KAN 2.0: Kolmogorov–Arnold Networks Meet Science

Deep Learning for the Natural Sciences

Jonathan Wheeler

Table of contents

1

Motivation

Current Landscape

The AI + Science Landscape

Current State

- AI is transformational
- Deep learning dominates current approaches

Major Achievements

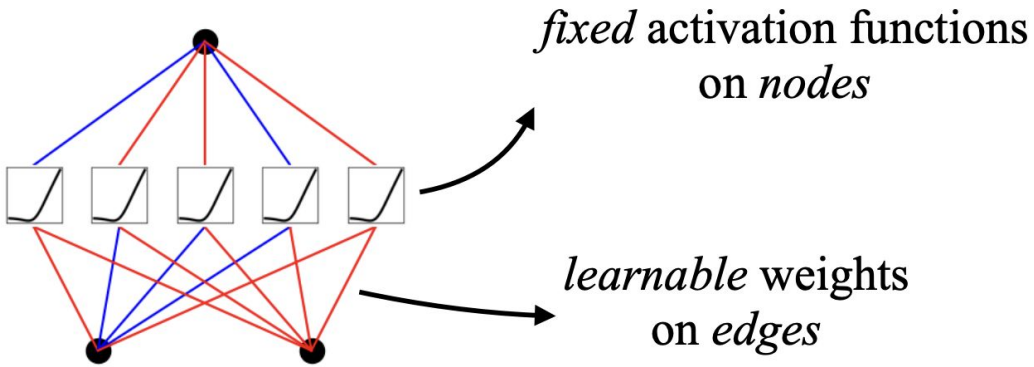
- **AlphaFold:** Near-perfect protein structure prediction
- **AI Weather Forecasting:** Beyond traditional accuracy

Remaining Challenges

- Limited scientific understanding from AI systems
- Need for more transparent AI approaches

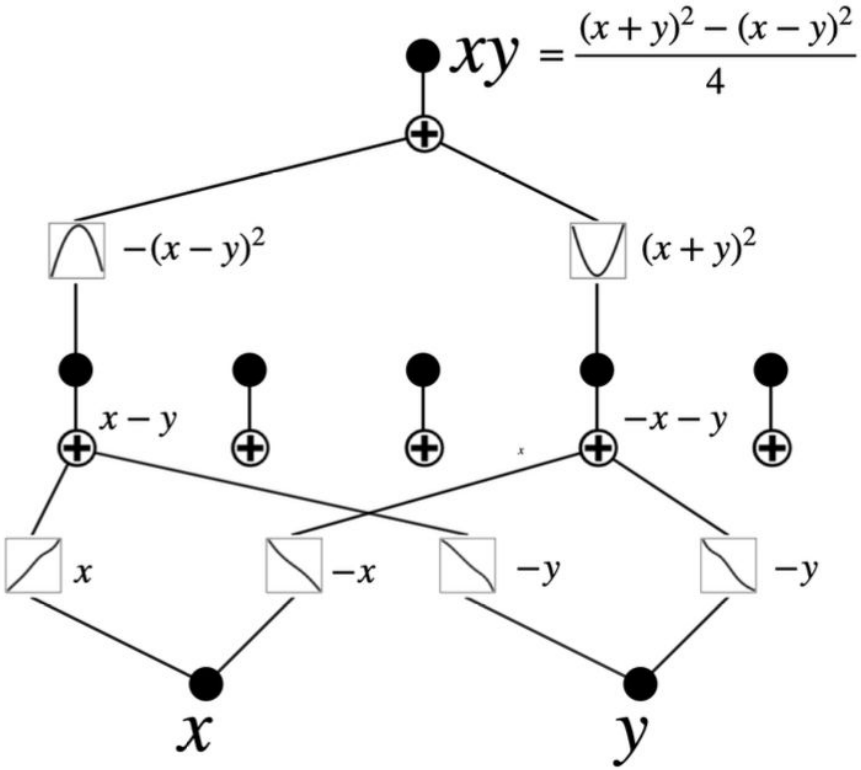
Application-driven Science (MLP)

- **What (is the output)?**
- Works for well formulated problems with clear objectives
- Less interpritible MLP models



Curiosity-driven Science (KAN)

- **Why (is this the result)?**
- Scientific discoveries require mathematical understanding and clear principles
- KAN's offer more interpretable structure through edge functions



2

Understanding KANs

Theoretical introductions

The Kolmogorov–Arnold Theorem

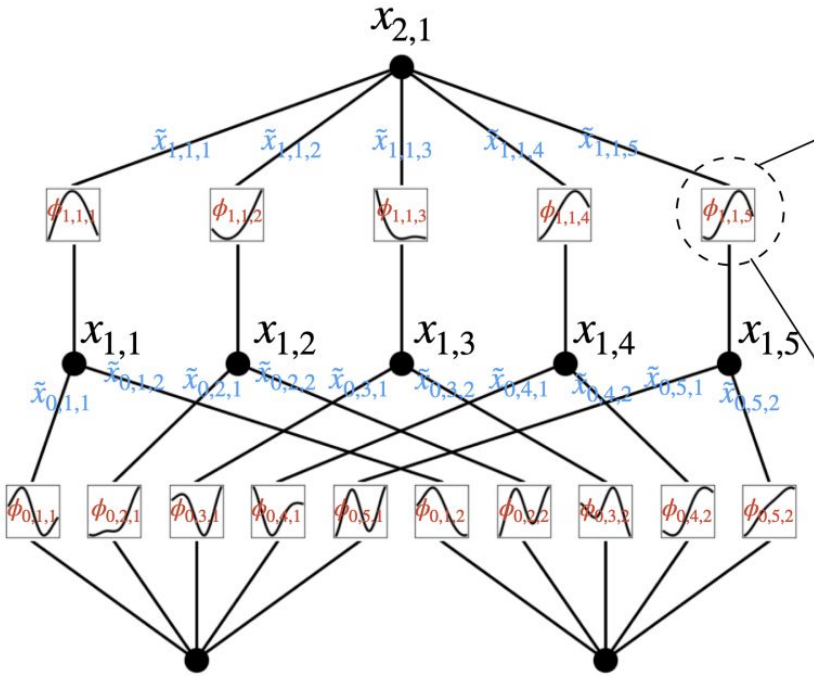
$$f(\mathbf{x}) = f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

- States that **any continuous function in a high-dimensional space can be represented as a finite composition of univariate continuous functions and additions**.
- Example:

$$xy = exp(logx + logy)$$

Two-layer Network

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



Key problem is has to be smooth functions

Key Breakthrough

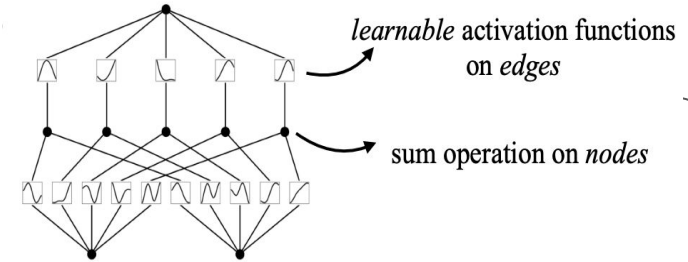
$$\mathbf{x}_{l+1} = \underbrace{\begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,2,1}(\cdot) & \cdots & \phi_{l,n_l,1}(\cdot) \\ \phi_{l,1,2}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,n_l,2}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,1,n_{l+1}}(\cdot) & \phi_{l,2,n_{l+1}}(\cdot) & \cdots & \phi_{l,n_l,n_{l+1}}(\cdot) \end{pmatrix}}_{\mathbf{\Phi}_l} \mathbf{x}_l$$

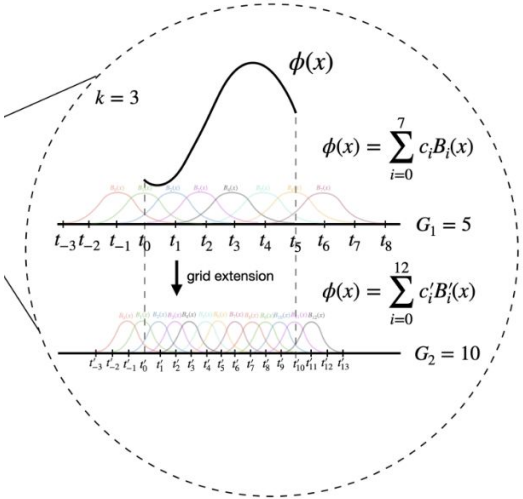
Generalizing the original Kolmogorov-Arnold representation, extending it to arbitrary depths

$$f(\mathbf{x}) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_L,i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \cdots \left(\sum_{i_2=1}^{n_2} \phi_{2,i_3,i_2} \left(\sum_{i_1=1}^{n_1} \phi_{1,i_2,i_1} \left(\sum_{i_0=1}^{n_0} \phi_{0,i_1,i_0}(x_{i_0}) \right) \right) \right) \cdots \right)$$

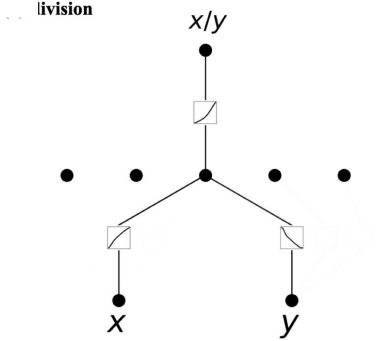
Basic KAN Architecture

- nodes (summation)
- edges (learnable activation functions)
  - B-Splines
- Symbolic regression gives output formula





$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_3 \circ \mathbf{\Phi}_2 \circ \mathbf{\Phi}_1)(\mathbf{x})$$



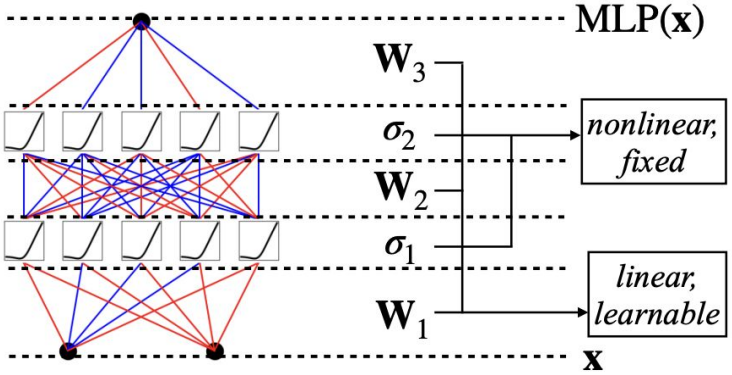
Comparison

MLPs:

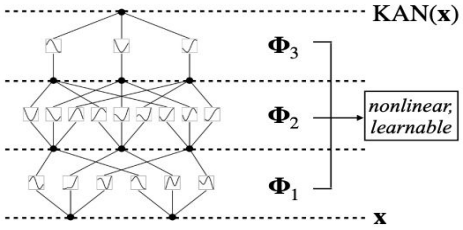
- Universal representation theorem based
- fixed nonlinear activation
- Each neuron processes weighted sum of inputs

KANs:

- Kolmogorov-Arnold representation based
- Edge-based univariate transformations
- Node-wise signal summation



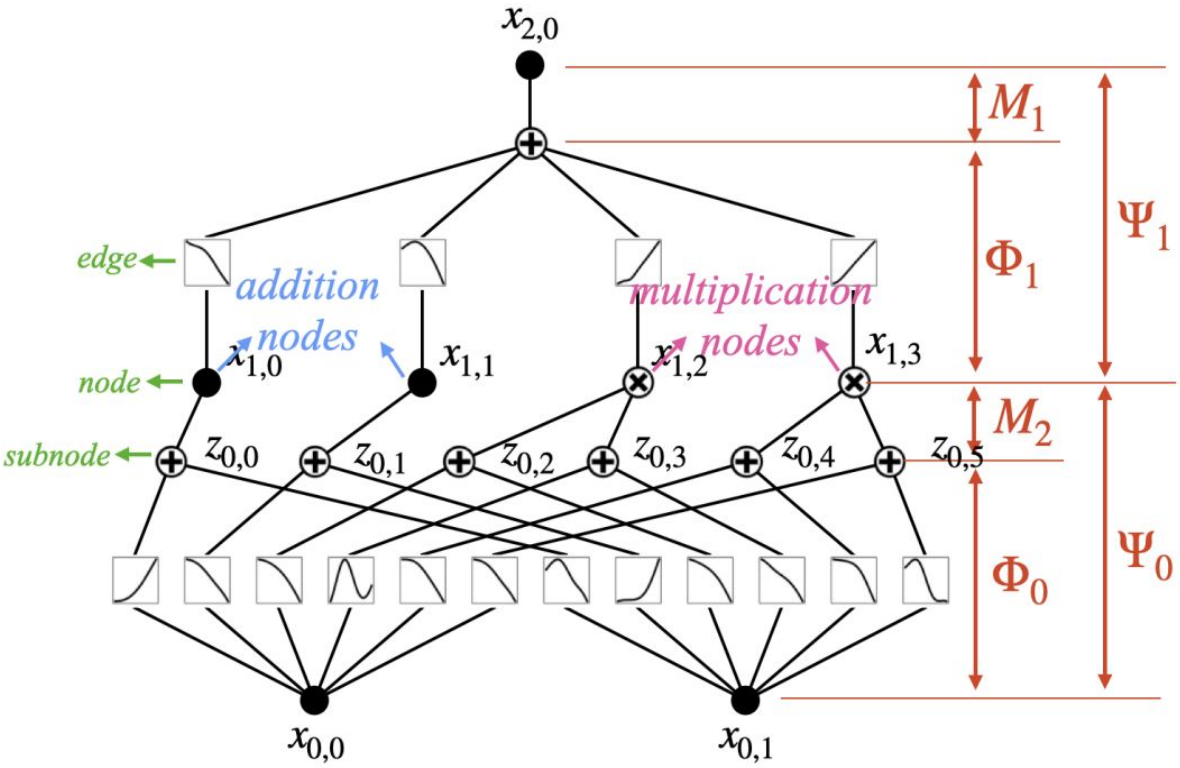
$$\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_{L-1} \circ \sigma \circ \mathbf{W}_{L-2} \circ \sigma \circ \cdots \circ \mathbf{W}_1 \circ \sigma \circ \mathbf{W}_0)\mathbf{x}.$$



$$\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_{L-1} \circ \mathbf{\Phi}_{L-2} \circ \cdots \circ \mathbf{\Phi}_1 \circ \mathbf{\Phi}_0)\mathbf{x}.$$

MultKAN

- Nodes(input) and subnodes(output)
- Standard KAN layer plus multiplication layer
- No additional trainable parameters
- Learns multiplication



$$\mathrm{MultKAN}(\mathbf{x}) = (\mathbf{\Psi}_L \circ \mathbf{\Psi}_{L-1} \circ \cdots \circ \mathbf{\Psi}_1 \circ \mathbf{\Psi}_0)\mathbf{x}.$$

Goal:

Problem: symbolic formulas are not always possible

Solution:

1. Build in scientific knowledge to KANs (section 3)

2. Extract out scientific knowledge from KANs (section 4)

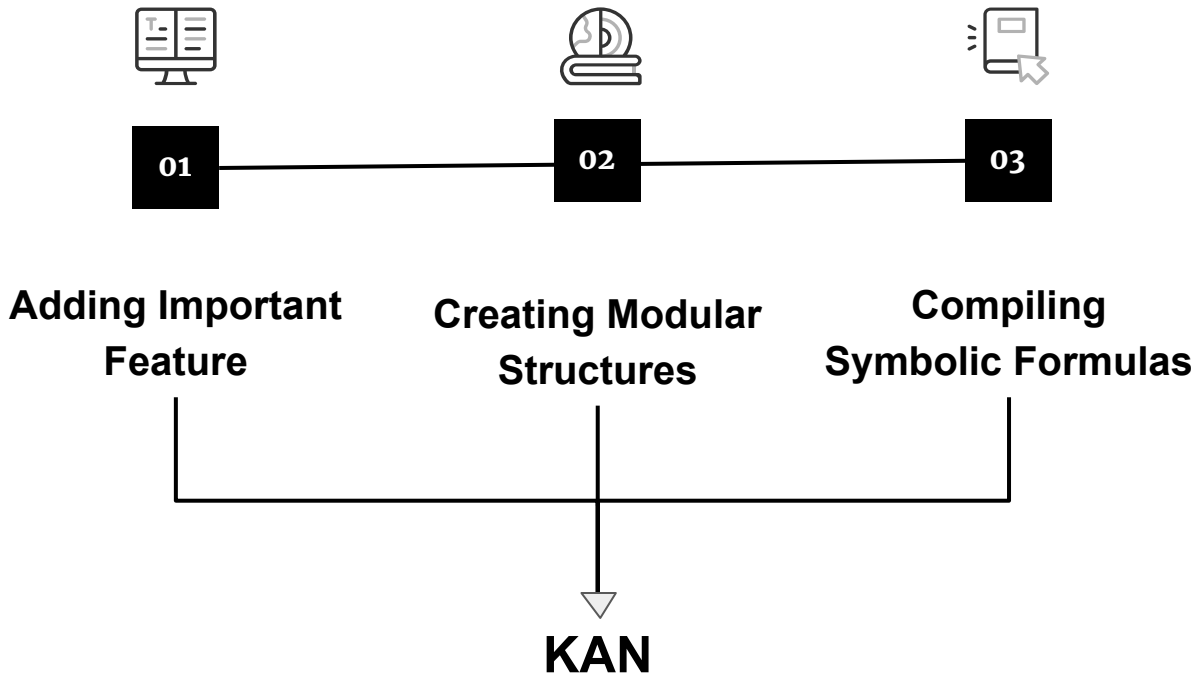
3

Building Scientific Knowledge

Knowledge Embedding

Overview of embedding knowledge

incorporate available inductive biases into KANs



1. Adding important features to KANs

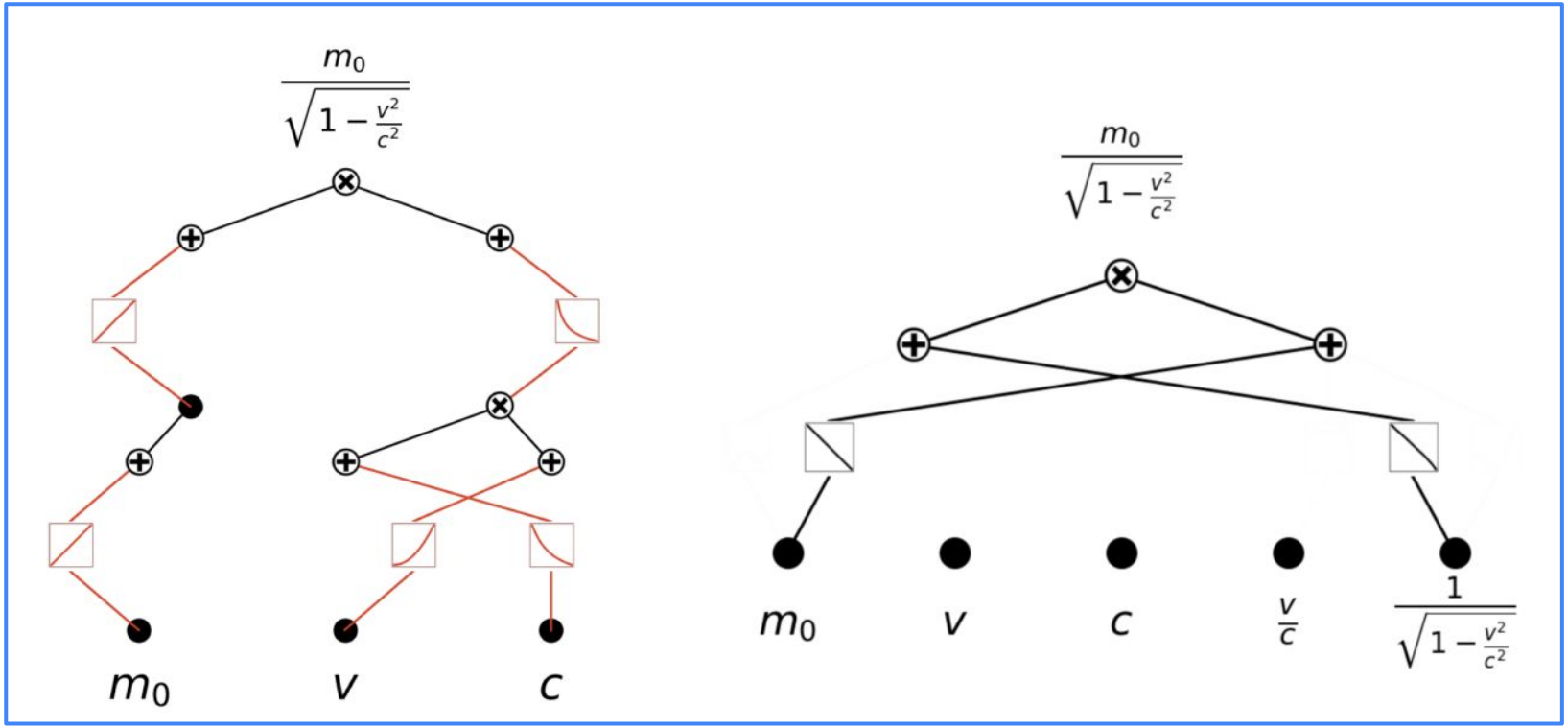
- Addition of auxiliary input variable
- Adds expressive power and interpretability to the network

$$y = f(x_1, x_2, \cdots, x_n)$$

$$a = a(x_1, x_2, \ldots, x_n)$$

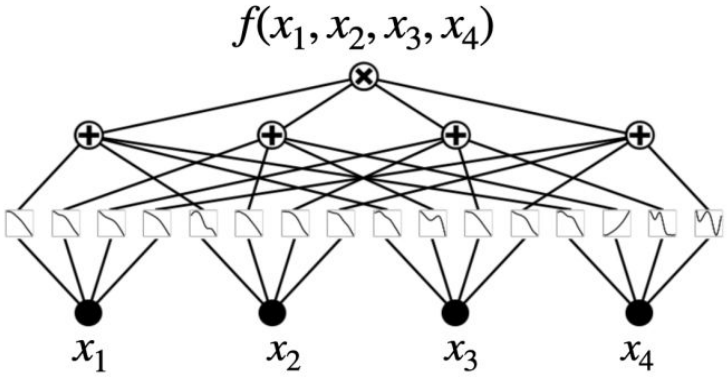
$$y = f(x_1, \cdots, x_n, a)$$

Example


$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$
m_0
v
c
$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$
m_0
v
c
$\frac{v}{c}$
$\frac{1}{\sqrt{1-\frac{v^2}{c^2}}}$

2. Modular structures

- Allows for extracting knowledge from clusters of neurons
- KAN allows for explicit creation of modular structures.
- Two types of modularity:
  1. Separability
  2. Generalized Symmetry




```
start_layer_id, '[nodes_id]->[subnodes_id]->[nodes_id]...'
```

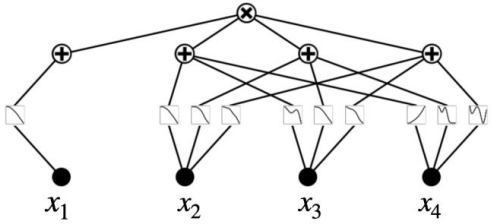
Separability

$f(x_1, x_2, x_3, x_4)$

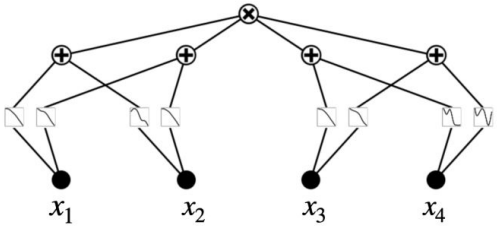
f is separable if it can be expressed as a sum or product of functions of non-overlapping variable groups

```
0,'[0]->[0]'
0,'[0,1]->[0,1]'
0,'[i]->[i]'
```

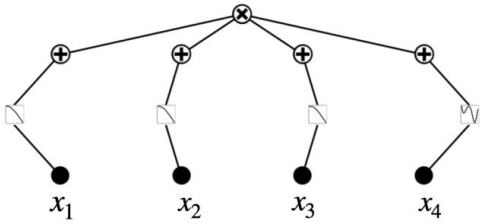
$f_1(x_1) \cdot f_2(x_2, x_3, x_4)$



$f_1(x_1, x_2) \cdot f_2(x_3, x_4)$



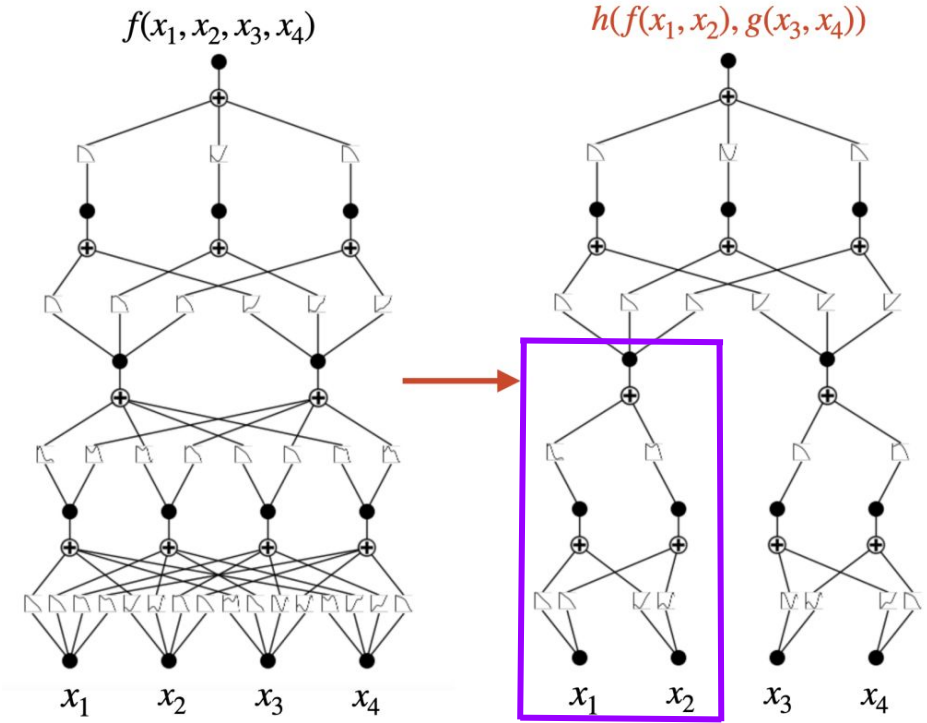
$f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_4)$



Generalized symmetry

$f(x_1, x_2, x_3, \cdots) = g(h(x_1, x_2), x_3, \cdots)$

```
0,‘[0,1]->[0,1]->[0,1]->[0]’
```



3. Compiling Symbolic Formulas

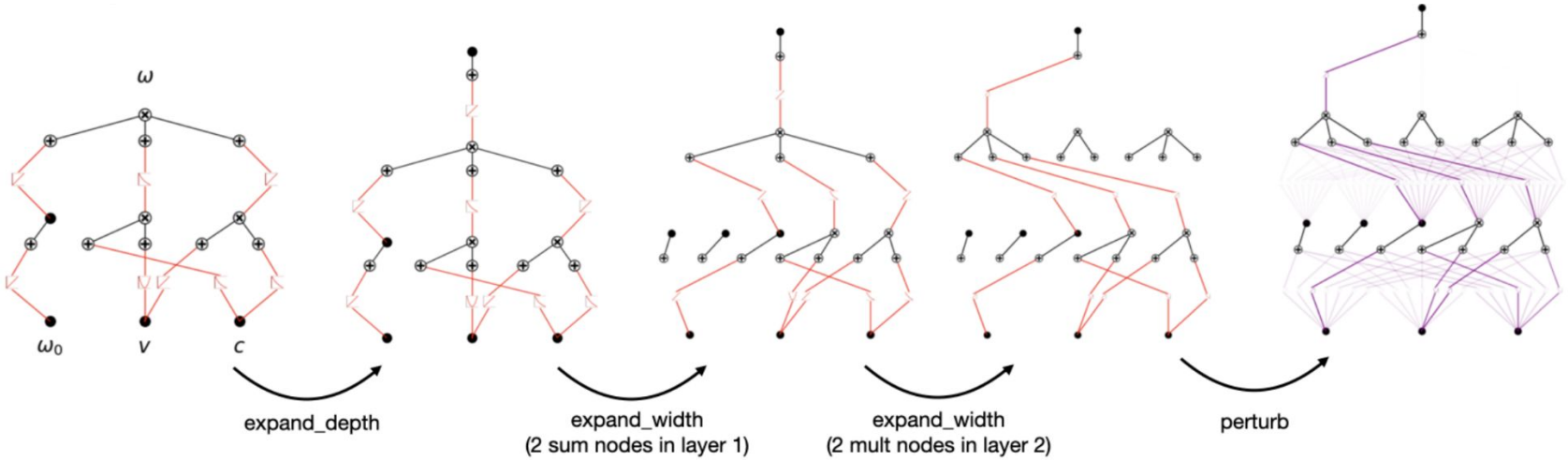
Combine expressivity of neural networks with symbolic explainable formulas

Two steps:

1. Compile formula into a KAN (introducing domain knowledge)
2. Train KAN on data (learn “new physics”)

KANPILER + Learning

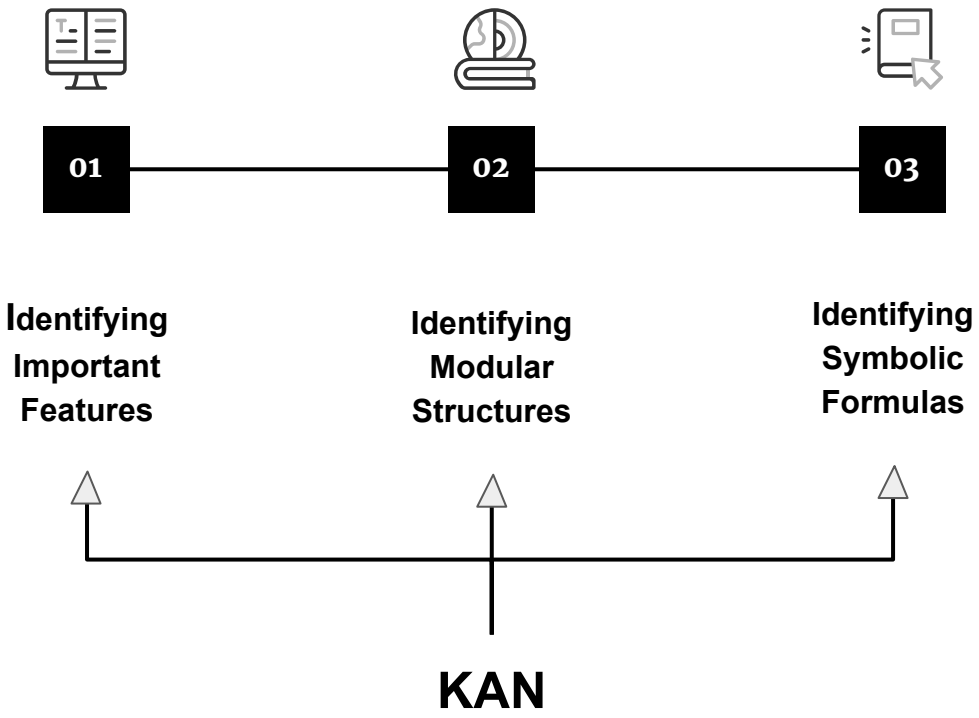
1. Parse Formula → Tree, Nodes = expressions, Edges = operations/functions
2. Transform Tree → KAN Structure Moves leaf nodes to input layer
3. Create Final Graph by combining variables in first layer



4

Extracting Knowledge

Overview of Knowledge Extraction

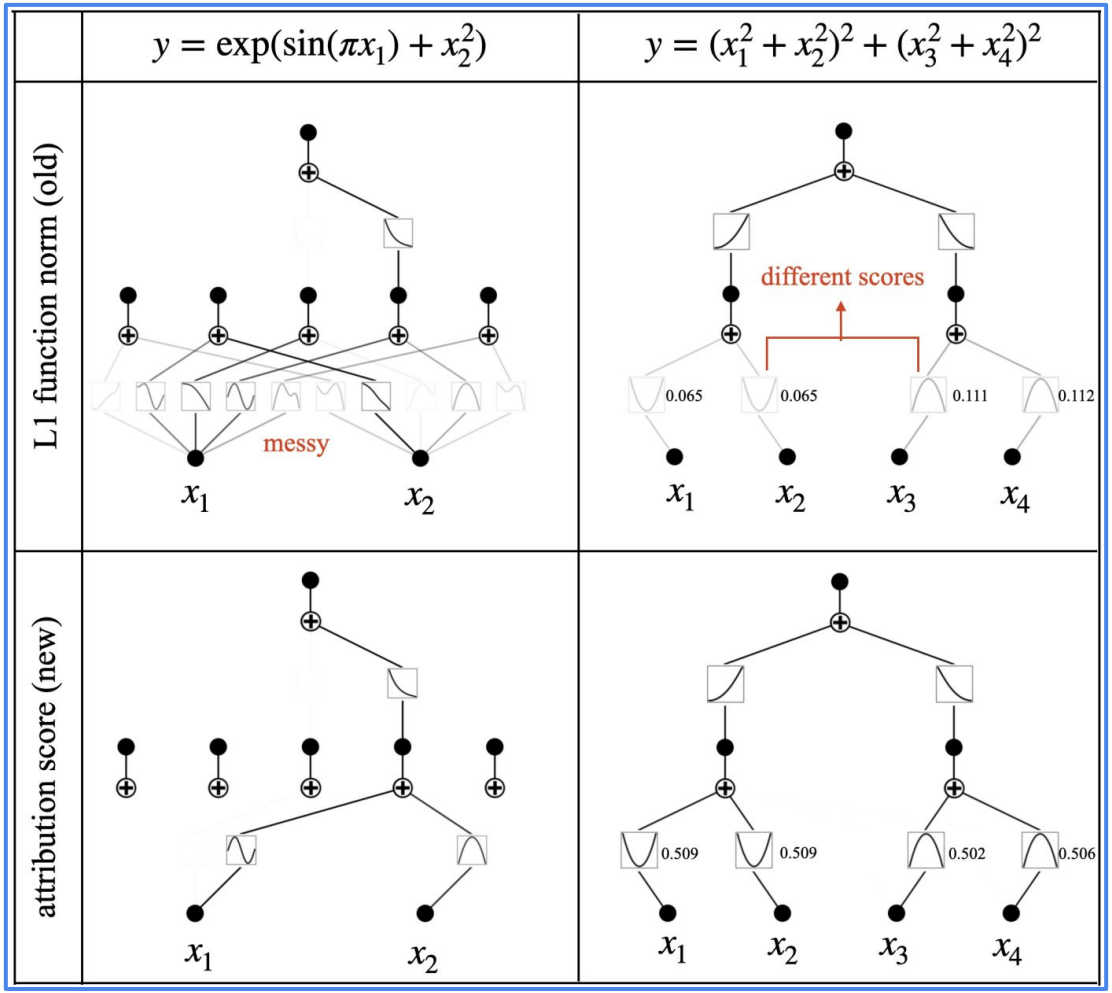


1. Identifying Important Features

- Assign scores to important inputs
- Previously: L1 norm (only considers local information)
  - $E_{l,i,j}$ = standard deviation of the function on each edge
  - $N_{l,i}$ = standard deviation at each node
- Update: Iterative computation of scores
  - node (attribution) score: $A_{l,i}$
  - edge (attribution) score: $B_{l,i,j}$

$$B_{l-1,i,j} = A_{l,j}\frac{E_{l,j}}{N_{l+1,j}}, \quad A_{l-1,i} = \sum_{j=0}^{n_l} B_{l-1,i,j}, \quad l = L, L-1, \cdots, 1.$$

Example


$y = \exp(\sin(\pi x_1) + x_2^2)$
$y = (x_1^2 + x_2^2)^2 + (x_3^2 + x_4^2)^2$
L1 function norm (old)
attribution score (new)
messy
different scores
0.065
0.065
0.111
0.112
0.509
0.509
0.502
0.506
x_1
x_2
x_3
x_4

2. Identifying modular structures

1

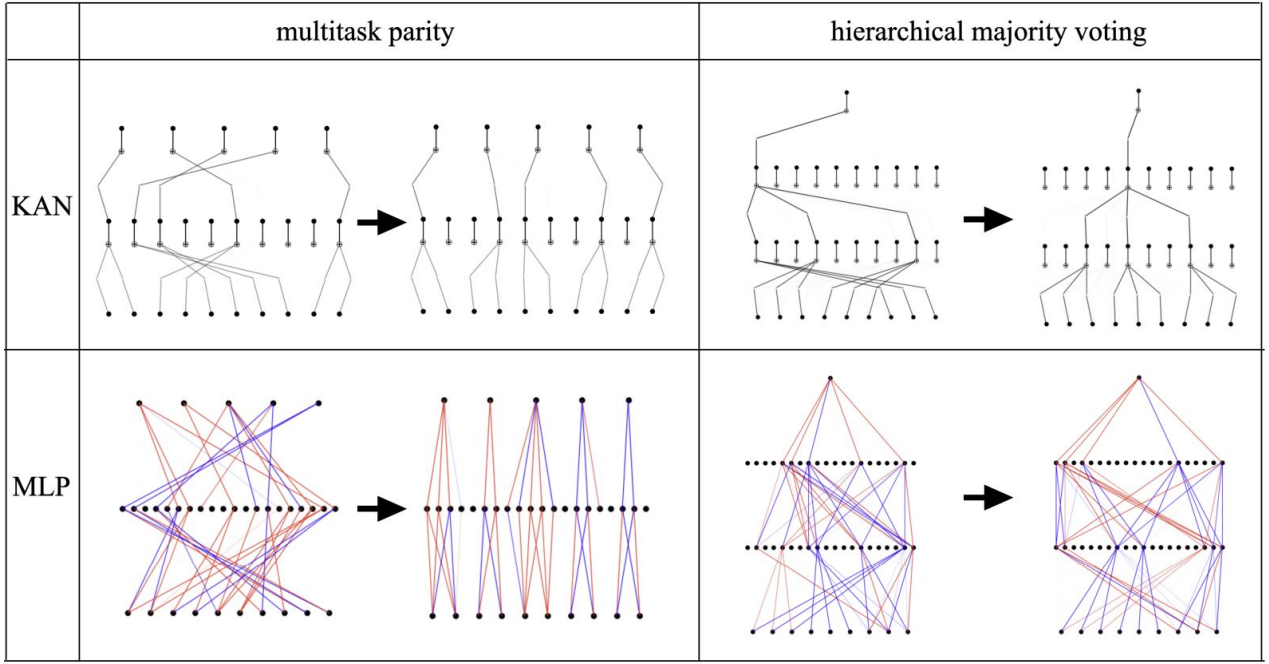
anatomical modularity

2

functional modularity

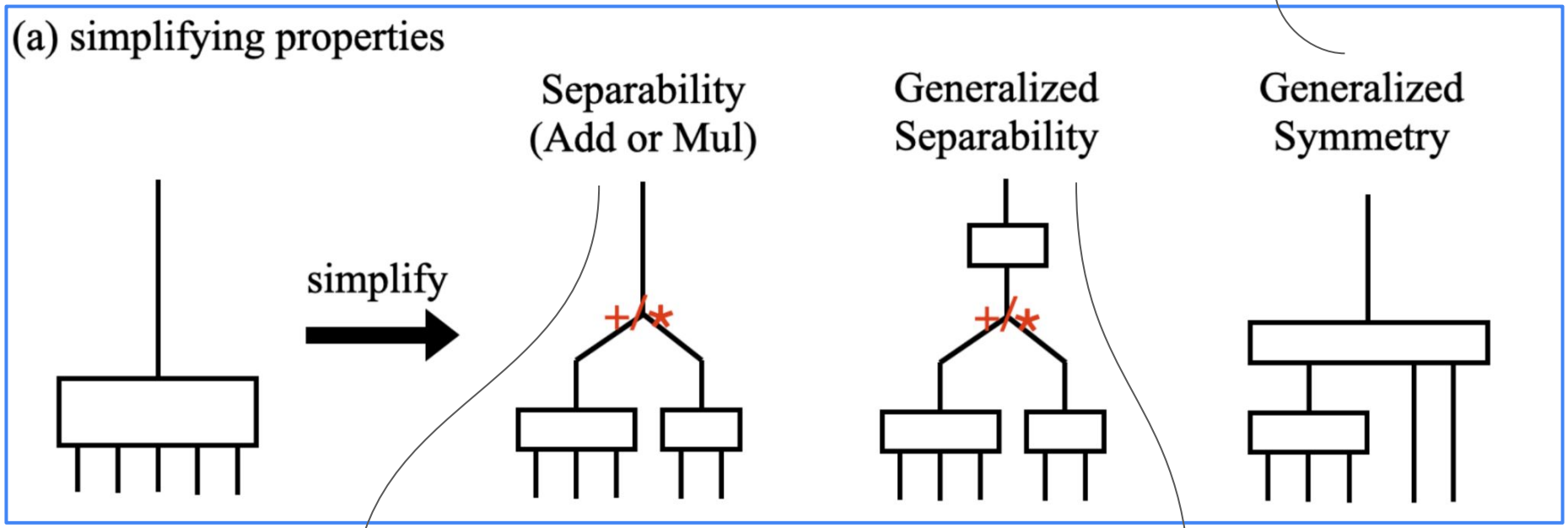
Anatomical Modularity

- Spatial proximity of neurons
- Autoswap: neuron swapping methods (preserves functionality)
- Easy to visually identify modular structures



Functional modularity

$$f(x_1, x_2, \cdots, x_n) = g(h(x_1, \cdots, x_k), x_{k+1}, \cdots, x_n)$$



$$f(x_1, x_2, \cdots x_n) = g(x_1, \ldots, x_k) + h(x_{k+1}, \ldots, x_n)$$

$$f(x_1, x_2, \cdots x_n) = F(g(x_1, \ldots, x_k) + h(x_{k+1}, \ldots, x_n))$$

Functional Modularity: separability

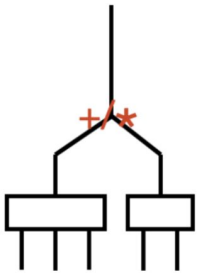
- Additively separable
- Compute the hessian matrix
- If $H_{ij} = 0$ for all $1 \le i \le k$ and $k + 1 \le j \le n$ then additively separable
- For multiplicative separability, convert to additive separability by taking the log

$$f(x_1, x_2, \cdots x_n) = g(x_1, \ldots, x_k) + h(x_{k+1}, \ldots, x_n)$$

$$\mathbf{H} \equiv \nabla^T \nabla f \ (\mathbf{H}_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j})$$

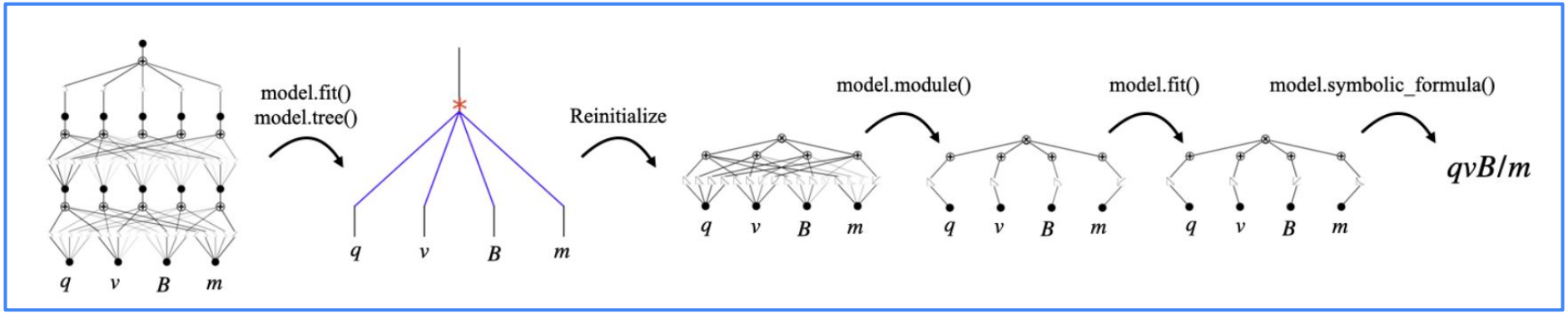
$$\log|f(x_1, x_2, \cdots, x_n)| = \log|g(x_1, \ldots, x_k)| + \log|h(x_{k+1}, \ldots, x_n)|$$

$$\mathbf{H}_{ij} \equiv \frac{\partial^2 \log|f|}{\partial x_i \partial x_j}$$



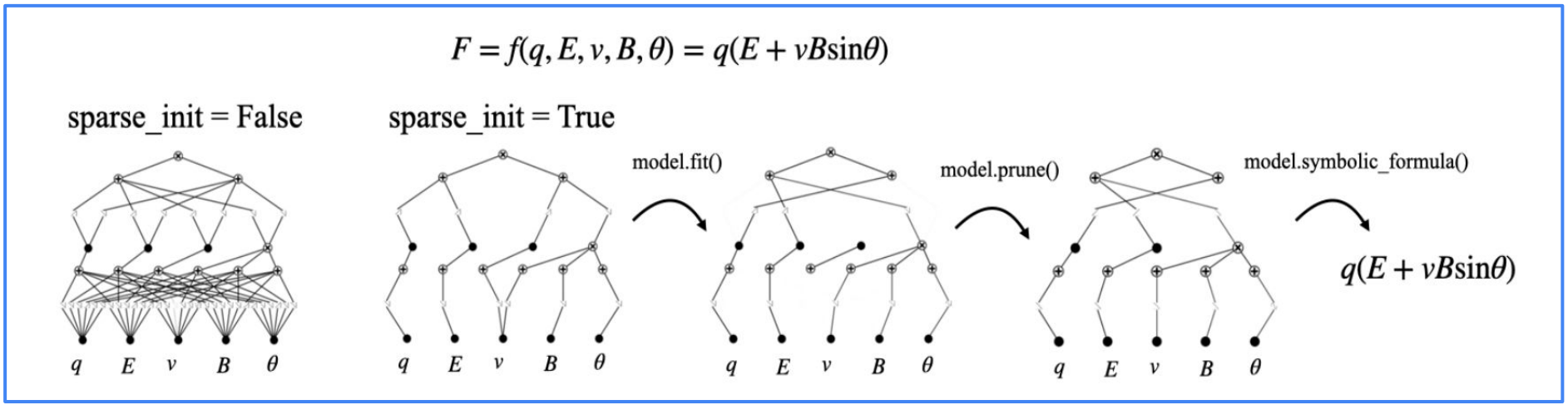
3. Identifying symbolic formulas

Trick A: discover and leverage modular structures



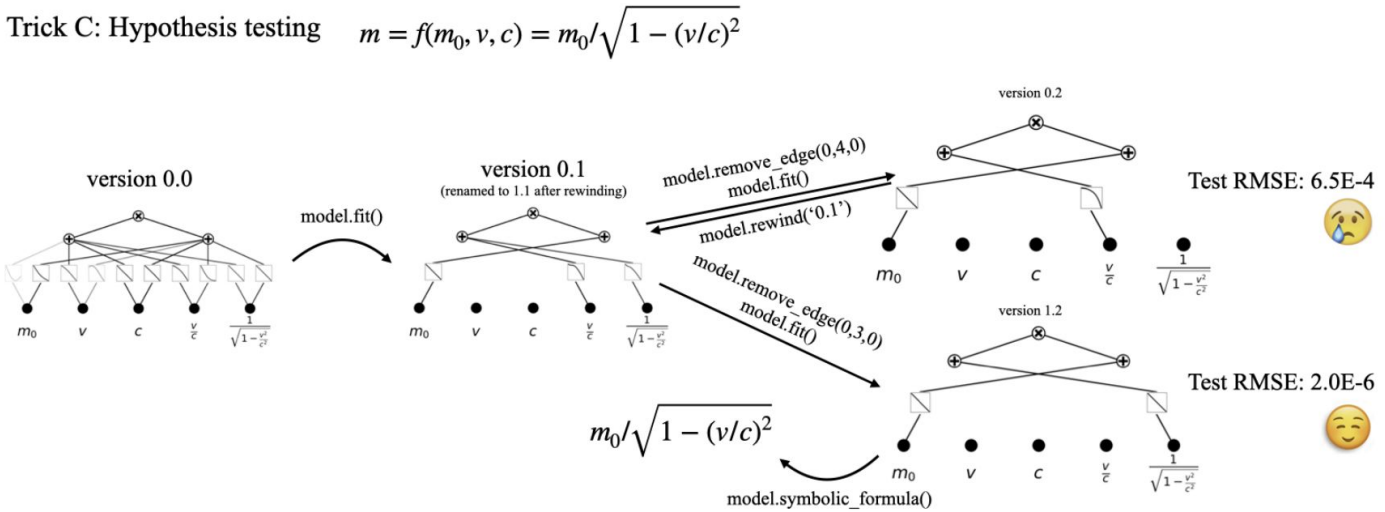
3. Identifying symbolic formulas

Trick B: Sparse initialization — a more natural fit



3. Identifying symbolic formulas

Trick C: Hypothesis testing - easy to try different ideas.



5

Real-World Applications

Application areas

1. Discovering Lagrangians

- Core idea: Learn lagrangian from data
  - Phase space: position and velocity
- Predict accelerations using Euler-Lagrange equations

How KANs Help:

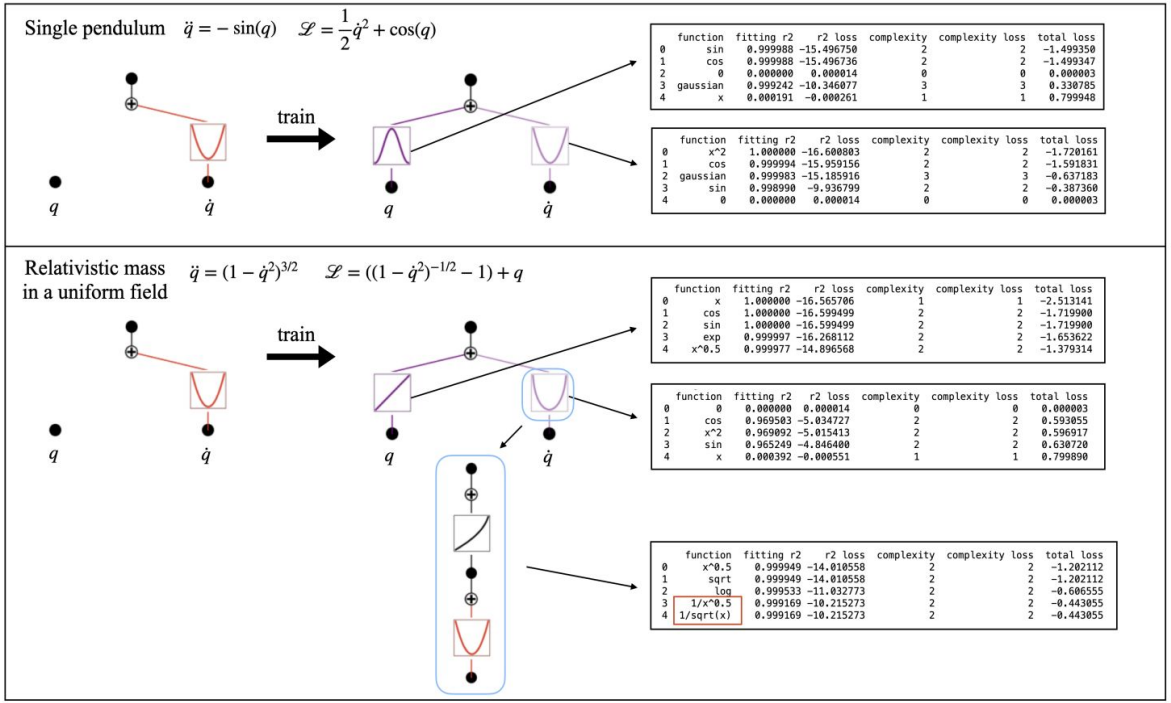
1. Numerical Stability:
   - Pre-encode kinetic energy
2. Interpretability:
   - Symbolic regression extracts explicit formulas from KAN edges

$$\mathrm{L}(\mathrm{q}, \dot{q}) = T - V \, from(q, \dot{q}, \ddot{q})$$

$$\ddot{q} = (\nabla_{\dot{q}} \nabla_{\dot{q}}^{T} L)^{-1} \left[\nabla_{q} L - \nabla_{q} \nabla_{\dot{q}}^{T} \dot{q} \right]$$

KAN solution

- **Key Results**:
  - **Stable Training**: Fewer numerical errors.
  - **Explainable Models**: Learns physics-aligned formulas.



	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	sin	0.999988	-15.496750	2	2	-1.499350
1	cos	0.999988	-15.496736	2	2	-1.499347
2	0	0.000000	0.000014	0	0	0.000003
3	gaussian	0.999242	-10.346077	3	3	0.330785
4	x	0.000191	-0.000261	1	1	0.799948

	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	x^2	1.000000	-16.600803	2	2	-1.720161
1	cos	0.999994	-15.959156	2	2	-1.591831
2	gaussian	0.999983	-15.185916	3	3	-0.637183
3	sin	0.998990	-9.936799	2	2	-0.387360
4	0	0.000000	0.000014	0	0	0.000003

	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	x	1.000000	-16.565706	1	1	-2.513141
1	cos	1.000000	-16.599499	2	2	-1.719900
2	sin	1.000000	-16.599499	2	2	-1.719900
3	exp	0.999997	-16.268112	2	2	-1.653622
4	x^0.5	0.999977	-14.896568	2	2	-1.379314

	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	0	0.000000	0.000014	0	0	0.000003
1	cos	0.969503	-5.034727	2	2	0.593055
2	x^2	0.969092	-5.015413	2	2	0.596917
3	sin	0.965249	-4.846400	2	2	0.630720
4	x	0.000392	-0.000551	1	1	0.799890

	function	fitting r2	r2 loss	complexity	complexity loss	total loss
0	x^0.5	0.999949	-14.010558	2	2	-1.202112
1	sqrt	0.999949	-14.010558	2	2	-1.202112
2	log	0.999533	-11.032773	2	2	-0.606555
3	1/x^0.5	0.999169	-10.215273	2	2	-0.443055
4	1/sqrt(x)	0.999169	-10.215273	2	2	-0.443055

2. Constitutive laws

- **Constitutive Laws**:
  - Define material behavior under forces or deformation (e.g., Hooke's Law).
  - Linear for small deformations (Hooke's law)
  - Nonlinear for larger deformations (e.g., Neo-Hookean).
- **Why Use KANs?**
  - predict the P (stress) tensor from the F (deformation) tensor
  - Incorporates **prior knowledge** (e.g., linear laws).
  - Extracts **interpretable symbolic formulas** via symbolic regression.

$$P_{12} = \mu(F_{12} + F_{21})$$

$$P_{12} = \mu(F_{11}F_{21} + F_{12}F_{22} + F_{13}F_{23})$$

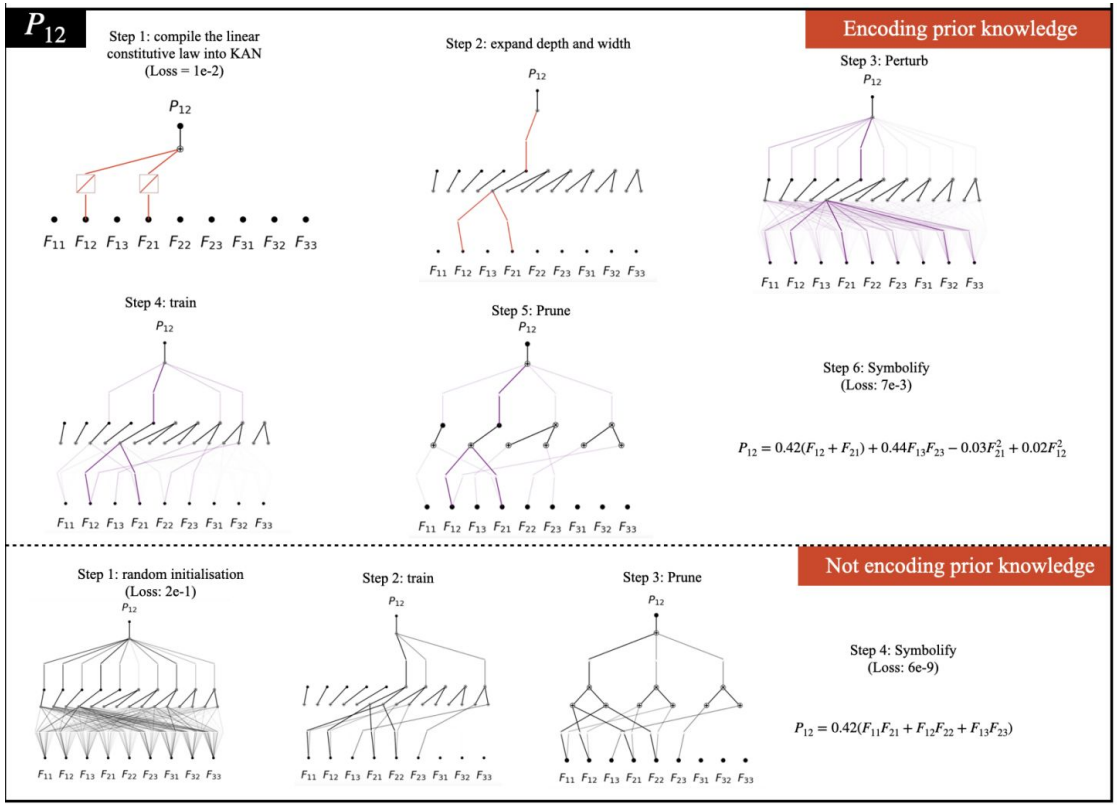
Results:

With Prior Knowledge:

- The initial KAN is based on the linear constitutive law.
- Helps guide training but may converge to poor local minima

- **Without Prior Knowledge**:
  - Start with a randomly initialized KAN.
  - Requires more data to train but avoids being biased

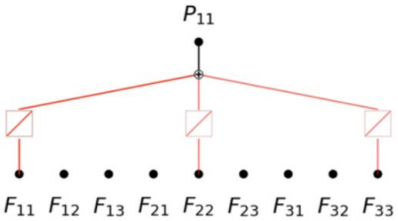
$$P_{12} = \mu(F_{12} + F_{21})$$



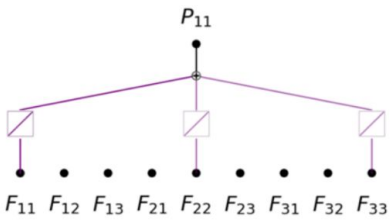
$$P_{12} = \mu(F_{11}F_{21} + F_{12}F_{22} + F_{13}F_{23})$$

P_{11}

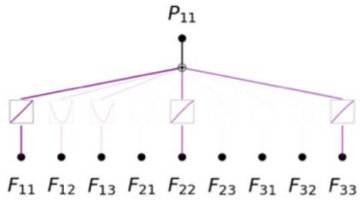
Step 1: compile the linear constitutive law into KAN (Loss = 1e-2)



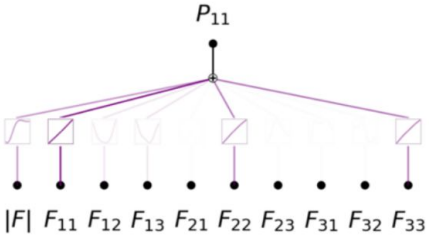
Step 2: Perturb



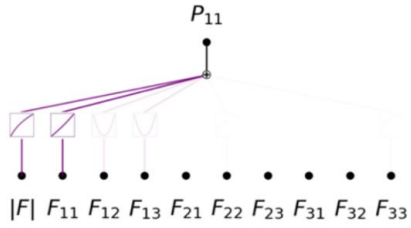
Step 3: train (Loss = 6e-3)



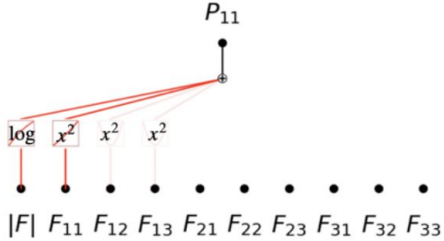
Step 4: Add the determinant as auxiliary variable



Step 5: train (Loss = 2e-4)



Step 6: symbolify (Loss = 3e-11)

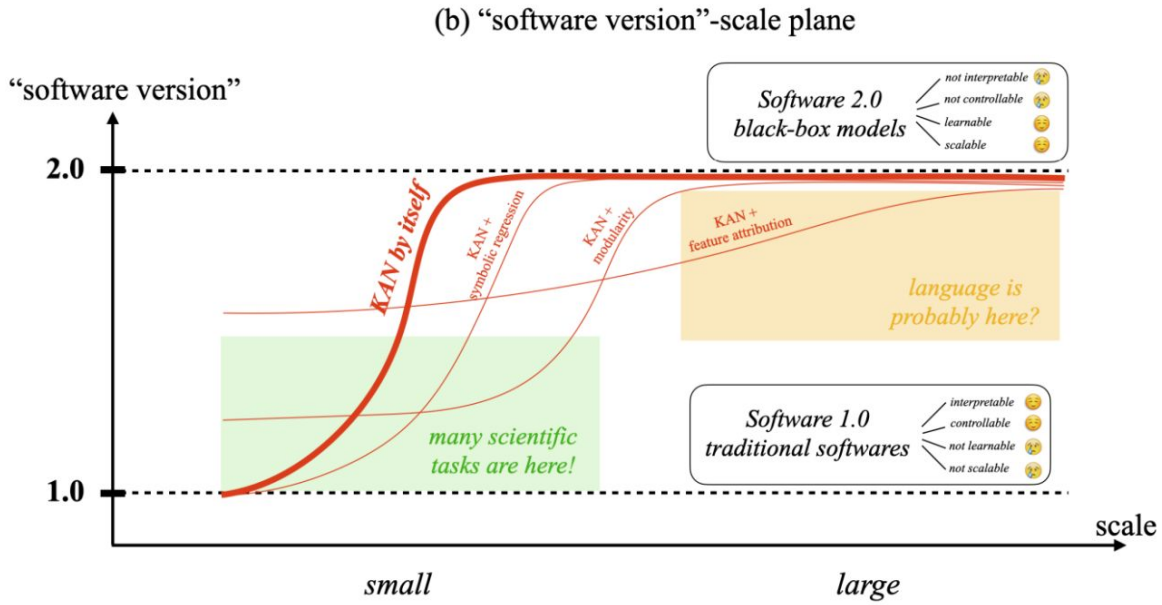


$$P_{11} = 0.42(F_{11}^2 + F_{12}^2 + F_{13}^2 - 1) + 0.28\log(|F|)$$

6

Future

Software


(b) “software version”-scale plane
“software version”
2.0
1.0
scale
small
large
Software 2.0
black-box models
not interpretable
not controllable
learnable
scalable
KAN by itself
KAN + symbolic regression
KAN + modularity
KAN + feature attribution
language is probably here?
many scientific tasks are here!
Software 1.0
traditional softwares
interpretable
controllable
not learnable
not scalable

Outlook

- **Scalability Challenge**
  - Limited interpretability at large scales
  - Need better methods for managing complexity
- **Key Research Areas**
  - Advanced interpretability methods
  - Scaling to larger problems
  - Expansion beyond physics

KAN VS MLP

MLP:

- Lower interpribility
- Well established and proven

KAN:

- 10x Slower training time
- Higher interpretability
- Continual learning
- More parameter efficient

References

1. Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," *arXiv preprint*, arXiv:2404.19756, Jun. 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2404.19756

2. Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark, "KAN 2.0: Kolmogorov-Arnold Networks meet science," *arXiv preprint*, arXiv:2408.10205, Aug. 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2408.10205

$$\phi(x) = w_b b(x) + w_s \text{spline}(x).$$

$$b(x) = \text{silu}(x) = x/(1 + e^{-x})$$

$$\text{spline}(x) = \sum_i c_i B_i(x)$$

1. Discovering conserved quantities

- Conserved quantities as differential equation solving
- $z \in R^d$ governed by the equation $dz/dt = f(z)$
- $H(z)$ is a conserved quantity if $f(z) \cdot \nabla H(z) = 0$ for all Z

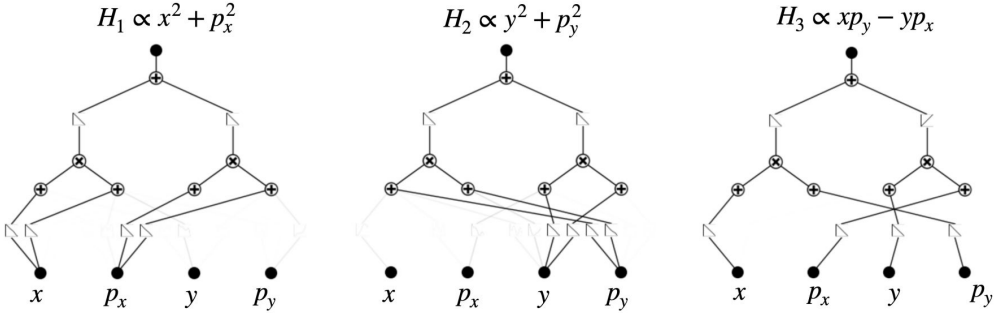
Harmonic Oscillator

- System Description:
  - Phase space: $z = (x, p)$
    - x: position
    - p: momentum
- Evolution:
  - $d(x,p)/dt = (p, -x)$

Harmonic Oscillator

Energy Conservation:

- Energy function: $H = \frac{1}{2}(x^2 + p^2)$
- Phase space: $z = (x, p)$
- Conservation proof:
  - Force vector: $f(z) = (p, -x)$
  - Gradient of H: $\nabla H(z) = (x, p)$
  - Dot product: $f(z) \cdot \nabla H(z) = p \cdot x + (-x) \cdot p = 0$
  - Zero result proves H is conserved



KAN:

- Parametrize H into network
- Train with loss function

$$\ell = \sum_{i=1}^{N} \left| \mathbf{f}(\mathbf{z}^{(i)}) \cdot \widehat{\nabla} H(\mathbf{z}^{(i)}) \right|^2$$