Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Vector Basis Model

Jia Fujie

hellipisqqs@qq.com

individual author

Research Article

Keywords:

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11792246/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Vector Basis Model

Jia Fujie

November 18, 2026

Contents

1 Introduction **2**

2 Related Work **2**

2.1 Capsule Networks and Soft Clustering 3

3 Method **3**

3.1 Construction of the High-Dimensional Directional Basis 4

3.2 Cosine-Gated Layer 4

3.3 Magnitude–Direction Decoupling 5

3.4 Initialization 6

4 Experiments **6**

4.1 Tasks and Protocol 6

4.2 Baselines 6

4.3 Per-Task Results 7

4.4 Significance Analysis 7

4.5 Overall Comparison and Training Dynamics 8

4.6 Mechanism Analysis 11

5 Backpropagation Gradient Analysis **16**

5.1 Backpropagation through One Layer 16

5.2 Gradient Composition and Geometric Interpretation 17

5.3 Time and Space Complexity 17

6 Conclusion **18**

A Experimental Configuration **18**

A.1 Data and Splits 18

A.2 Optimization and Training 18

A.3 Model Structure and Initialization 18
A.4 Baseline Configurations 19
A.5 Significance Tests and Ablation 19
A.6 Mechanism-Experiment Configuration 19

Abstract

Fully connected networks represented by the MLP project inputs onto one-dimensional nonlinear units and stack multiple layers to fold complex decision boundaries, and constitute one of the most widely used deep-learning paradigms; their pointwise one-dimensional nonlinearity has proven effective on a wide range of tasks, yet on some tasks with complex high-dimensional manifold structure it may face challenges in expressive capacity and generalization stability, and the information propagated between nodes can be susceptible to noise. In this paper we propose the **Vector Basis Model** (VBN), which constructs nonlinearity explicitly in a high-dimensional basis space: nonlinearity depends only on the **directional** differences between vectors, while the norm (magnitude) of the input is left to the weights to encode feature importance. At each node the input is projected into a direction vector in $\mathbb{R}^K$, enabling the node to represent, cluster and discriminate patterns in a high-dimensional manifold in a more regularized way. Decoupling magnitude from pattern allows feature importance and the manner of participation to be distinguished separately: information is computed after projection onto the manifold, and the nonlinearity between nodes is determined by the degree of similarity in the projection space.

1 Introduction

Neural networks need to rotate and fold the feature space in a data-driven, layer-wise manner, yet data cannot always be fitted by a model through regularized projection; this paper provides a viable route to this end.

Conventional models such as MLPs and SIRENs project each nonlinear operation onto a single one-dimensional processing unit; such combinations of pointwise nonlinear basis functions have proven effective on a wide range of tasks, though their mode of expression may leave room for improvement on certain complex tasks. The essential difference between the proposed model and conventional ones is that **nonlinearity is determined by similarity**. Each node projects the input into a direction in a high-dimensional basis space, tends to retain components close to its reference direction, and determines the retention level by computing the angle with the expected feature; the norm of the linear pathway is not amplified by the nonlinearity, thereby decoupling the amount of information (importance) from the manner of participation (pattern). This can be viewed as a directional form of spatial projection and information-conditioning mechanism: the range of nonlinear variation is controllable, the model learns node-specific attributes spontaneously, and it maintains regularization efficiency and stable generalization while propagating complex features.

2 Related Work

Backpropagation provides a general gradient-learning framework for multilayer networks [1]; the MLP accumulates nonlinear capacity layer by layer and learns intrinsic features implicitly, with a simple and highly general structure, and can be seen as an implicit "searcher" [1]. LoRA projects weight updates into a low-rank space and changes the learning dynamics by altering the *way weight matrices are combined*, but it is an improvement targeted at the

weight matrix and cannot replace the expressive power of an activation function [2]. The KAN (Kolmogorov–Arnold Network) places learnable one-dimensional spline functions on network edges, explicitly parameterizing "how the function bends", and performs remarkably on tasks such as symbolic regression and scientific computing [3]; its nonlinearity is processed edge-wise within one dimension, and its expressive capacity and stability depend on the allocation of spline degrees of freedom, requiring a trade-off between the number of degrees of freedom and noise. SIREN propagates information implicitly via sinusoidal activations with periodic initialization and excels at signal fitting and implicit neural representations; in our experiments the parameter-matched SIREN with $\omega_0 = 5$ is likewise a competitive baseline [4]. Unlike the conventional activation functions above, which nonlinearize the input pointwise, the gate proposed in this paper nonlinearizes the input in a high-dimensional space according to directional similarity, supporting the processing of richer input-attribute information.

2.1 Capsule Networks and Soft Clustering

Capsule networks (CapsNet) [11] replace scalar neurons with vector–matrix "capsules" and establish direction-sensitive aggregation between capsules via dynamic routing: transformed outputs of lower capsules are scored by dot-product similarity against candidate directions of higher capsules, routing weights are updated iteratively by similarity, and information with aligned directions is softly clustered into the same higher capsule, with the vector norm encoding instantiation parameters of that feature. This is mechanically related to the gate of this paper—both softly cluster information by cosine/dot-product directional similarity rather than pointwise scalar discrimination. The two differ, however, in their standpoint: CapsNet emphasizes constructing **discrete** high-level feature entities (e.g., pose and location instantiation parameters) directly from the routed vectors, its information manifold being implicit and formed by routing; VBN/VBNDyn instead realize soft clustering by projection and weighting, explicitly constraining each node's information **within a nonlinear manifold defined by similarity**, with all computations (linear projection, gating, channel mixing) subsumed into this similarity-projection framework. Relative to CapsNet, the proposed method describes the shape and variation of the manifold more explicitly (Sections 4.6 and 5 give geometric visualization and a gradient interpretation, respectively), expresses features of different kinds more generally on complex tasks, and is better suited as a bottom-layer information-processing module embedded in larger models.

3 Method

All notation below is for a single sample (no batch dimension). The symbols are indexed as follows:

Symbol	Meaning	Shape
x	input vector	$\mathbb{R}^d$
V_0	initial random unit direction basis (L_2-normalized by row)	$\mathbb{R}^{d\times K}$
A	node matrix; row i is the K-dimensional vector of node i	$\mathbb{R}^{n\times K}$
W	inter-layer linear projection weight	$\mathbb{R}^{n\times m}$
e_u (e_3)	learnable bias	$\mathbb{R}^{m\times K}$
e_{ref} / e_2	reference direction (VBN learnable constant / VBNDyn dynamic bias)	$\mathbb{R}^{m\times K}$
W_2, W_3	two maps of the dynamic reference pathway	$\mathbb{R}^{nK\times K}$, $\mathbb{R}^{K\times mK}$
P	channel-mixing matrix (initialized to I_K)	$\mathbb{R}^{K\times K}$
c, g	per-node cosine similarity and gate	$\mathbb{R}^m$
b^{weight}, b^{bias}	per-dimension scale and bias	$\mathbb{R}^K$, $\mathbb{R}^{m\times K}$
W^{head}, b^{head}	output-head projection	$\mathbb{R}^{n\times o}$, $\mathbb{R}^{o\times K}$

where d is the input dimension, n and m are the numbers of input/output nodes in a layer, K is the dimension of the directional basis at each node, o is the output dimension, and L is the number of layers.

3.1 Construction of the High-Dimensional Directional Basis

Let the input vector be $x \in \mathbb{R}^d$ and the random unit-direction matrix $V_0 \in \mathbb{R}^{d\times K}$ (L_2-normalized by row), whose i-th row $v_i \in \mathbb{R}^K$ is the direction basis of the i-th input dimension. The initial node matrix is

$$A^{(0)}_{i,:} = x_i\, v_i, \qquad A^{(0)} \in \mathbb{R}^{d\times K}. \tag{3.1}$$

That is, each input dimension corresponds to one node: magnitude information is preserved in the norm of the node vector, while directional information is encoded by its orientation in the K-dimensional basis space, which is fixed by default.

3.2 Cosine-Gated Layer

A cosine-gated layer takes the node matrix $A \in \mathbb{R}^{n\times K}$ as input and outputs $B' \in \mathbb{R}^{m\times K}$. With $W \in \mathbb{R}^{n\times m}$, first compute a linear projection for each output node:

$$V_{j,:} = \sum_i A_{i,:}\, W_{i,j} \in \mathbb{R}^K. \tag{3.2}$$

Introduce a learnable bias $e_u \in \mathbb{R}^{m\times K}$ (code variable `e_u`, denoted e_3 in the notation of this paper), giving $u_j = V_{j,:} + (e_u)_{j,:}$. The reference direction $r_j \in \mathbb{R}^K$ has two constructions:

- **VBN:** the reference direction is a learnable constant $e_{\text{ref}} \in \mathbb{R}^{m\times K}$, i.e. $r_j = (e_{\text{ref}})_{j,:}$. - **VBNDyn:** the reference direction is generated dynamically from the input via two linear maps: a shared reference bottleneck $h = \text{vec}(A)\, W_2 \in \mathbb{R}^K$, projected by W_3 and reshaped as

$$d = \text{reshape}(h\, W_3) \in \mathbb{R}^{m\times K}, \qquad r_j = d_{j,:} + (e_2)_{j,:}, \tag{3.3}$$

where $W_2 \in \mathbb{R}^{nK\times K}$ and $W_3 \in \mathbb{R}^{K\times mK}$, and e_2 is a learnable bias. In the main experiments e_{ref} and e_2 use **exactly the same initialization**: a Gaussian vector is sampled independently for each node and then L_2-normalized to obtain a random unit vector of norm one.

Compute the cosine similarity and construct the gate per node:

$$c_j = \cos(u_j, r_j) = \frac{\langle u_j, r_j\rangle}{\|u_j\|\,\|r_j\|}, \qquad g_j = \text{clip}(1 + c_j, 0, 1). \tag{3.4}$$

The gate determines retention by the sign of the angle: after clipping, $g_j = 1$ for $c_j \geq 0$ and the signal is fully retained; for $c_j \in (-1, 0)$, $g_j = 1 + c_j$ decays continuously as the directions diverge; and as $c_j \to -1$ the signal is progressively closed. The orthogonal point $c_j = 0$ is the critical threshold of full retention. Because g_j depends only on direction and not on magnitude, the linear norm is not amplified by the nonlinearity.

Viewed through the geometric intuition of the spatial manifold, the explicitly constructed manifold of this work can be described as **a coupled nonlinear space**: the decay and enhancement of the nonlinearity depend directly on the direction of the input — the directional change (rotation) of the input vector after projection into the space directly determines the change of

each node's cosine similarity, so the strength of the nonlinearity is directly tied to the input direction; meanwhile, the nonlinear nodes in this space are not independent of one another, but are mutually correlated through the shared reference directions and similarities. Consequently, nodes with similar directions are naturally clustered onto well-posed modes: the variation of the input vector lands on this basis in the form of coordinated co-variation across nodes, so the update of a single layer is constrained and projected onto a well-posed (regular) spatial manifold; and this manifold is not fixed — its reference directions are learned via e_{ref} or the dynamic reference pathway, making the manifold itself learnable and shapeable. On this manifold, the positive-similarity side constitutes a **non-decaying direction interval** ($g_j \equiv 1$ for $c_j \geq 0$): gradients flowing back through it are not attenuated, which better stabilizes the gradient flow; on this interval the output remains linearly related to the input, making linear regimes easier to express. The "non-decaying–linear" positive interval and the "coupled-decay–nonlinear" negative interval together characterize the geometry of this coupled nonlinear space: the former offers a clear path for gradient stability and linear expression, while the latter supplies the mechanism for node clustering and coordinated update.

A channel-mixing matrix $P \in \mathbb{R}^{K\times K}$ then performs the linear combination, followed by a per-dimension scale and bias:

$$V'_{j,:} = g_j\, V_{j,:}\, P, \qquad B'_{j,:} = b^{\text{bias}}_{j,:} + b^{\text{weight}} \odot V'_{j,:}. \tag{3.5}$$

Here $b^{\text{weight}} \in \mathbb{R}^K$ is initialized to all ones, $b^{\text{bias}} \in \mathbb{R}^{m\times K}$ to zero, and P by default to the identity I_K.

Stack L gated layers as $A^{(\ell+1)} = \text{CosGateLayer}_\ell(A^{(\ell)})$. The output head linearly projects the final node matrix and averages over the K dimension:

$$D_{o,:} = \sum_i A^{(L)}_{i,:}\, W^{\text{head}}_{i,o} + b^{\text{head}}_{o,:} \in \mathbb{R}^K, \qquad \hat{y}_o = \frac{1}{K}\sum_{k=1}^{K} D_{o,k}. \tag{3.6}$$

Averaging over K retains sign information while preserving magnitude.

3.3 Magnitude–Direction Decoupling

Here the weight matrix serves a dual, explicit and implicit role: W decides "which inputs participate and in what direction", while the gate g decides "how much to retain along that direction". Since g is determined solely by the angular difference between directions, the degree of nonlinearity is unchanged when the input pattern is unchanged; nonlinear adjustment is therefore confined to regular, continuous, trend-like manifold variation, while the angular constraint requires the adjustments among pieces of information to cooperate. A node's input direction is obtained by a magnitude-weighted vote over its inputs, and W merely rotates and scales magnitudes and changes the importance of each direction at different target nodes; directional change is constrained within node-wise scaling, projection, weight voting and similarity, so the updated nonlinearity is more regular.

The composition of the manifold differs by model: for VBNDyn the manifold is jointly determined by the K-dimensional projected nodes of the reference pathway and the K-dimensional projection of the input, whereas for VBN it is determined only by the input's K-dimensional projection. The multiplicative cosine gate makes information whose direction is closer to a reference feature easier to retain, so the model can naturally distinguish which nodes are interrelated with one another and construct a spatial network in the manifold sense. When the task is complex, deep stacking drives this explicit manifold progressively towards the reference

manifold of the target function: related patterns tend to cluster nearby through multiple layers, spatial variation remains continuous wherever the gate is continuous, and the model can still express asymmetric forms and rapidly capture abrupt changes in direction, thereby expressing richer information attributes with limited parametric complexity.

3.4 Initialization

- $W \sim \mathcal{N}(0, 1/n_{\text{in}})$, corresponding to the dimensional setting $\operatorname{Var} W \approx 1/(\mathbb{E}[g^2] \cdot n_{\text{in}})$ (see the variance correction below); - e_u (e_3): $\mu = 0, \sigma = 0$ (initialized to zero); - e_{ref} (VBN) and e_2 (VBNDyn): **initialized in exactly the same way**, by sampling $\mathcal{N}(0, I_K)$ independently per node followed by L_2 normalization, giving random unit vectors of norm one; - $W_2 \sim \mathcal{N}(0, 1/(nK))$, $W_3 \sim \mathcal{N}(0, 1/K)$, $P = I_K$, $b^{\text{weight}} = \mathbf{1}$, $b^{\text{bias}} = 0$.

The default `randn` mode gives the linear projection unit variance. However, because the gate acts multiplicatively ($V' = g \odot V\,P$), if the input and reference directions are approximately independent then $\mathbb{E}[g^2] < 1$, and forward variance decays layer by layer. We therefore introduce a variance-corrected initialization (`gate_var`), scaling the standard deviation of W to

$$\sigma_W = \frac{1}{\sqrt{n_{\text{in}}\,\mathbb{E}[g^2]}} = \frac{1}{\sqrt{n_{\text{in}}\,(eg2 + mean2)}} \tag{3.7}$$

so that $\mathbb{E}[g^2] \cdot n_{\text{in}}\sigma_W^2 = 1$, keeping the forward variance unchanged after one gated layer. Here $g = \text{clip}(1 + \cos, 0, 1)$; with u, r isotropic K-dimensional unit directions and $t = \cos(u, r)$ symmetrically distributed, $\mu_K = \mathbb{E}|t| = \Gamma(K/2)/(\sqrt{\pi}\,\Gamma((K+1)/2))$, $\mathbb{E}g = 1 - \mu_K/2$, $\mathbb{E}[g^2] = 1 - \mu_K + 1/(2K)$, with $eg2 = \operatorname{Var}(g)$, $mean2 = (\mathbb{E}g)^2$ and $eg2 + mean2 = \mathbb{E}[g^2]$. For $K = 4$, $\mu_4 \approx 0.4244$ and $\mathbb{E}[g^2] \approx 0.7006$, so σ_W is enlarged by about 1.19×; for $K = 3$, $\mathbb{E}[g^2] = 2/3$, enlarged by about 1.22×. Both the main experiments and the depth ablation adopt this initialization.

4 Experiments

4.1 Tasks and Protocol

The experiments cover two families of tasks: (1) nine real UCI regression datasets: abalone, airfoil_self_noise, auto-mpg, Concrete_Data, energy_efficiency, kin8nm, winequality-red, winequality-white and yacht_hydrodynamics; (2) three synthetic function tasks: `sin20x` (high-frequency $y = \sin 20x$), `complex_trend` and `complex_trend2` (a multi-segment, continuous, non-differentiable complex trend). Data are split 80/20 with input standardization, and all models share the same split; training uses Adam with cosine learning-rate decay for 300 epochs and a batch size of 512 (the remainder is merged into the final batch); four random seeds (0–3) are used for each task and each model. The metric is the coefficient of determination R^2 (in percent). The complete hyperparameters are listed in Appendix A.

4.2 Baselines

In addition to VBN and VBNDyn, the baselines include: a **parameter-matched MLP** (hidden width searched per task over four layers so that the parameter count exactly matches VBN or VBNDyn, with the same number of layers, learning rate 10^{-2} and 300 epochs); **KAN** (learnable B-spline one-dimensional functions, matched exactly to the parameter count of VBNDyn); **SIREN** (a sinusoidal-activation network, matched exactly to the parameter count of VBN);

and common baselines CatBoost, LightGBM, XGBoost, RandomForest, ExtraTrees, HistGB and SVR. The key settings of these baselines are given in Appendix A.

4.3 Per-Task Results

Table 1 and 2 report the per-task R^2 (mean ś std) of all 13 models on the nine UCI tasks and the three synthetic tasks, respectively.

Table 1. Per-task R^2 (%) on UCI regression tasks (mean ś std over 4 seeds)

Task	VBN	VBNDyn	MLP(VBN)	MLP(VBNDyn)	KAN(VBNDyn)	SIREN(VBN)	CatBoost	LightGBM	XGBoost	RF	ET	HistGB	SVR
abalone	26.7ś6.5	26.6ś3.1	31.3ś12.9	26.7ś4.9	47.5ś4.1	32.0ś6.4	56.2ś1.9	52.0ś1.9	53.8ś1.7	56.1ś1.8	55.5ś2.0	52.1ś2.3	56.6ś2.0
airfoil_self_noise	95.8ś0.7	96.5ś0.5	95.2ś1.5	96.0ś1.1	91.5ś5.2	93.4ś2.0	94.5ś0.6	95.0ś0.5	95.4ś0.5	91.9ś0.4	92.8ś0.7	94.6ś0.6	83.8ś1.7
auto-mpg	85.8ś5.5	82.7ś4.8	85.5ś4.4	82.4ś3.4	90.3ś2.4	88.3ś3.3	91.4ś1.1	89.3ś1.9	90.7ś2.2	90.4ś1.8	90.3ś1.7	89.5ś1.9	91.1ś2.1
Concrete_Data	92.0ś1.4	89.8ś2.1	91.3ś1.4	92.0ś1.5	89.6ś2.4	88.0ś1.5	93.7ś0.6	93.1ś0.5	93.3ś0.2	89.9ś0.8	90.9ś1.3	93.3ś0.7	88.5ś2.0
energy_efficiency	99.8ś0.1	99.6ś0.2	99.6ś0.1	99.7ś0.1	99.7ś0.0	99.5ś0.2	99.9ś0.0	99.8ś0.1	99.9ś0.1	99.7ś0.1	99.7ś0.1	99.8ś0.0	98.7ś0.2
kin8nm	91.5ś0.4	89.9ś0.4	90.0ś0.3	89.5ś0.8	94.1ś0.2	90.1ś0.5	84.5ś0.8	82.9ś0.9	80.5ś0.8	72.1ś1.2	73.7ś1.3	82.8ś0.9	92.3ś0.2
winequality-red	13.3ś4.9	21.5ś6.3	11.9ś7.3	14.7ś4.1	4.4ś15.4	7.0ś10.6	47.6ś3.8	47.4ś3.4	48.4ś4.5	50.0ś5.2	50.7ś5.1	45.8ś4.0	35.8ś6.6
winequality-white	20.9ś4.7	25.8ś3.6	19.9ś4.4	28.0ś4.5	18.1ś8.2	-0.4ś5.7	47.4ś0.3	51.8ś1.4	53.6ś1.3	53.2ś0.7	55.1ś0.6	51.7ś0.9	40.2ś1.0
yacht_hydrodynamics	98.5ś0.6	92.9ś4.3	87.3ś3.7	81.4ś13.6	99.4ś0.1	16.4ś28.6	98.3ś1.7	99.6ś0.2	99.5ś0.1	99.4ś0.3	99.5ś0.3	94.9ś0.1	86.1ś7.0

Table 2. Per-task R^2 (%) on synthetic function tasks (mean ś std over 4 seeds)

Task	VBN	VBNDyn	MLP(VBN)	MLP(VBNDyn)	KAN(VBNDyn)	SIREN(VBN)	CatBoost	LightGBM	XGBoost	RF	ET	HistGB	SVR
complex_trend	96.5ś2.1	99.6ś0.3	90.0ś2.1	89.9ś1.9	81.3ś1.7	99.9ś0.1	99.8ś0.1	99.8ś0.1	99.8ś0.1	100.0ś0.0	99.8ś0.0	99.7ś0.0	82.5ś2.1
complex_trend2	98.3ś0.2	99.3ś0.1	97.1ś0.6	97.9ś0.6	92.5ś1.2	99.9ś0.0	96.1ś0.4	97.6ś0.1	54.6ś5.1	97.3ś0.1	97.8ś0.2	97.2ś0.3	93.2ś0.5
sin20x	99.4ś0.5	100.0ś0.0	80.8ś13.2	71.4ś19.9	83.2ś12.5	100.0ś0.0	99.5ś0.0	99.6ś0.1	99.5ś0.1	99.9ś0.0	99.4ś0.1	99.1ś0.3	0.7ś4.8

On the three synthetic functions VBN/VBNDyn obtain 99.4%/100.0% on `sin20x` and 98.3%/99.3% on `complex_trend2`, exceeding the parameter-matched MLP (80.8%/71.4% and 97.1%/97.9%) and KAN (83.2% and 92.5%) on this group of tasks. On the nine UCI tasks, each model has its own strengths: KAN(VBNDyn) leads on abalone (47.5%) and kin8nm (94.1%), and SIREN(VBN) is on par with the best on smooth tasks such as energy_efficiency; VBN/VBNDyn are close to the best results on airfoil, Concrete, energy, kin8nm and yacht, but fall below the tree models on winequality-red/white and abalone (e.g., 26.7% for VBN versus 52.0% for LightGBM on abalone).

4.4 Significance Analysis

Table 3. Per-task paired significance of VBN relative to each competitor

Task	MLP(VBN)	MLP(VBNDyn)	KAN(VBNDyn)	SIREN(VBN)	CatBoost	LightGBM	XGBoost	SVR
abalone	-4.6	-0.0	-20.8*	-5.3	-29.4*	-25.3*	-27.1*	-29.9*
airfoil_self_noise	+0.6	-0.2	+4.3	+2.4*	+1.3*	+0.8*	+0.4	+12.0*
auto-mpg	+0.2	+3.4	-4.6	-2.6	-5.6	-3.5	-4.9	-5.4
Concrete_Data	+0.7	-0.0	+2.4*	+4.0*	-1.7	-1.1	-1.3	+3.5
energy_efficiency	+0.2*	+0.1	+0.1	+0.3	-0.1*	-0.1	-0.1*	+1.0*
kin8nm	+1.5*	+2.0*	-2.5*	+1.5*	+7.1*	+8.6*	+11.1*	-0.8*
winequality-red	+1.4	-1.5	+8.8	+6.3	-34.4*	-34.2*	-35.1*	-22.5*
winequality-white	+1.0	-7.2	+2.7	+21.3*	-26.5*	-31.0*	-32.8*	-19.3*
yacht_hydrodynamics	+11.2*	+17.1	-1.0	+82.1*	+0.2	-1.2*	-1.0	+12.3*
complex_trend	+6.4*	+6.5*	+15.1*	-3.4	-3.3	-3.3	-3.3*	+14.0*
complex_trend2	+1.2*	+0.4	+5.8*	-1.7*	+2.2*	+0.7*	+43.6*	+5.1*
sin20x	+18.6	+28.0	+16.2	-0.6	-0.1	-0.2	-0.1	+98.7*

Each cell is the per-task R^2 difference (this model minus the competitor), from a paired t-test over four seeds with Holm correction [10]; * denotes corrected $p < 0.05$, with positive values favoring the proposed model. The competitor columns are the parameter-matched MLP, SIREN(VBN, $\omega_0 = 5$), KAN(VBNDyn), CatBoost, LightGBM, XGBoost and SVR. From the starred entries: relative to the parameter-matched MLP, VBN is positive on most tasks with margins mostly within two percentage points, VBNDyn differs in sign across tasks, and the

Table 4. Per-task paired significance of VBNDyn relative to each competitor

Task	MLP(VBN)	MLP(VBNDyn)	KAN(VBNDyn)	SIREN(VBN)	CatBoost	LightGBM	XGBoost	SVR
abalone	-4.7	-0.1	-20.9*	-5.4	-29.5*	-25.4*	-27.2*	-30.0*
airfoil_self_noise	+1.4	+0.5	+5.0	+3.2*	+2.1*	+1.5*	+1.1*	+12.7*
auto-mpg	-2.8	+0.4	-7.6*	-5.6*	-8.6*	-6.6*	-7.9*	-8.4*
Concrete_Data	-1.5	-2.2	+0.2	+1.8	-3.9*	-3.3	-3.5*	+1.3
energy_efficiency	+0.0	-0.0	-0.0	+0.2	-0.2	-0.2	-0.2	+0.9*
kin8nm	-0.2	+0.3	-4.2*	-0.2	+5.4*	+6.9*	+9.4*	-2.4*
winequality-red	+9.6	+6.8	+17.1	+14.5*	-26.1*	-25.9*	-26.9*	-14.3*
winequality-white	+5.9	-2.2	+7.7	+26.2*	-21.6*	-26.0*	-27.8*	-14.4*
yacht_hydrodynamics	+5.6*	+11.5	-6.5	+76.5*	-5.4*	-6.7*	-6.6	+6.8
complex_trend	+9.6*	+9.7*	+18.2*	-0.3	-0.2	-0.2	-0.2	+17.1*
complex_trend2	+2.2*	+1.4*	+6.8*	-0.6*	+3.2*	+1.7*	+44.7*	+6.1*
sin20x	+19.2	+28.6	+16.8	-0.0*	+0.5*	+0.4*	+0.4*	+99.3*

three are overall comparable; relative to SIREN(VBN), with $\omega_0 = 5$ it becomes a competitive matched baseline, and VBN/VBNDyn are close to it on smooth and synthetic tasks (differences mostly within a few percentage points) with favorable margins on some tabular tasks; relative to KAN(VBNDyn), both models are positive on the synthetic tasks, while KAN shows clear advantages on UCI tasks such as abalone and kin8nm, with wins on both sides; relative to tree models [5][6][7][8][9], both are mostly significantly negative on winequality and abalone, and positive or non-significant on several synthetic and smooth tasks.

4.5 Overall Comparison and Training Dynamics

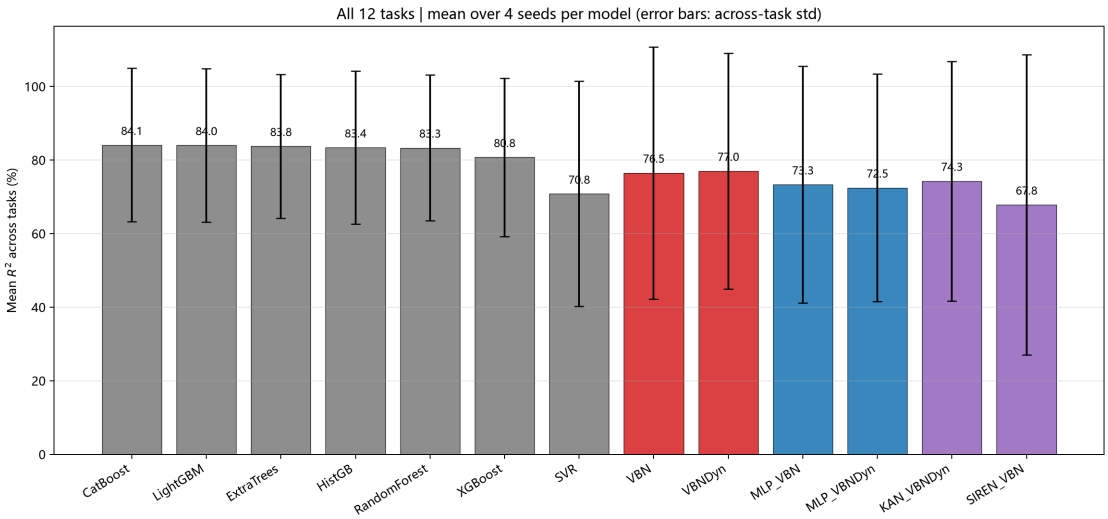


Figure 1. Mean R^2 of all models across all 12 tasks (4 seeds; error bars are cross-task standard deviations)

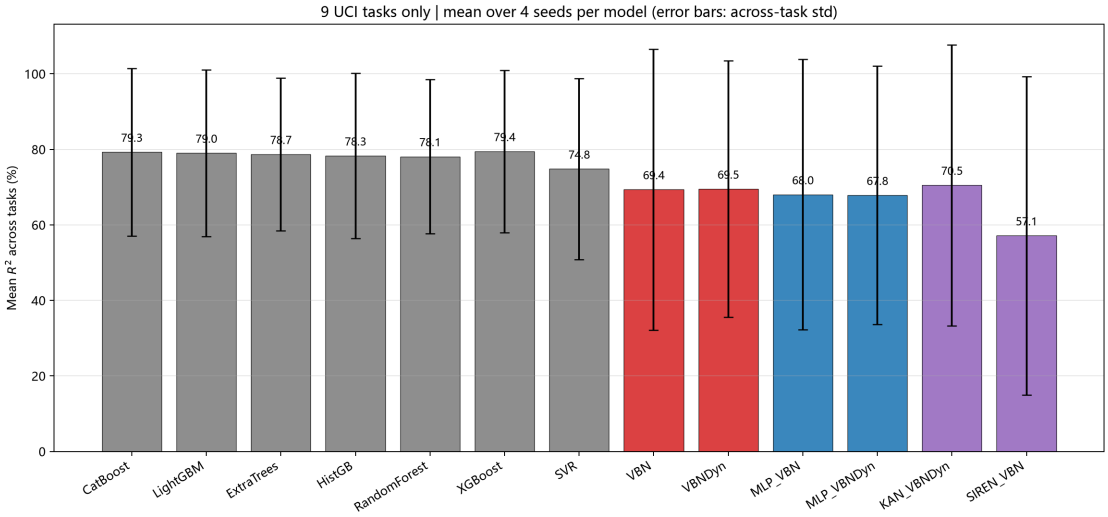


Figure 2. Only the 9 UCI tasks; strong tree models lead overall

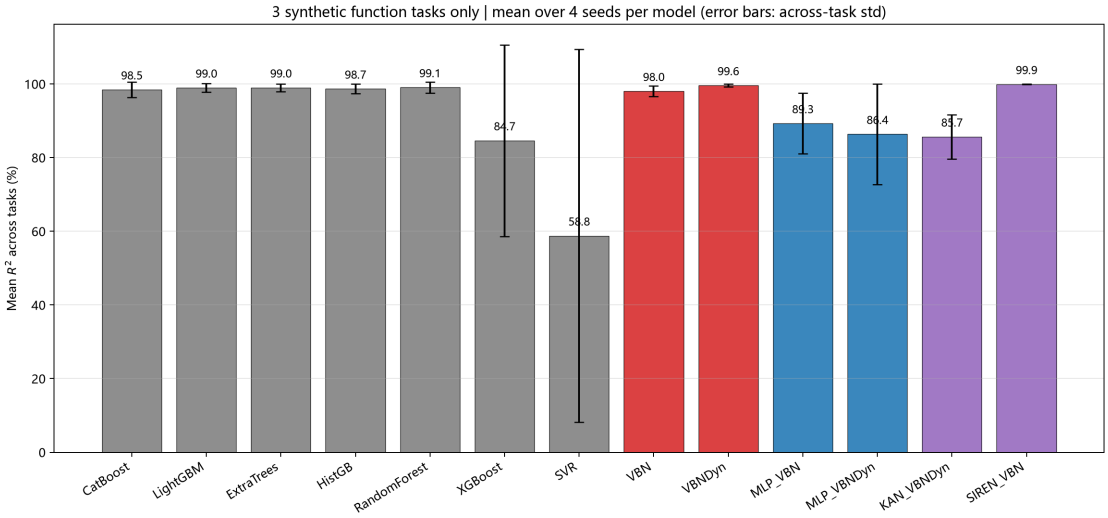


Figure 3. Only the 3 synthetic function tasks; the VBN family is comparable to tree models

Figure 4 reports, task by task, the mean R^2 and the 4-seed standard deviation for VBN, VBN-Dyn and the parameter-matched MLP: on smooth tasks (`sin20x`, `complex_trend`, `energy_efficiency`) all three have small standard deviations, while on tabular tasks such as `winequality` and `abalone` the standard deviations are markedly larger, with comparable magnitudes across the three models. It should be emphasized that, because the conclusions for real UCI tasks and synthetic tasks point in different directions, the overall mean masks this difference; the main text therefore relies on the per-task tables as the primary evidence.

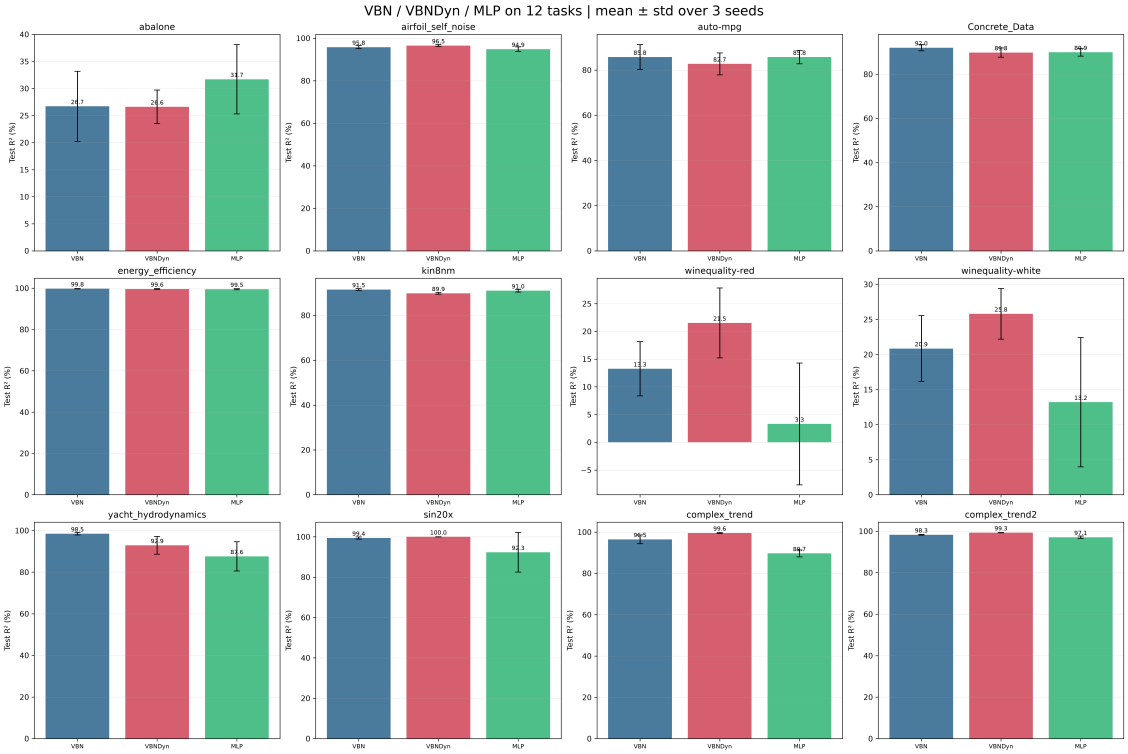


Figure 4. Per-task R^2 of VBN, VBNDyn and the parameter-matched MLP over 12 tasks (mean ś std over 4 seeds)

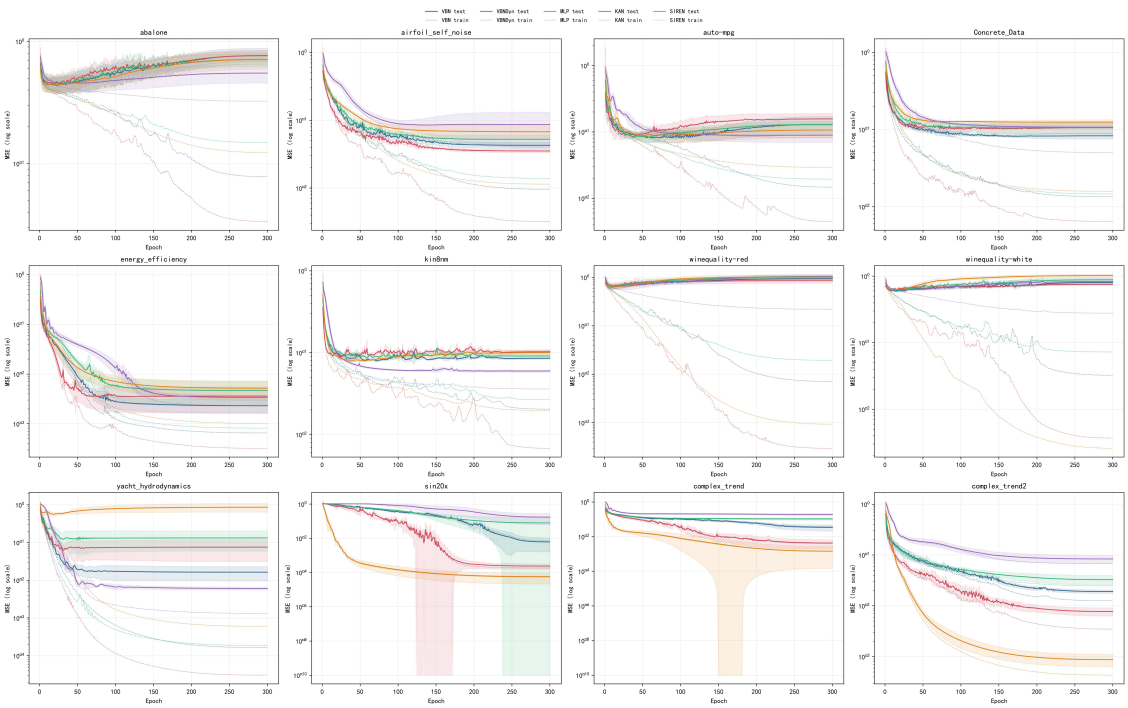


Figure 5. Training/test loss curves for all tasks

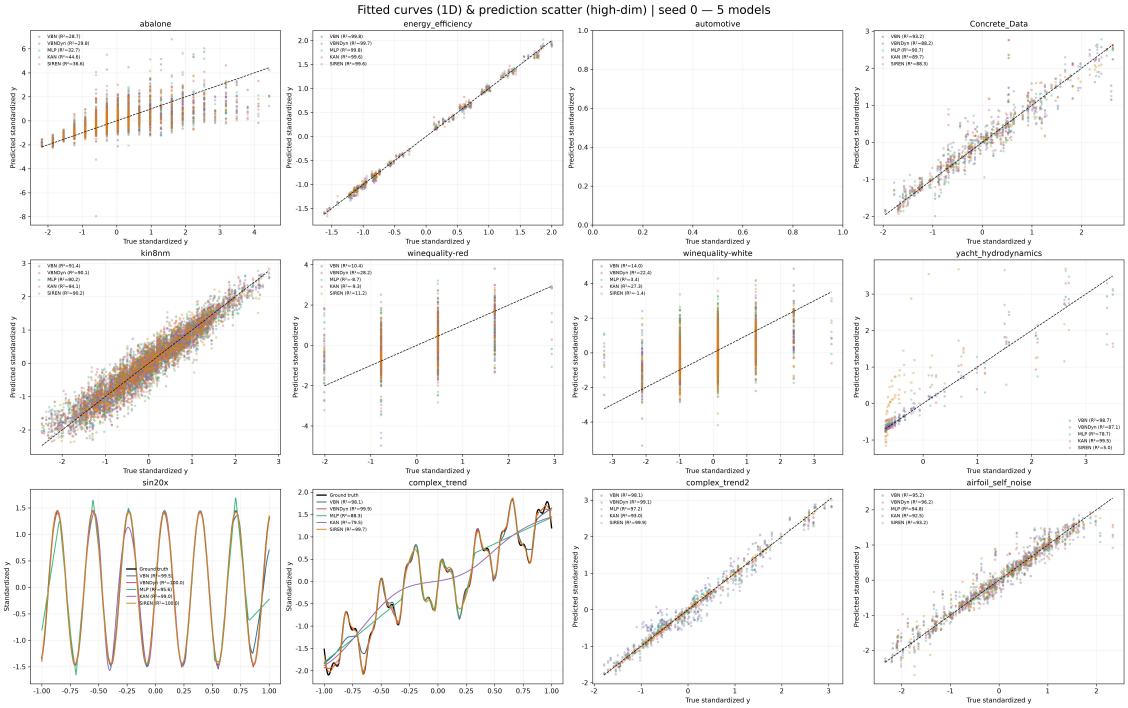


Figure 6. Prediction–ground-truth fits for all tasks

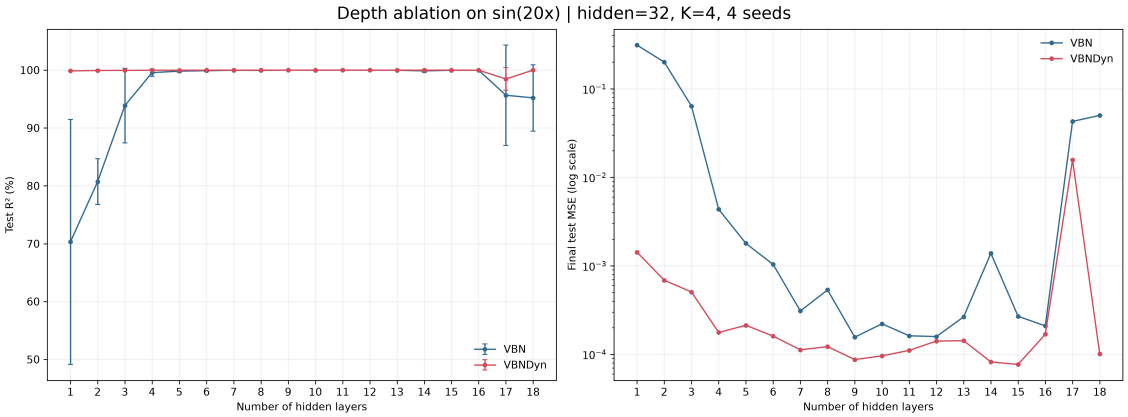


Figure 7. Depth ablation on sin20x over layers 1–18

In the depth ablation (`sin20x`, layers 1–18), VBN attains its highest mean at 9 layers and VBNDyn at 15 layers, both around 99.99%; beyond these depths the performance saturates and does not improve with further layers.

4.6 Mechanism Analysis

The mechanism experiments in this section are an independent visualization analysis using a small-scale configuration different from the main experiments: $K = 3$, 8 nodes per layer, a two-layer network, 1000 epochs and 3 random seeds, on the analytic tasks sinx, linear and quadratic whose ground truth is known. The main configuration ($K = 4$, 32 nodes, 4 layers) is not used because the mechanism figures must display the geometric behavior of the directional basis node by node and channel by channel, and too many nodes or channels are hard to distinguish in a 2D plot; the 1000 epochs ensure that the small network converges to a stable structure.

The mechanism analysis is a qualitative observation, for which 3 random seeds already suffice to show stability, without the multi-seed statistical testing required for performance comparison.

Figures 8–10 report the pairwise cosine-similarity distributions of nodes for the linear, quadratic and sinx targets, respectively, under the random-unit e_2 initialization. In each figure the top row is e_2 and the bottom row is e_3 (i.e. e_u), while the three columns correspond to within-layer 1, within-layer 2 and cross-layer pairs; blue denotes before training and red after training, and each distribution pools 3 seeds (28 pairs per seed within a layer, 64 pairs per seed across layers). Initialized as random unit vectors and updated as learnable biases during training, the pairwise cosines of e_2 remain approximately uniform before and after training (mean near 0, standard deviation about 0.55–0.61), showing that the reference directions are nearly isotropic in distribution. e_3 starts at zero and its initial cosine is undefined (noted in the figures), so only the post-training distribution is reported; it exhibits target- and layer-dependent structural differences—for example, the standard deviation for sinx within layer 2 reaches 0.687, above the isotropic reference $1/\sqrt{3} \approx 0.577$—indicating that the direction distribution of the learnable bias varies with the target and the layer.



Figure 8. Node pairwise cosine-similarity distributions of e_2/e_3 on the linear target (top: e_2, bottom: e_3; left: within layer 1, middle: within layer 2, right: cross-layer; blue: before training, red: after training)



Figure 9. Node pairwise cosine-similarity distributions of e_2/e_3 on the quadratic target (panel layout as in Figure 8)

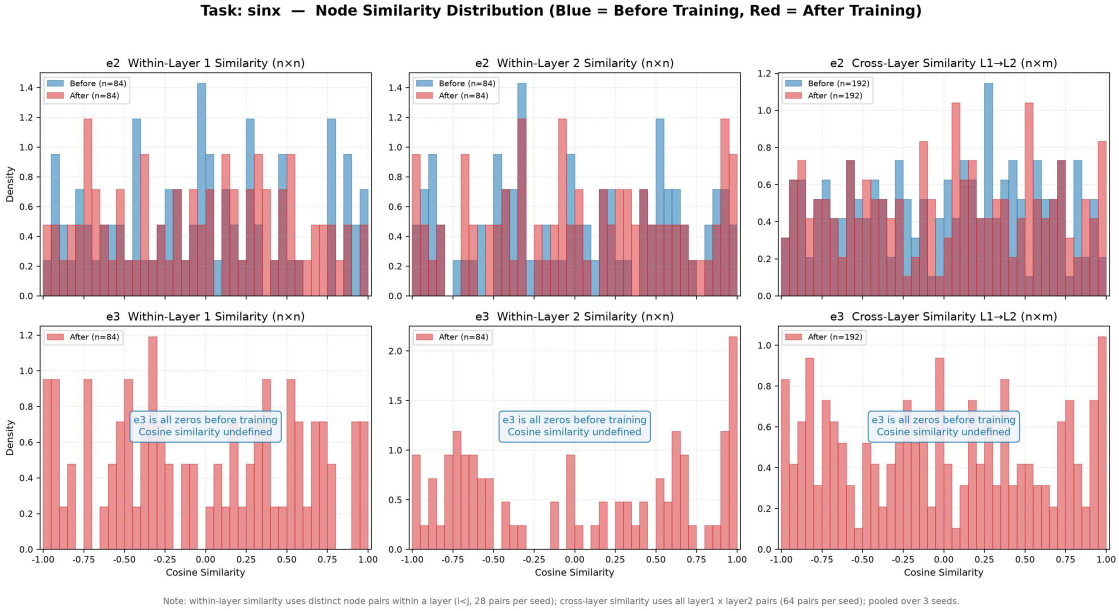


Figure 10. Node pairwise cosine-similarity distributions of e_2/e_3 on the sinx target (panel layout as in Figure 8)

Figure 11 visualizes the node-wise dynamic projection d in $K = 3$ space on sinx: each panel corresponds to one node, and the curve traces d as the input x moves from -1 to 1 (red–blue encodes the input value); the upper and lower pairs of rows correspond to layers 1 and 2, before and after training. After training, the trajectories of individual nodes bend from near-straight segments into branching, turning curves, showing that the dynamic reference pathway produces input-dependent directional changes for different nodes. Figure 12 further shows the variation of each K-channel component of the shared reference bottleneck $h = \mathrm{vec}(A)W_2$ with the input: after training the channels differ in slope and shape (e.g. the K1 channel of layer 2 changes from linear to S-shaped), being one source of nonlinearity for the reference direction d. Figure 13 reports the node-wise, per-K-component outputs of the dynamic projection $d = hW_3$, which

likewise bend nonlinearly after training, consistent with the reference trajectories. Figure 14 summarizes the final model fits on the three analytic tasks; all three seeds reach about 100% R^2, indicating that this small-scale configuration reproduces the three analytic functions and providing a background for the above geometric observations.

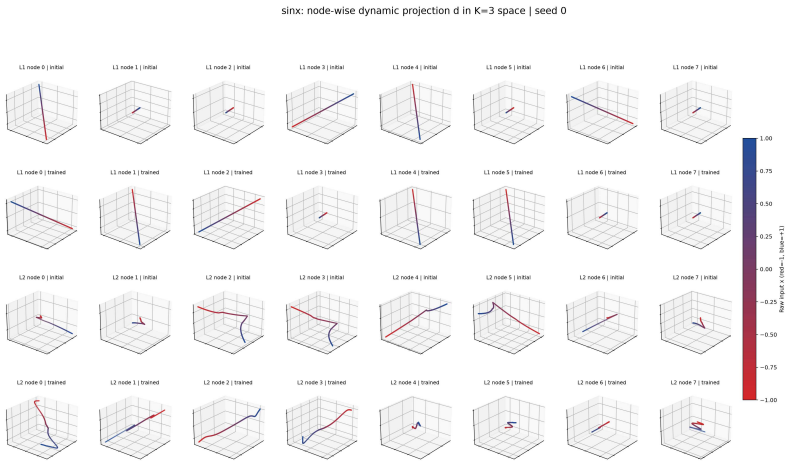


Figure 11. Trajectories of the node-wise dynamic projection d in $K = 3$ space on sinx (red–blue corresponds to input $-1 \to 1$; upper/lower groups are layers 1/2, each with an initial and a trained row)

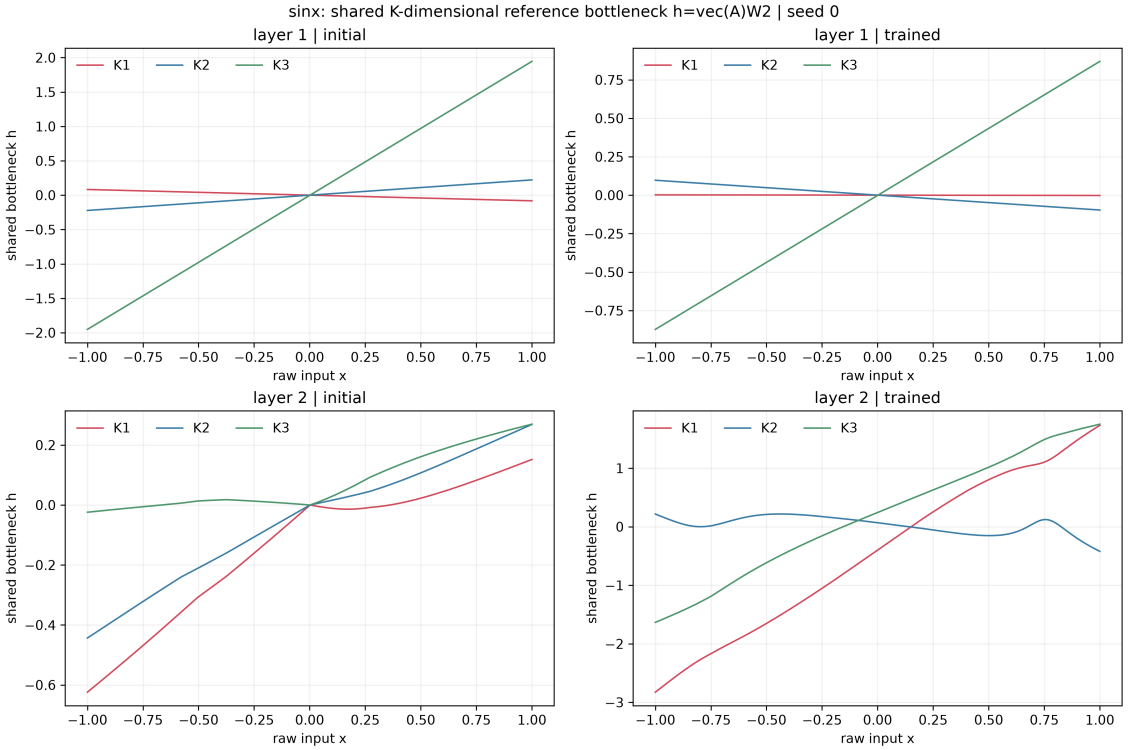


Figure 12. Components of the shared reference bottleneck $h = \mathrm{vec}(A)W_2$ along each K channel on sinx (left: initial; right: after training; top/bottom: layer 1/2)

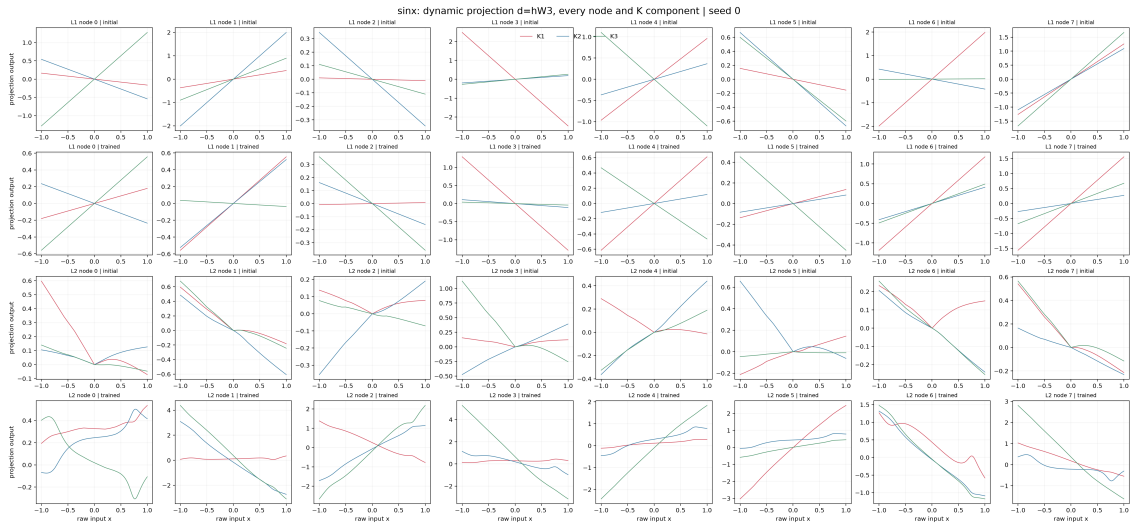


Figure 13. Node-wise, per-K-component outputs of the dynamic projection $d = hW_3$ on sinx (initial/trained groups, rows for the 8 nodes of layers 1/2)

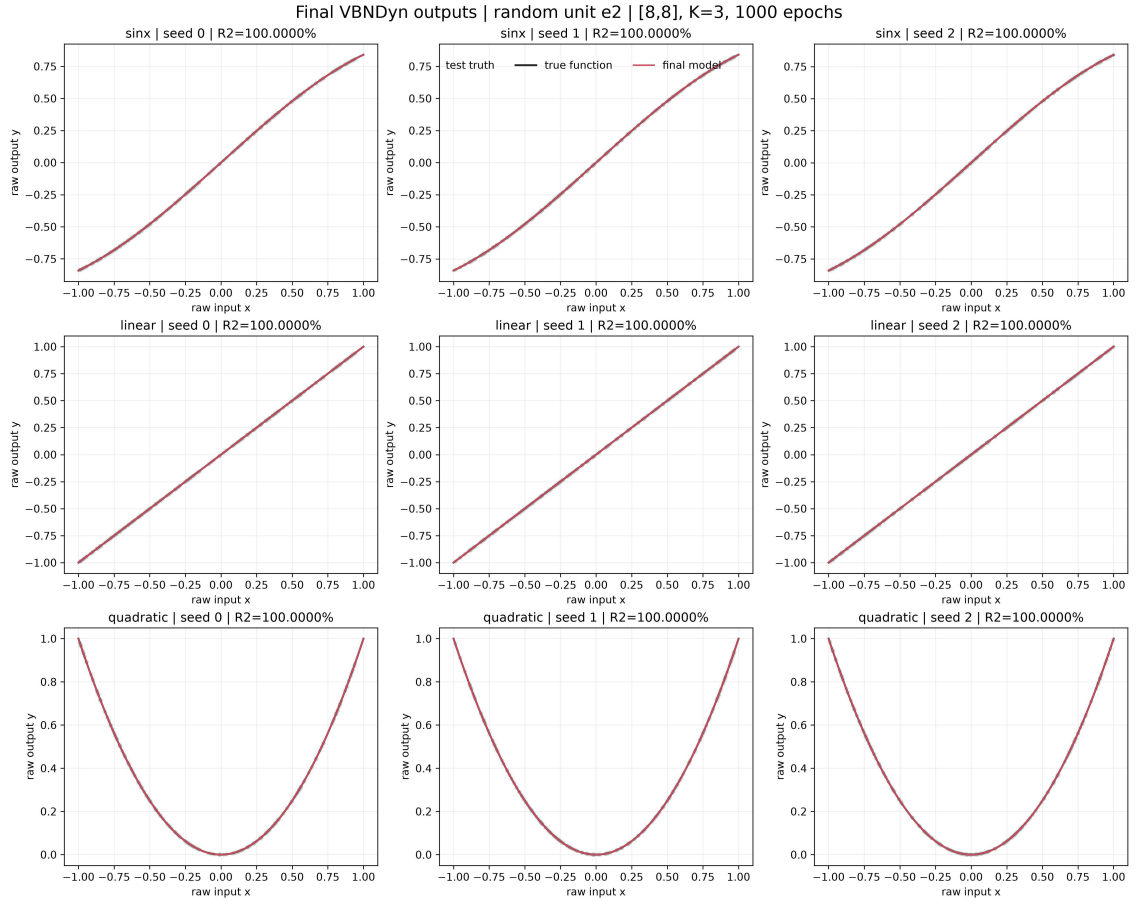


Figure 14. Final model fits on the analytic tasks sinx, linear and quadratic (3 seeds; R^2 in each panel title)

5 Backpropagation Gradient Analysis

This section derives the backpropagation of a single gated layer (excluding the output layer), and on this basis analyzes the composition of the parameter gradients, the computational and storage cost, and how the gradient flow shapes the manifold and softly clusters information. Notation follows Section 3: the layer input is a node matrix $A \in \mathbb{R}^{n\times K}$, the linear projection $V = AW \in \mathbb{R}^{m\times K}$, the vector fed to the gate $u = V + e_u$, the reference direction r (e_{ref} for VBN, $d + e_2$ for VBNDyn), the similarity $c = \cos(u, r)$ and gate $g = \text{clip}(1 + c, 0, 1)$, the gated signal $V' = g\,V\,P$, and the layer output $B' = b^{\text{bias}} + b^{\text{weight}} \odot V'$. All quantities are for a single sample with the batch dimension omitted; let $E = \partial\mathcal{L}/\partial B' \in \mathbb{R}^{m\times K}$ denote the upstream gradient.

5.1 Backpropagation through One Layer

From the definition of B' we obtain the gradients with respect to the gated signal and the biases directly:

$$\frac{\partial\mathcal{L}}{\partial V'} = b^{\text{weight}} \odot E, \qquad \frac{\partial\mathcal{L}}{\partial b^{\text{weight}}} = \sum_{\text{samples},\,j} V' \odot E, \qquad \frac{\partial\mathcal{L}}{\partial b^{\text{bias}}} = E. \tag{5.1}$$

Since $V' = g\,V\,P$, the gradient of the gate opening is the inner product, after channel mixing, of the upstream gradient with the pre-gate signal:

$$\frac{\partial\mathcal{L}}{\partial g_j} = \Big\langle b^{\text{weight}} \odot E_{j,:},\ (VP)_{j,:} \Big\rangle, \tag{5.2}$$

and $\partial g/\partial c = \mathbf{1}[-1 < c < 0]$ (g saturates at 1 for $c \geq 0$ and at 0 for $c \leq -1$, and is differentiable only in the transition region $(-1, 0)$). The effective gradient of the gating branch is therefore

$$s_j = \mathbf{1}[-1 < c_j < 0] \cdot \frac{\partial\mathcal{L}}{\partial g_j}, \tag{5.3}$$

and the gradients with respect to u_j and r_j, obtained by differentiating the cosine similarity, have a key **orthogonal-projection structure**:

$$\frac{\partial c_j}{\partial u_j} = \frac{\hat{r}_j - c_j\hat{u}_j}{\|u_j\|}, \qquad \frac{\partial c_j}{\partial r_j} = \frac{\hat{u}_j - c_j\hat{r}_j}{\|r_j\|}, \tag{5.4}$$

where $\hat{u} = u/\|u\|$ and $\hat{r} = r/\|r\|$ are unit directions. One can verify that $\hat{r} - c\hat{u}$ is orthogonal to $\hat{u}$, and $\hat{u} - c\hat{r}$ to $\hat{r}$: the gate-branch update to u always lies in the subspace **orthogonal to the current direction**—training rotates the direction without scaling its magnitude, leaving magnitude information entirely to the linear pathway W. Let

$$\bar{u}_j = s_j\,\frac{\hat{r}_j - c_j\hat{u}_j}{\|u_j\|}, \qquad \bar{r}_j = s_j\,\frac{\hat{u}_j - c_j\hat{r}_j}{\|r_j\|}. \tag{5.5}$$

The total gradient flowing into the linear projection is

$$\bar{V}_{j,:} = \underbrace{g_j\,(b^{\text{weight}} \odot E_{j,:})\,P^\top}_{\text{main path}} + \underbrace{\bar{u}_j}_{\text{gate branch}}, \qquad \frac{\partial\mathcal{L}}{\partial W_{ij}} = \sum_k A_{ik}\,\bar{V}_{jk}, \qquad \frac{\partial\mathcal{L}}{\partial e_{u,j}} = \bar{u}_j. \tag{5.6}$$

For the VBNDyn reference pathway, back-propagating through $r = d + e_2$, $d = \text{reshape}(hW_3)$ and $h = \text{vec}(A)W_2$ gives

$$\frac{\partial \mathcal{L}}{\partial e_{2,j}} = \bar{r}_j, \qquad \frac{\partial \mathcal{L}}{\partial W_3} = h^\top \text{vec}(\bar{r}), \qquad \frac{\partial \mathcal{L}}{\partial h} = \text{vec}(\bar{r})\, W_3^\top, \qquad \frac{\partial \mathcal{L}}{\partial W_2} = \text{vec}(A)^\top \frac{\partial \mathcal{L}}{\partial h}, \tag{5.7}$$

and the gradient continues to the previous layer along $\text{vec}(A)$. All gradient formulas above have been verified numerically against autograd term by term (maximum error $< 10^{-6}$).

5.2 Gradient Composition and Geometric Interpretation

From $\partial \mathcal{L}/\partial W_{ij} = \sum_k A_{ik} \bar{V}_{jk}$, the **magnitude of the weight update is jointly determined by four quantities**: the amount of incoming information at node i (the activation A_{ik} and its norm $\|A_i\|$), the gate opening g_j (i.e., the similarity c_j), the upstream gradient $E_{j,:}$, and the target-node direction $\hat{r}_j$ (appearing in $\bar{u}_j$). The main-path term is proportional to g_j, so information whose direction is closer to the reference, and hence more open, transmits gradient and updates weights more effectively; the gate-branch term is given by the direction residual $\hat{r}_j - c_j \hat{u}_j$, with magnitude inversely proportional to $\|u_j\|$, making updates to large-norm nodes gentler. Their combination lets W learn both "which inputs participate in which direction" and "how much to retain".

For e_u, e_2 and the reference pathway, the gradients are strictly constrained to the orthogonal subspace of the direction formed by (AW), i.e., the input plus bias: $\nabla e_u = \bar{u}$ and $\nabla e_2 = \bar{r}$ are respectively the orthogonal-projection residuals of r onto u and of u onto r, and ∇W_2, ∇W_3 are obtained by back-propagating $\bar{r}$ along the linear reference pathway. This constraint requires the target node, during training, to lie **within the manifold jointly constrained by the input nodes**: the learnable region of the gate is only $-1 < c < 0$, so learning nonlinearity must properly plan "how the gate nonlinearizes information as the input changes", rather than scaling without constraint. Taken together, the model constrains the shape of the manifold through gradient flow—W determines the projection direction of nodes on the manifold, $e_u/e_2/W_2/W_3$ determine the geometry of the reference direction, and the gate determines the degree of nonlinearization; during layer-wise propagation, information with similar directions receives similar gates and references and thus aligned gradients, so gradient flow and information are **softly clustered** into the same manifold region, consistent with the visualization in Section 4.6.

5.3 Time and Space Complexity

We analyze a single sample (following the batch-free notation of this section), with layer input/output node counts n, m and basis dimension K, ignoring lower-order additive and elementwise terms. The dominant per-layer forward costs are the linear projection $V = AW$ at $\mathcal{O}(nmK)$, the reference pathway $h = \text{vec}(A)W_2$ and $d = hW_3$ at $\mathcal{O}((n+m)K^2)$, and the gated mixing $V' = gVP$ at $\mathcal{O}(mK^2)$; hence one forward layer costs

$$\mathcal{O}\big(nmK + (n+m)K^2 + mK^2\big), \tag{5.8}$$

which is about $\mathcal{O}(h^2K)$ in the common setting $n = m = h$ with $K \ll h$. Backpropagation is of the same order as the forward pass (the transpose of a matrix product has the same cost), so one forward+backward layer is on the order of $\mathcal{O}(h^2K)$, and an L-layer network costs $\mathcal{O}(Lh^2K)$ in total training—about K times a same-width MLP (one MLP layer is $\mathcal{O}(h^2)$). In

space, the per-layer activations A, V, u, r, g, V' and gate intermediates must be cached, costing $\mathcal{O}((n+m)K)$; the parameter count is $\mathcal{O}(nm+(n+m)K+(n+m)K^2+K^2)$, dominated by the nm term shared with the MLP, with the remaining K-channel terms as lower-order overhead.

6 Conclusion

This paper constructs information propagation explicitly in a high-dimensional basis space: the input vector determines where a node rotates and changes in space, the channel-mixing strength depends on the associated weights and magnitude, and the cosine gate depends only on the pattern (direction). The burden carried by the weight matrix is thereby decoupled into projection propagation over spatial nodes, and decoupling magnitude from direction gives nodes an emergent voting capability, so that the importance and the pattern of different weights can be distinguished separately.

Based on this spatially structured projection propagation, node information constructs the representational meaning of each dimension: different directions represent different patterns, spatial variation remains continuous wherever the gate is continuous, and while preserving smoothness the model can express asymmetric forms and rapidly capture abruptly changing directional forms, thereby more readily inductively capturing high-dimensional complex variations with limited parametric complexity. Future work will apply this method to models such as Transformers and explore more reasonable, more regularized and more expressive manifolds for gated projections.

A Experimental Configuration

Section 4 summarizes the experimental setup; this appendix lists the full configuration for reproducibility. All reported numbers are obtained from the raw experimental data and reproduction scripts supplied with the paper.

A.1 Data and Splits

Table 5. Tasks and data splits

Item	Configuration
Real datasets	Nine UCI regression tasks: abalone, airfoil_self_noise, auto-mpg, Concrete_Data, energy_efficiency, kin8nm, winequality-red, winequality-white, yacht_hydrodynamics
Synthetic functions	Three: sin20x (y=sin 20x), complex_trend, complex_trend2
Splits	80% training / 20% test; inputs standardized; all models share the same split

A.2 Optimization and Training

A.3 Model Structure and Initialization

The main configuration of VBN and VBNDyn is $K = 4$, hidden width 32 and 4 layers. The weights use variance-corrected initialization (gate_var), whose standard deviation is scaled by the input dimension and the expected squared gate (see Section 3.4). e_u (e_3) is initialized to zero; the e_{ref} of VBN and the e_2 of VBNDyn use exactly the same initialization, namely a Gaussian sampled per node followed by L_2 normalization. W_2 and W_3 are initialized with variances $1/(nK)$ and $1/K$, respectively; P is initialized to the identity, b^{weight} to ones and b^{bias} to zero.

Table 6. Optimization and training hyperparameters

Item	Configuration
Optimizer	Adam
Learning-rate schedule	Cosine decay, minimum learning rate 1e-5
Initial learning rate	VBN, VBNDyn, parameter-matched MLP: 1e-2; KAN: 3e-3; SIREN: 1e-3
Training epochs	300
Batch size	512 (the remainder is merged into the final batch)
Random seeds	0, 1, 2, 3 (four seeds)
Metric	Coefficient of determination R^2 (in percent)

A.4 Baseline Configurations

The parameter-matched MLP searches a single hidden width over four layers per task so that its parameter count matches VBN or VBNDyn. KAN uses SiLU basis functions with B-splines (grid=5, degree=3, grid range -3 to 3) and is matched to the parameter count of VBNDyn. SIREN uses sinusoidal activations (omega0=5) and is matched to the parameter count of VBN. Tree baselines include CatBoost, LightGBM, XGBoost, RandomForest, ExtraTrees, HistGB and SVR.

A.5 Significance Tests and Ablation

Significance is assessed by paired t-tests per task over four seeds, with Holm correction [10] controlling for multiple comparisons; Wilcoxon signed-rank tests and Cohen's d_z are also computed (details in the supplied data). The depth ablation is conducted on sin20x over layers 1–18 for VBN and VBNDyn only.

A.6 Mechanism-Experiment Configuration

The mechanism analysis is an independent small-scale visualization experiment: $K = 3$, 8 nodes per layer, 2 layers, 1000 epochs and 3 random seeds (0, 1, 2), on the analytic tasks sinx, linear and quadratic whose ground truth is known. The differences from the main configuration and their rationale are given in Section 4.6.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 323(6088):533–536, 1986.

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In ICLR, 2022.

[3] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark. KAN: Kolmogorov–Arnold networks. arXiv:2404.19756, 2024.

[4] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In NeurIPS, 33:7462–7473, 2020.

[5] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: Unbiased boosting with categorical features. In NeurIPS, 31:6638–6648, 2018.

[6] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In NeurIPS, 30:3146–3154, 2017.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In KDD, pages 785–794, 2016.

[8] L. Breiman. Random forests. Machine Learning, 45(1):5–32, 2001.

[9] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In NeurIPS, 9:155–161, 1997.

[10] S. Holm. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics, 6(2):65–70, 1979.

[11] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In NeurIPS, 30:3856–3866, 2017.