symbolic Regression at Scale," arXiv preprint arXiv:2007.10784, 2020.

[Cra23] M. Cranmer, "Interpretable machine learning for science with PySR and SymbolicRegression.jl," arXiv preprint arXiv:2305.01582, 2023.