INFINITE FROM FINITE: RECURSIVE BASIS NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Depth is a fundamental axis of neural scaling, yet deep neural networks typically obtain deeper computation by stacking more parameterized layers, tightly coupling functional depth to parameter depth. In this paper, we explore a new route to neural scaling, where functional depth grows through synthesis from a finite parameter reservoir. We propose **Recursive Basis Networks (RBNs)**, which achieve *dynamic functional synthesis* by recursively recomposing a shared basis dictionary through input-conditioned states. This enables RBNs to synthesize heterogeneous functional transformations without layer-specific parameters, turning depth into a runtime-controllable dimension while keeping the transformation parameter budget fixed. Therefore, RBNs transform parameter accumulation into functional composition, yielding a large compositional space from finite primitives. Across diverse approximation and representation-learning settings, RBNs demonstrate the effectiveness of dynamic functional synthesis, with strong expressivity, parameter efficiency, and flexible depth scaling. These results establish it as a general principle for deriving expandable functional complexity from finite parameterization. Our code will be released on our website.

1 INTRODUCTION

The representation capacity of deep neural networks (DNNs) is tied to their functional depth. Theoretically, successive heterogeneous functional transformations expand the complexity of the representable function space with depth (Montúfar et al., 2014; Raghu et al., 2017), a principle borne out by the empirical success of deep architectures such as ResNets and Transformers (He et al., 2016; Vaswani et al., 2017). However, in such layer-stacked architectures, this principle induces a structural coupling between functional transformations (*functional depth*) and allocated parameter layers (*parameter depth*). Each additional heterogeneous transformation is instantiated by an additional set of layer-specific parameters, binding functional depth to a parameter budget fixed at training time (Cybenko, 1989; Pinkus, 1999). As a result, the model is incapable of executing a chain deeper than its instantiated layer count (Fig.1a-b, **finite from finite**). This constraint becomes particularly pronounced in foundation models, where task complexity varies widely while pre-trained parameters remain fixed (Kaplan et al., 2020; Bommasani et al., 2021). In contrast, human cognition and linguistics rely on the principle of "discrete infinity", allowing for the generation of an infinite variety of concepts from a finite set of primitives (Humboldt, 1988; Chomsky, 2002; Hauser et al., 2002) (**infinite from finite**). This motivates a question: *can heterogeneous functional depth be decoupled from parameter depth, enabling an expandable representable function space from finite parameters?*

Function approximation theory establishes that complex nonlinear functions can be decomposed into a set of basis functions (Pinkus, 1999; Kolmogorov, 1963). This theory has historically been operationalized by Radial Basis Networks (Park & Sandberg, 1991) and has recently been revitalized by Kolmogorov-Arnold Networks (KANs) (Liu et al., 2025), yielding an implication: the network's representation capacity (nonlinear function) is able to be driven by the compositional formulation of bases (Fig.1c). Therefore, the stacking of parameters to construct functional transformations can be reformulated as the dynamic compositions of basis functions at each step, echoing the principle of "discrete infinity" by synthesizing infinite functional variations from finite components. Specifically, if a network maintains a dictionary of basis functions, although the parameter budget is fixed, it will synthesize distinct functions at each step via the dynamic composition of bases in the dictionary (Fedus et al., 2022; Chen et al., 2022) (Fig.1d). By applying this recursively, the compositional paradigm executes chains of variable length, enabling functional depth to scale independently of parameter depth and deriving vast complexity from a finite set of basis functions.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

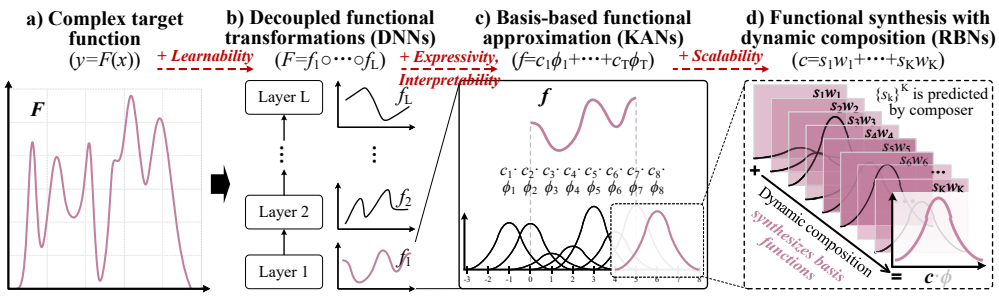


Figure 1: **From Static Parameter Stacking to Dynamic Functional Synthesis.** RBNs build upon historical milestones to empower DNNs with an expandable functional space, where representation power scales independently of the parameter budget. a) The target: a complex function. b) DNNs factorize complexity into a stack of parameter layers coupling functional depth with parameter inflation. c) KANs improve expressivity and interpretability with learnable basis functions, but remain constrained by static parameter stacking. d) In RBN, each basis function is dynamically synthesized from candidate profiles in a shared dictionary. This compositionality, from candidates to bases and bases to functions, unlocks a large functional space from a finite shared parameter dictionary.

In this paper, we propose the **Recursive Basis Networks (RBNs)**, a neural architecture that operationalizes this principal paradigm by replacing the one-to-one mapping of functional depth to parameter depth with a shared basis dictionary that is dynamically composed to synthesize functions at each transformation step. RBN consists of three components: **1)** *Learnable Basis Dictionary* maintains a set of basis functions and their learnable coefficients, decoupling the parameter budget from the complexity of individual functional transformations dynamically. **2)** *Dynamic Composer* composes bases from the basis dictionary at each step to synthesize distinct functions via predicting input-conditioned compositional states. **3)** *Recursive Depth Unrolling* applies this compositional mechanism iteratively over a variable number of steps using shared parameters, such that functional depth scales with the unrolling steps rather than the allocated parameter layers.

RBN accesses an expandable function space within a fixed parameter budget, enabling scalable expressivity beyond foundational paradigms. Basis-based approaches such as KANs (Liu et al., 2025) improve single-layer approximation and interpretability through basis functions, but remain constrained within a fixed-depth stack. Conditional-computation methods, like Mixture-of-Experts (MoE) (Fedus et al., 2022; Chen et al., 2022; Cai et al., 2025), utilize input-dependent parameterization or routing to improve dynamic adaptability, yet operate within statically instantiated architectures. Recursive or weight-shared models such as RNNs (Sherstinsky, 2020) and Universal Transformers (Dehghani et al., 2019) decouple execution depth from parameter count by reusing the same parameterized transformation across recursive steps, limiting the diversity of functional mappings (Bai et al., 2019; Pascanu et al., 2013). **In contrast**, RBN synthesizes distinct functional transformations at each recursive step through input-conditioned composition over a shared basis dictionary, enabling heterogeneous transformations without introducing layer-specific parameters. This yields a new capability, *dynamic functional synthesis*, where distinct transformations are achieved across steps, evolving representations within a fixed parameter budget.

This paper formulates a neural architecture that enables functional depth scaling independently of parameter depth, providing an expandable representable function space in DNNs. Our contributions are threefold: **(1)** We formulate a compositional paradigm in which complex transformations are constructed through the composition of basis functions, preserving the expressivity of basis-based approaches while enabling dynamic functional synthesis. **2)** We propose Recursive Basis Networks (RBNs) as a constructive realization, synergizing a basis dictionary with a composer within a recursive depth unrolling architecture, achieving scalable depth under a fixed parameter budget. **3)** Our empirical evidence across function approximation, foundation model pre-training, and diverse ablation and analysis demonstrates RBN's strong expressivity and competitive representation versatility.

2 Recursive Basis Networks (RBNs)

RBNs enable function synthesis, decoupling functional depth from the parameter budget. For clarity, we formulate in a scalar form and omit width N, which is extended channel-wise when instantiating.

2.1 THEORETICAL FOUNDATIONS OF FUNCTIONAL SYNTHESIS

Functional Decomposition in Deep Neural Networks The representation capacity of DNNs can be characterized by decomposing a complex target function $F : \mathcal{X} \to \mathcal{Y}$ along two dimensions, i.e., *functional depth* and *basis expansion*. A complex function F is factorized into a sequence of L discrete transformations $\{f_1, \ldots, f_L\}$ where functional depth distributes the overall complexity across multiple functional transformation steps (Montúfar et al., 2014; Raghu et al., 2017). Approximation theory (Pinkus, 1999; Kolmogorov, 1963) indicates that any nonlinear function within a compact domain can be represented as a linear composition of basis functions. Therefore, each functional transformation f_l is further decomposable into a set of basis functions like KANs (Liu et al., 2025). Formally, let $\{\phi_1, \ldots, \phi_T\}$ denote a basis set, each transformation f_l is able to be synthesized by these bases and the overall DNN mapping F will be formulated as:

$$F = \overbrace{f_1 \circ f_2 \circ \cdots \circ f_L}^{\text{functional depth}} \quad \text{with} \quad f_l = \overbrace{c_{l,1}\phi_1 + c_{l,2}\phi_2 + \cdots + c_{l,T}\phi_T}^{\text{basis expansion}}, \tag{1}$$

where $c_{l,t}$ is a coefficient that determines the contribution of the t-th basis function at the l-th depth, "$\circ$" is the stacking of the transformation f_l. By defining a coefficient matrix $\mathbf{C} = \{c_{l,t}\}^{L,T} \in \mathbb{R}^{L\times T}$, where the l-th line $\mathbf{c}_l = [c_{l,1}, \ldots, c_{l,T}]$ is the coefficient vector for l-th transformation, the mapping F can be formalized as the basis-derived operators with the basis vector $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_T]^\top$:

$$F \sim \begin{bmatrix} c_{L,T} & c_{L,T-1} & \cdots & c_{L,1} \\ c_{L-1,T} & c_{L-1,T-1} & \cdots & c_{L-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1,T} & c_{1,T-1} & \cdots & c_{1,1} \end{bmatrix} \begin{bmatrix} \phi_T \\ \phi_{T-1} \\ \vdots \\ \phi_1 \end{bmatrix}. \tag{2}$$

In this formulation, DNNs, especially for KANs, are able to instantiate $\mathbf{C}$ as a collection of static parameters, binding functional depth to the layer count. This imposes a one-to-one coupling between functional depth and parameter allocation, constraining the attainable representable function space to those realizable within a statically fixed parameter budget.

Functional Synthesis via Composition The above formulation enables a transformative heuristic: instead of assigning a coefficient vector of the basis functions to each layer, all coefficients are pooled into a dictionary, from which entries are selected and composed to synthesize each transformation f_l. When each of the T basis functions draws from one of K candidate coefficient profiles, i.e., under top-1 composition, the number of discrete candidate transformations can grow up to $\mathcal{O}(K^T)$. More general top-k composition further relaxes these discrete choices by continuously composing multiple candidate profiles. Therefore, RBN provides a combinatorially large compositional space that far exceeds the preset depth L. When $K^T \gg L$, the number of available candidate routing patterns is larger than the required recursive depth, allowing a fixed parameter budget to support an expandable family of transformations.

Formally, each coefficient $c_{l,t}$ is factorized as $c_{l,t} = \mathbf{w}_t^\top \mathbf{s}_t^l$, where $\mathbf{W} = \{\mathbf{w}_t = [w_{1,t}, \cdots, w_{K,t}]\}_{t=1}^T \in \mathbb{R}^{K\times T}$ is the shared dictionary of K candidate coefficient profiles independent of layers, and $\mathbf{S}^l = \{\mathbf{s}_t^l = [s_{1,t}^l, \cdots, s_{K,t}^l]\}_{t=1}^T \in \mathbb{R}^{K\times T}$ is the dynamic compositional states of coefficient profiles targeted for the l-th transformation layer f_l. Therefore, the overall mapping F is able to be re-formulated as multiplications among the $\mathbf{W}$, $\mathbf{S}$, and $\boldsymbol{\phi}$:

$$F \sim \begin{bmatrix} \mathbf{w}_T^\top \mathbf{s}_T^L & \mathbf{w}_{T-1}^\top \mathbf{s}_{T-1}^L & \cdots & \mathbf{w}_1^\top \mathbf{s}_1^L \\ \mathbf{w}_T^\top \mathbf{s}_T^{L-1} & \mathbf{w}_{T-1}^\top \mathbf{s}_{T-1}^{L-1} & \cdots & \mathbf{w}_1^\top \mathbf{s}_1^{L-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_T^\top \mathbf{s}_T^1 & \mathbf{w}_{T-1}^\top \mathbf{s}_{T-1}^1 & \cdots & \mathbf{w}_1^\top \mathbf{s}_1^1 \end{bmatrix} \begin{bmatrix} \phi_T \\ \phi_{T-1} \\ \vdots \\ \phi_1 \end{bmatrix}, \mathbf{w}_t^\top \mathbf{s}_t^l = \begin{bmatrix} w_{K,t} \\ w_{K-1,t} \\ \vdots \\ w_{1,t} \end{bmatrix}^\top \begin{bmatrix} s_{K,t}^l \\ s_{K-1,t}^l \\ \vdots \\ s_{1,t}^l \end{bmatrix}. \tag{3}$$

The dynamic compositional states $\mathbf{S}_l$ targeted for each layer f_l are predicted by an input-conditioned lightweight network like MoEs Cai et al. (2025). This factorization decouples the network's representation capacity from its parameter constraints: the number of dictionary coefficients in this compact formulation is $O(T \times K)$ and is independent of the functional depth L. In practice, L will become an additional input during inference to control the number of recursive steps. By recursively predicting new states $\mathbf{S}$, the network dynamically instantiates each layer's transformation from a compositional space with exponentially many discrete routing configurations, yielding what we term *dynamic functional synthesis*.

Theorem 1 (Approximation Error Decomposition of Dynamic Functional Synthesis)**.** *Let $\mathcal{F}$ be a target function space, and $F \in \mathcal{F}$ admit a depth-L compositional representation as in Eq.1. Suppose each transformation is synthesized via a basis expansion $f_l = \sum_{t=1}^{T} c_{l,t}\phi_t$, where $\{\phi_t\}_{t=1}^{T}$ constitutes a basis set. The coefficients are dynamically composed from a shared parameter dictionary $\mathbf{W} \in \mathbb{R}^{K\times T}$ as $c_{l,t} = \mathbf{w}_t^\top \mathbf{s}_t^l$, with compositional states $\mathbf{s}_t^l \in \mathbb{R}^K$. Assume that the finite basis expansion incurs a truncation error $\mathcal{E}_{\text{basis}}(T)$ and that the shared dictionary incurs a composition error $\mathcal{E}_{\text{com}}(K)$. Under Lipschitz-continuous dependence of the composed mapping on its coefficients,*

$$\|F - F_{\text{RBN}}^{T,K}\|_\infty \le C_1\mathcal{E}_{\text{basis}}(T) + C_2\mathcal{E}_{\text{com}}(K), \tag{4}$$

where $C_1, C_2 > 0$ depend on the target function, basis functions, and composition stability of the recursive mapping. The parameter budget of $F_{\text{RBN}}^{T,K}$ is independent of the functional depth L.

Remark 1 (Compositional Space of Functional Synthesis). The $\mathcal{O}(K^T)$ quantity characterizes the number of discrete candidate compositions underlying functional synthesis. Under top-1 composition, each of the T basis functions independently selects one of K candidate coefficient profiles, yielding up to K^T discrete routing configurations. Therefore, $K^T \gg L$ provides a sufficient *counting* condition for accommodating distinct routing patterns across an L-step composition. Top-k composition further replaces each hard selection with a continuous mixture over the selected candidate profiles, providing a continuous relaxation of these discrete configurations.

Theorem 1 separates the approximation error into two sources (Boyd & Vandenberghe, 2004): **1)** *Basis Truncation Error ($C_1\mathcal{E}_{\text{basis}}(T)$):* This reflects the capacity of the T bases to approximate the required functional transformations. The behavior of $\mathcal{E}_{\text{basis}}(T)$ with respect to T depends on the approximation properties of the selected basis family and the regularity of the target transformations (Pinkus, 1999; Schumaker, 2007). **2)** *Dictionary Composition Error ($C_2\mathcal{E}_{\text{com}}(K)$):* This measures the residual incurred when the ideal coefficients are synthesized from the shared dictionary W (Gray & Neuhoff, 2002). Increasing the dictionary capacity K can improve the coverage of the coefficient space and thereby reduce $\mathcal{E}_{\text{com}}(K)$. This approximation property is distinct from the compositional space in Remark 1: K^T characterizes the number of discrete candidate compositions available for functional synthesis, whereas $\mathcal{E}_{\text{com}}(K)$ measures the accuracy with which the shared dictionary covers the coefficients required by the target transformations. These properties allow RBN to synthesize diverse transformations from a fixed parameter reservoir, shifting the expressivity bottleneck from parameter stacking to basis composition. We leave the proof and discussion in App.B.1.

2.2 RBN *for Dynamic Functional Synthesis*

Building upon the theoretical foundations of functional synthesis, we instantiate the **Recursive Basis Networks (RBNs)**. Our RBN architecture synergizes a basis dictionary with a dynamic composer within a recursive depth unrolling architecture, achieving scalable depth under a fixed parameter budget. As shown in Fig.2, its powerful ability stems from three core components:

Dynamic Composer explores the compositional space $\mathcal{O}(K^T)$ by predicting the dynamic compositional states $\mathbf{S} \in \mathbb{R}^{K\times T}$ to synthesize each transformation f (Fig.2a). Analogous to the routing mechanism in MoE (Fedus et al., 2022), it leverages a lightweight linear projection to dynamically assign input-conditioned weights to the shared basis dictionary. Formally, let $\pi \in \mathbb{R}^{N\times KT}$ be a learnable projection matrix, where N is the input feature width. Given the input representation $\mathbf{h} \in \mathbb{R}^N$, the composer projects it onto $K \times T$ directions to obtain the raw routing logits $\mathbf{Z} = \mathbf{h}\pi \in \mathbb{R}^{KT}$. We then reshape these logits into a set of T distinct vectors, yielding $\mathbf{Z} \in \mathbb{R}^{K\times T} = [\mathbf{z}_1, \ldots, \mathbf{z}_T]$, where each $\mathbf{z}_t \in \mathbb{R}^K$ corresponds to the scores for the K candidate profiles of the t-th basis function. To obtain the final compositional states, we apply a softmax independently over the K candidate profiles for each basis function t:

$$\mathbf{s}t = \sigma\text{softmax}\left(\frac{\mathbf{z}_t}{\sqrt{K}}\right) \in \mathbb{R}^K, \quad \mathbf{Z} = \mathbf{h}\pi \in \mathbb{R}^{KT} = [\mathbf{z}_1, \ldots, \mathbf{z}_T] \in \mathbb{R}^{K\times T}. \tag{5}$$

Here, $\mathbf{s}_t$ defines the normalized compositional weights over the K candidate coefficient profiles of basis t. The factor $1/\sqrt{K}$ is fixed a priori in all experiments to normalize the aggregate routing-logit magnitude as K varies and stabilize softmax routing, following the general scaling principle of scaled dot-product attention (Vaswani et al., 2017).

Learnable Basis Dictionary serves as the parameter reservoir for the entire network, operationalizing both the basis functions ϕ and the dictionary of candidate coefficient profiles $\mathbf{W} \in \mathbb{R}^{K\times T}$ as

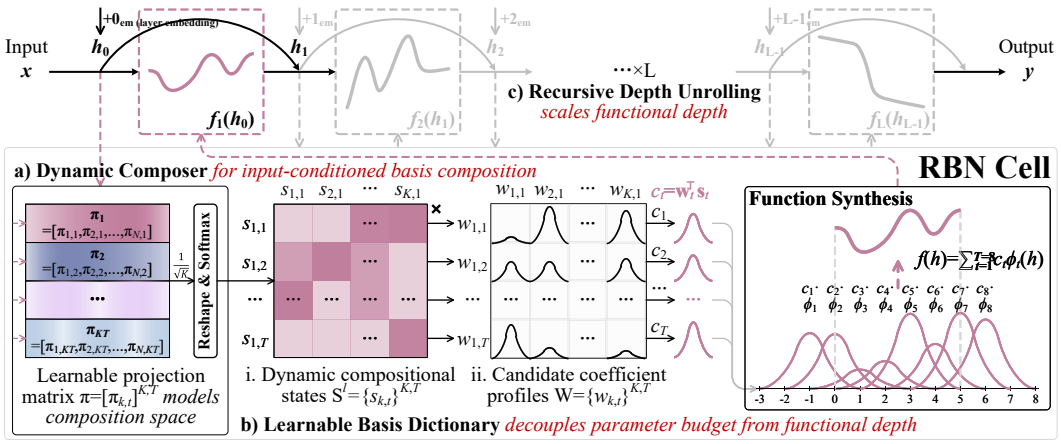


Figure 2: **Overview of our Recursive Basis Network (RBN) Architecture.** The RBN comprises three core components: **a)** a *Dynamic Composer* that projects the hidden representation onto $T \times K$ directions via a learnable matrix $\pi \in \mathbb{R}^{N \times KT}$ to predict the dynamic compositional states $\mathbf{S}^l = \{s_{k,t}\}^{K,T}$; **b)** a *Learnable Basis Dictionary* that maintains a shared set of candidate coefficient profiles $\mathbf{W} = \{w_{k,t}\}^{K,T}$, from which layer-specific coefficients $c_t = \mathbf{w}_t^\top \mathbf{s}_t$ are synthesized to assemble each functional transformation $f_l(\mathbf{h}) = \sum_{t=1}^{T} c_t \phi_t(\mathbf{h})$ over parameterized B-spline bases; and **c)** a *Recursive Depth Unrolling* mechanism that iteratively applies the RBN Cell (a and b) for L steps via a residual stream, enabling parameter-free scaling of functional depth at inference time.

indicated in Eq.3. It decouples the network's parameter budget from its functional depth (Fig.2b). **1)** *For basis functions* $\phi_t(\cdot)$, they can be flexibly instantiated in various forms such as polynomials, radial basis functions (RBFs), splines, or neural basis functions. Following KANs (Liu et al., 2025), we adopt T parameterized B-splines as the basis functions $\phi_t(\cdot)$, defined over a fixed grid. Each $\phi_t(\cdot)$ corresponds to a spline basis localized on the grid, providing strong nonlinear expressivity and interpretability. **2)** *For candidate coefficient profiles* $\{w_{k,t}\}_{k=1,t=1}^{K,T}$, K redundant coefficients corresponding to each basis are maintained in the shared dictionary $\mathbf{W}$, forming an overcomplete set to learn reusable functions prepared for compositions across depths. During training, $\mathbf{W}$ is learned across all recursive steps, forcing the emergence of functional patterns reused throughout the network. This component's parameter cost is bounded by $\mathcal{O}(K \times T)$ independent of depth L.

Recursive Depth Unrolling scales functional depth with a variable number of recursive steps (Fig.2c). With the coefficients of the bases synthesized $c_{l,t} = \mathbf{w}_t^\top \mathbf{s}_t^l$, the transformation step l is materialized as $f_l(\cdot) = \sum_{t=1}^{T} c_{l,t} \phi_t(\cdot)$. The architecture executes this via a recursive unrolling loop. To stabilize the forward trajectory and mitigate gradients associated with deep recursive systems (Pascanu et al., 2013), we embed the synthesized function within a residual stream:

$$\mathbf{h}^l = \mathbf{h}^{l-1} + f_l(\mathbf{h}^{l-1} + l_{\text{em}}) = \mathbf{h}^{l-1} + \sum_{t=1}^{T} \left(\mathbf{w}_t^\top \mathbf{s}_t^l\right) \phi_t(\mathbf{h}^{l-1} + l_{\text{em}}), \tag{6}$$

where l_{em} denotes the embedding of the recursive step l. The RBN cell is shared across all recursive steps: the Learnable Basis Dictionary (W, ϕ) and the Dynamic Composer π are independent of the step index l. Consequently, increasing the unrolling depth L does not instantiate additional transformation parameters, making L a runtime-controllable dimension rather than a parameterized architectural depth. Therefore, an RBN trained at L_{train} can be evaluated at a deeper or shallower L_{test} without modifying the shared RBN parameters.

Remark 2 (Computation-Capacity Trade-off). RBN shifts the cost of increasing functional depth from parameter allocation to recursive computation. For a network with feature width N, basis size T, dictionary size K, and depth L, its parameter budget is $\mathcal{O}(N^2TK)$ and remains independent of L, while computation scales as $\mathcal{O}(BN^2TK \cdot L)$ for batch size B. In return, each recursive step can synthesize an input-conditioned transformation from the combinatorially large compositional space characterized in Remark 1. Therefore, increasing L does not require instantiating new parameterized layers, but instead repeatedly explores and recomposes a fixed parameter reservoir to realize heterogeneous functional transformations. This contrasts with layer-stacked MLPs (LeCun et al., 2015) and KANs (Liu et al., 2025), whose additional functional depth requires additional layer-specific

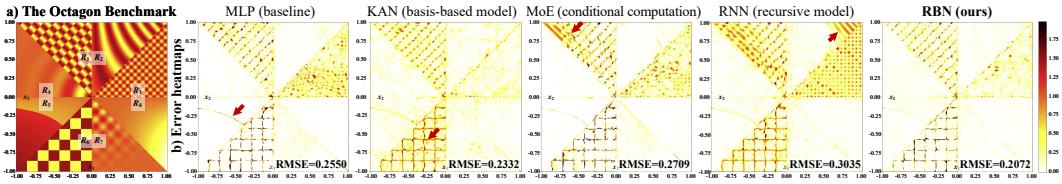


Figure 3: **Visualization of the Octagon Benchmark** Each heatmap (b) shows the error between the predicted function approximation and ground truth for MLP, KAN, MoE, RNN, and our RBN (optimal). RBN achieves the lowest overall RMSE while maintaining accuracy across all functional regions, showing the advantage of its dynamic function synthesis mechanism over fixed baselines.

parameters. The computational overhead can be reduced to $\mathcal{O}(BN^2Tk \cdot L)$ through top-k ($k < K$) activation (App.C.2), with modest empirical performance trade-offs. Details are in App.B.2.

Summary of the Properties In redesigning the relationship between parameters and functional transformations, RBNs exhibit highly desirable properties: **1)** *Parameter-Free Depth Scaling/Adaptation (analyzed in Sec.3.3):* Because the RBN cell is shared across recursive steps, increasing the functional depth L does not instantiate additional transformation parameters. Therefore, functional depth becomes a controllable variable that can be scaled via adjusting L and adapted via halting (Dehghani et al., 2019). **2)** *Transformation-Level Heterogeneity (analyzed in App.F.2):* RBN dynamically composes a shared basis dictionary to synthesize heterogeneous functional transformations across steps. Unlike weight sharing (e.g., Universal Transformer), which reuses the same transformation parameters across steps, this enables functional diversity without layer-specific parameters. **3)** *Inherent Interpretability (analyzed in App.F.1):* By grounding each recursive transformation in basis sets (e.g., B-splines), RBNs inherit the transparent, decomposable nature of basis-based models like KANs. The mappings synthesized at depth l is able to be extracted and analyzed.

3 Experiments

We validate the parameter efficiency, expressivity, and elastic computation capabilities of RBNs via low-dimensional manifold approximation, high-dimensional visual representation learning, and comprehensive ablation studies. More details and results are in App.D and E.

3.1 Toy Study: Functional Approximation on Manifolds

The Octagon Benchmark To rigorously validate the expressivity and local adaptability of RBNs, we construct a spatially heterogeneous function approximation benchmark termed the **"Octagon"**, akin to a fighting cage (Fig.3a). The name reflects the topological structure of our arena: the input domain $(x_1, x_2) \in [-1, 1]^2$ is partitioned via polar coordinates (r, θ) into eight disjoint angular regions $\mathcal{R}_1$ to $\mathcal{R}_8$, each hosting a qualitatively distinct function (formulated in App.D.1). The regions collectively cover a broad spectrum of approximation-theoretic challenges: high-frequency oscillations, non-separable multiplicative interactions, sign discontinuities, radially decaying envelopes, and pure polynomial cross-terms. These regions span a broad spectrum of approximation-theoretic challenges that are challenging to address with any single architectural prior, making the benchmark a sensitive diagnostic of a model's functional coverage across heterogeneous manifold topologies.

Implementation We evaluate RBN against MLP, KAN (Liu et al., 2025), MoE (Fedus et al., 2022), and RNN (Sherstinsky, 2020). Models are trained on $10,000$ samples, validated on $2,000$ samples, and tested on a dense 300×300 grid to predict $F(x_1, x_2)$. To analyze capacity trade-offs, we sweep across depths $L \in \{2, 4, 12\}$ and varying widths, yielding parameter from $\sim$ 300 to 185M and 806M, respectively. RBN uses a basis size $T = 12$ and dictionary $K = 8$. Performance is reported as global and per-region RMSE. All models are trained for $1,800$ epochs (details in App.D.2).

Fig.3 and Fig.4 jointly reveal the expressivity and scalability characteristics of each architecture on the Octagon Benchmark. We structure our analysis around three observations.

Qualitative Analysis: Overcoming Topological Imbalances As shown by the regional error distributions (Fig.3b), static baselines exhibit severe topological imbalances, performing well on specific functional regions at the expense of others. Deep MLPs mitigate checkerboard discontinuities

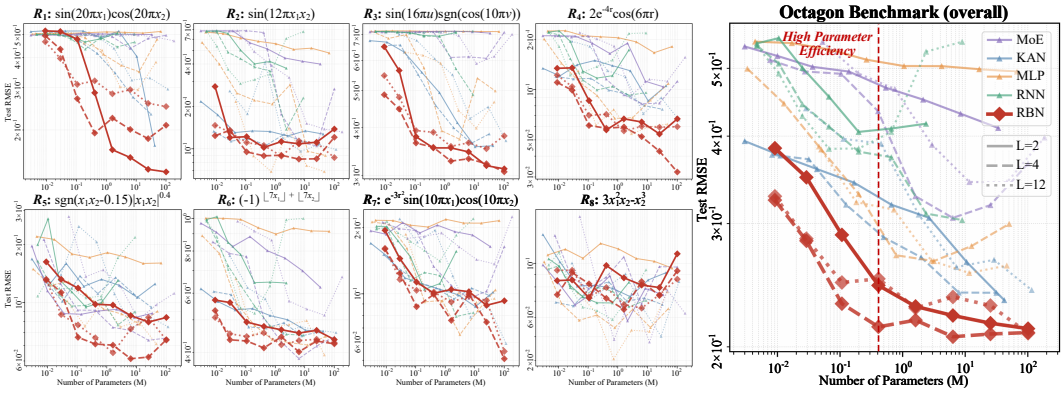


Figure 4: **Scaling Behaviors** across all eight regions ($\mathcal{R}_1$-$\mathcal{R}_8$) and the overall benchmark as a function of the number of parameters. RBN (red) consistently achieves the lowest RMSE across nearly all regions and parameter budgets, with a stable descending scaling trajectory that contrasts with the saturation or degradation observed in competing architectures at larger model sizes.

($\mathcal{R}_6$) but struggle with extreme high-frequency oscillations in $\mathcal{R}_1$, while KANs perform well on polynomial interactions ($\mathcal{R}_8$) but degrade substantially under compounded complexity and rotated discontinuities in $\mathcal{R}_3$. In contrast, RBN achieves a substantially more balanced error distribution and the lowest overall RMSE (0.2072), without catastrophic failure on individual regions. It performs particularly well on challenging regions such as $\mathcal{R}_1$ and $\mathcal{R}_4$ while remaining competitive across the others. By dynamically recomposing basis functions, RBN adapts to heterogeneous functional topologies while alleviating the cross-region interference or capacity bottlenecks of static baselines.

Quantitative Analysis: High Parameter Efficiency The scaling behaviors across varying parameter budgets reveal RBN's profound advantage in parameter efficiency. While baselines require exponential parameter growth to incrementally reduce overall error, our RBN achieves best accuracy at a fraction of the parameter cost. Crucially, a lightweight RBN achieves an overall RMSE of 0.2249 utilizing merely 0.42M parameters. To merely approach this performance, the KAN architecture requires its largest configuration, consuming over 115M parameters to yield an inferior RMSE of 0.2411. This empirical scaling is consistent with the approximation property in Theorem 1 and the large discrete compositional space in Remark 1 that RBN extracts massive expressivity, decoupling representational capacity from static parameter constraints.

Depth Robustness: Stabilizing Deep Recursive Synthesis Beyond parameter efficiency, the dataset reveals a critical divergence in depth scaling stability. For conditional computation and recursive models, increasing functional depth triggers severe optimization collapse. For instance, increasing depth from $L = 4$ to $L = 12$ causes MoE's overall error to explode from 0.2709 to 0.3982, and RNN's error to degrade catastrophically from 0.3035 to 0.5456. KANs also exhibit visible saturation and slight degradation at $L = 12$. Conversely, RBN maintains highly stable expressivity. While peak optimization is achieved at $L = 4$ (RMSE 0.2072), unrolling the RBN to deeper steps ($L = 12$) avoids the catastrophic divergence of recurrent algorithms, maintaining a highly robust error trajectory (~ 0.2279). Because the Learnable Basis Dictionary and Dynamic Composer are shared, unrolling the RBN to deeper steps synthesizes complex transformations without catastrophically inflating the optimization landscape, enabling robust functional synthesis.

3.2 Application Experiment: Self-Supervised Learning in Vision

Visual RBN (ViR) and Implementation. To evaluate RBN's scalability to high-dimensional vision, we instantiate ViR within the Masked Autoencoder (MAE) framework (He et al., 2022), asking whether a single foundation model can serve as an elastic computational primitive instead of training discrete variants at different capacities. ViR replaces ViT's layer stack with a single shared ViR Cell, combining Multi-Head Self-Attention (Dosovitskiy et al., 2021) with an RBN Cell, and is trained from scratch on ImageNet-100 (Tian et al., 2020). During pre-training, we adopt stochastic depth sampling, $L_{\text{train}} \sim \mathcal{U}(L_{\min}, L_{\max})$, to learn depth-agnostic compositional rules and enable elastic inference at $L \in \{1, 2, 4, 8\}$. ViR uses Top-1 activation, following MoEs (Fedus et al., 2022), to reduce computation. We evaluate downstream transferability by linear probing on IN-100, SUN-

Networks	FLOPs (G)	IN-100		SUN-397		CIFAR-100		Food-101		AVG
		Acc_1	Acc_5	Acc_1	Acc_5	Acc_1	Acc_5	Acc_1	Acc_5	%
ViT-tiny	1.08	20.16	46.20	8.31	22.93	22.72	49.10	16.72	38.82	28.12
ViT-base (blk×1)	1.51	10.76	30.32	3.02	10.25	15.52	38.20	8.16	23.29	17.44
ViT-base (blk×2)	2.91	21.54	48.26	9.36	24.46	23.13	50.26	15.97	37.84	28.85
ViT-base (blk all)	16.87	45.14	75.18	23.77	47.60	42.25	72.97	38.72	66.19	51.48
ViT-large	59.70	58.10	83.76	30.85	56.64	50.05	79.12	47.28	73.77	59.95
Our ViR ($L = 1$)	2.07	26.88	53.84	13.54	31.67	32.63	61.25	19.45	42.11	**35.17**
Our ViR ($L = 2$)	4.02	34.48	63.22	21.31	44.06	42.43	70.66	27.52	53.43	**44.64**
Our ViR ($L = 4$)	7.93	41.80	70.82	25.22	49.77	45.91	74.43	32.04	58.58	49.82
Our ViR ($L = 8$)	15.74	47.54	76.10	29.80	56.19	48.19	76.13	38.74	65.87	**54.82**

Table 1: **Linear Probing Evaluation.** Top-1 and Top-5 accuracy across four datasets for models pre-trained on IN-100 via MAE. The same pre-trained ViR is elastically evaluated at different inference depths ($L \in \{1, 2, 4, 8\}$), demonstrating representation scaling across varying FLOP budgets. ViT variants are reported as FLOP reference points.blk means blocks.

397 (Xiao et al., 2010), CIFAR-100 (Krizhevsky et al., 2009), and Food-101 (Bossard et al., 2014), using ViT variants as FLOPs reference points. All ViR inference depths share the same pre-trained parameter reservoir. More details are provided in App.D.

Results and Analysis. Tab.1 validates ViR's capacity for elastic visual representation, yielding two observations: **1)** *Elastic Computation via Dynamic Unrolling:* Unlike ViTs with rigid layers, our ViR scales its representation at downstream tasks. Truncating a ViT-base to lower FLOPs budgets (1 or 2 blocks)substantially degrades its downstream representation quality (17.44% and 28.85% average accuracy). As the unrolling depth L of ViR increases from 1 to 8, the accuracy improves robustly. Even at $L = 1, 2$, ViR retains complete functional features (35.17% and 44.64%), showing effective functional synthesis without depth stacking. **2)** *Favorable Compute-to-Accuracy Trade-off:* Using ViTs as FLOPs references, ViR shows a superior compute-accuracy curve. At $L = 4$ (7.93G FLOPs), ViR operates at less than half the FLOPs of ViT-base (16.87G) while achieving a competitive 49.82% average accuracy. When unrolled to $L = 8$ (15.74G FLOPs), ViR reaches 54.82%, outperforming the FLOPs-comparable ViT-base (51.48%), with notable margins (e.g., 48.19% vs. 42.25% on CIFAR-100). These results demonstrate a favorable representation-to-FLOP trade-off for dynamic basis composition relative to the ViT references.

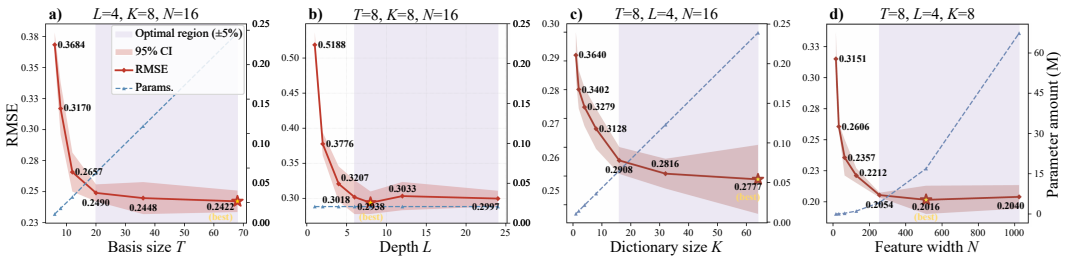


Figure 5: **Ablation of Hyper-parameters.** The panels show the impact of scaling **a)** basis size T, **b)** recursive depth L, **c)** dictionary size K, and **d)** feature width N on the Octagon Benchmark. Red line is the mean RMSE (left axis) over five runs with a 95% confidence interval (CI), blue line is the parameter count (right axis), and the purple shaded region is the optimal region (5% best RMSE).

3.3 Ablation Study and Analysis

To validate the foundational claims of *dynamic functional synthesis*, we dissect the scaling behaviors of RBN's compositional space and depth. More ablations and analysis are in App.E.

Hyper-Parameter Ablation To validate the complexity-capacity trade-offs of RBN, we analyze its hyper-parameters on the Octagon Benchmark over 5 runs (Fig.5). **a)** Increasing the basis size T reduces approximation error, consistent with the truncation term $C_1\mathcal{E}_{\text{basis}}(T)$, with RMSE dropping from 0.3684 at $T = 6$ to 0.2490 at $T = 20$ before reaching diminishing returns around $T \in [20, 68]$. **b)** Increasing the unrolling depth L reduces RMSE from 0.5188 at $L = 1$ to 0.2938 at $L = 8$ while keeping the parameter budget unchanged, demonstrating stable recursive optimization. **c)** Increasing the dictionary size K improves coefficient-space coverage and reduces approximation error from 0.3640 at $K = 1$ to 0.2777 at $K = 64$, although $K > 16$ yields diminishing returns with

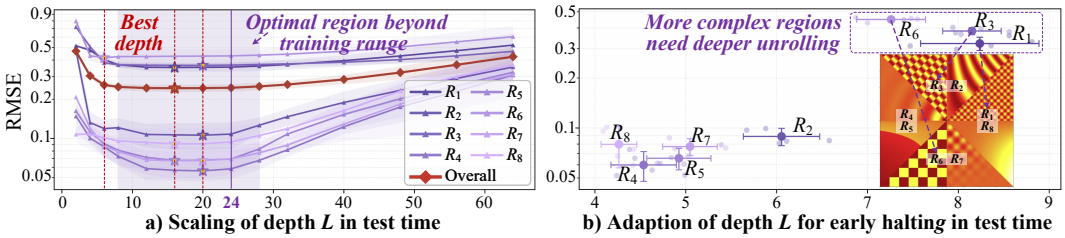


Figure 6: **Analysis of Test-time Depth Scaling and Adaptation. a)** RBN generalizes beyond its training-depth range under test-time unrolling, while excessive depth eventually degrades performance. **b)** An ACT-style halting mechanism (Dehghani et al., 2019; Graves, 2016) enables RBN to automatically allocate different inference depths across functional regions, with more challenging regions receiving deeper computation. Results report the mean and 95% CI over five runs.

higher parameter cost and variance. **d)** Increasing the feature width N improves functional capacity but quadratically inflates the parameter budget. These trends show that RBN balances basis span, dictionary capacity, and depth to effectively exploit its compositional space.

Analysis of Test-time Depth Scaling. An advantage of decoupling functional depth from the parameter budget is that the same trained RBN can be elastically unrolled to different depths at test time without additional parameters. We train RBN with stochastic depth sampling, $L_{\text{train}} \sim \mathcal{U}(2, 24)$, exposing the shared RBN cell to varying recursive depths during training, and vary the unrolling depth at inference (Fig.6a). We evaluate depths beyond the training range to test the expandability of the learned composition. RBN maintains stable performance under depth extrapolation, showing that its functional depth is not restricted to the depths observed in training. However, the benefit of additional computation eventually saturates, and excessively deep unrolling leads to gradual performance degradation. Thus, parameter-free depth scaling makes test-time depth a flexible inference variable, but does not imply that increasing depth monotonically improves performance. This observation motivates adapting the depth to the computational requirements of individual inputs.

Analysis of Test-time Depth Adaptation. The depth-scaling results reveal that different functional regions favor different amounts of recursive computation, motivating adaptive inference depth. Therefore, we augment RBN with an ACT-style halting mechanism (Graves, 2016; Dehghani et al., 2019) (App.C.3), which determines at each recursive step whether computation should continue, subject to a maximum depth of $L_{\max} = 24$. The halting mechanism is jointly optimized with RBN using the standard ACT ponder loss, encouraging a balance between prediction quality and computational depth. Importantly, it does not alter the shared RBN cell or introduce step-specific transformation parameters, but only determines how many recursive steps are executed for each input. As shown in Fig.6b, the learned halting depths vary systematically across the Octagon regions: challenging regions such as $\mathcal{R}_1$, $\mathcal{R}_3$, and $\mathcal{R}_6$ receive deeper computation, whereas simpler regions terminate substantially earlier. These results show that RBN can further convert functional depth from a manually specified inference variable into an input-adaptive execution variable.

4 Conclusion

This paper proposes **Recursive Basis Networks (RBNs)**, which decouple functional depth from parameter depth by replacing layer-wise parameter stacking with recursive functional synthesis over a finite shared basis dictionary. Through input-conditioned composition, RBNs dynamically synthesize heterogeneous functional transformations across recursive steps, allowing functional depth to expand without instantiating additional transformation parameters. Across heterogeneous function approximation and high-dimensional visual representation learning, RBNs demonstrate strong parameter efficiency, competitive representation quality, and flexible inference-time depth scaling. More fundamentally, RBNs shift neural scaling from accumulating depth-specific parameters toward repeatedly recomposing a finite parameter reservoir, providing a new architectural route to derive growing functional complexity from finite primitives, an instantiation of the principle of "infinite from finite" (Humboldt, 1988). This flexibility trades parameter growth for recursive computation: deeper unrolling incurs additional computational cost and does not necessarily yield monotonic performance gains, motivating more efficient adaptive halting and sparse functional composition.

AI use statement In this work, we used generative AI tools to optimize the writing and have checked the content manually. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pp. 446–461. Springer, 2014.

John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. Towards understanding the mixture-of-experts layer in deep learning. *Advances in neural information processing systems*, 35:23049–23062, 2022.

Noam Chomsky. *Syntactic structures*. Walter de Gruyter, 2002.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyzdRiR9Y7`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44 (6):2325–2383, 2002.

David Ha, Andrew M Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Marc D Hauser, Noam Chomsky, and W Tecumseh Fitch. The faculty of language: what is it, who has it, and how did it evolve? *science*, 298(5598):1569–1579, 2002.

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

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Wilhelm Humboldt. On language: The diversity of human language-structure and its influence on the mental development of mankind. 1988.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Translations American Mathematical Society*, 2(28):55–59, 1963.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Hou, and Max Tegmark. Kan: Kolmogorov–arnold networks. In *International conference on learning representations*, volume 2025, pp. 70367–70413, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Larry Schumaker. *Spline functions: basic theory*. Cambridge university press, 2007.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica d: Nonlinear phenomena*, 404:132306, 2020.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pp. 776–794. Springer, 2020.

Richard Van Noorden and Jeffrey M Perkel. Ai and science: what 1,600 researchers think. *Nature*, 621(7980):672–675, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010.

ORGANIZATION OF THE APPENDIX

To provide a comprehensive supplement to the main text, this appendix is organized to guide the reader through theoretical foundations, reproducibility details, and extended empirical analyses:

- ◇ **Appendix A: Extended Discussion of Related Works**
  Provides a deeper contextualization of the proposed RBNs against existing architectural paradigms.
- ◇ **Appendix B: Theoretical Proofs and Complexity Analysis**
  Contains the formal proof for Theorem 1 and a detailed discussion of the computational complexity and capacity trade-offs.
- ◇ **Appendix C: Complementary Strategies for Elastic Computation**
  Details the training- and inference-time strategies that complement RBN, including stochastic depth sampling, sparse functional composition, and adaptive halting.
- ◇ **Appendix D: Detailed Architecture and Experimental Setups**
  Outlines the complete configurations, including the Octagon Benchmark formulations, hyperparameters, and ViR architectures, improving reproducibility.
- ◇ **Appendix E: Additional Experiments and Ablations**
  Presents extensive empirical evaluations, including further functional approximation tasks, PDEBench results, and structural ablation studies.
- ◇ **Appendix F: Interpretability of RBNs**
  Offers a deep dive into the "white-box" nature of RBNs, visualizing the dynamic functional synthesis and progressive depth morphing.
- ◇ **Appendix G: Limitations and Future Directions**
  Discusses current limitations and directions for complementary technologies and potential architectural variants.

A EXTENDED DISCUSSION OF RELATED WORKS

We discuss the paradigms surrounding our proposed RBNs. We position RBNs against the foundational baseline of dense networks and at the intersection of three key architectural paradigms (Fig.7 and Fig.8): basis-based models (Liu et al., 2025; Park & Sandberg, 1991), conditional-computation models (Ha et al., 2017; Dai et al., 2024; Fedus et al., 2022), and recursive models (Sherstinsky,

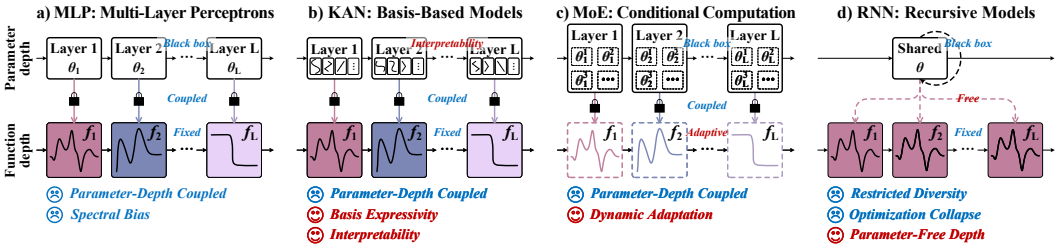


Figure 7: **Comparison of Architectural Paradigms.** The diagrams illustrate how different models map their parameter budget (top row) to functional transformation steps (bottom row). The padlock icon denotes an architectural coupling where scaling functional depth necessitates instantiating additional parameter layers. **a) MLPs** enforce a rigid mapping with static weights and suffer from spectral bias. **b) KANs** construct learnable basis functions that achieve high expressivity and interpretability, yet their capacity remains structurally bottlenecked by layer-wise parameter instantiation. **c) MoEs** unlock dynamic functional adaptation (indicated by dashed boxes) but still require massive layer-specific parameter inflation and risk routing collapse. **d) RNNs** decouple parameter depth from functional depth via weight sharing ("Free"), but apply structurally homogeneous transformations across steps, which restricts representation diversity and triggers optimization collapse.

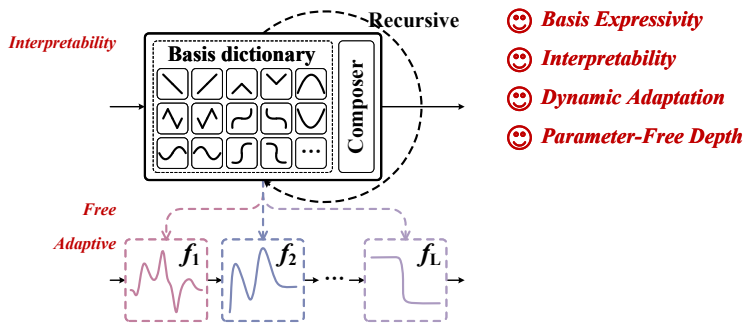


Figure 8: (Continued from the Fig.7) **RBN Architectural Paradigm.** Our RBN is structurally designed to unify the disjoint advantages shown in Fig.7, including basis expressivity, dynamic adaptation, and parameter-free depth, while eliminating their intrinsic bottlenecks (blue indicators).

2020; Dehghani et al., 2019). RBN is the first paradigm to simultaneously achieve basis expressivity, input-conditioned adaptability, and parameter-free depth scaling within a unified framework.

Multi-Layer Perceptrons and Static Parameterization MLPs form the backbone of modern representation learning, acting as the ubiquitous feed-forward mechanism in architectures like Transformers (Vaswani et al., 2017). While their universal approximation capabilities are well-established (Cybenko, 1989), MLPs rely on a rigid architectural scaling law: expanding functional capacity dictates the sequential instantiation of independent, static parameter layers. This coupling inherently binds the model's functional depth to its total parameter budget. Furthermore, dense MLPs are susceptible to spectral bias (Rahaman et al., 2019), requiring massive parameter over-provisioning to capture variations or complex topological shifts which are non-linear distribution. Our RBN departs from this one-to-one coupled parameterization paradigm. It abandons the static stacking of discrete layers in favor of dynamic functional synthesis. By extracting representations from a basis dictionary, RBN enables basis expressivity with powerful non-linear fitting ability, while rendering the functional depth independent of its parameter budget.

Basis-Based Models for Functional Approximation The foundation of expressing complex multivariate functions via univariate bases stems from the Kolmogorov-Arnold representation theorem (Kolmogorov, 1963). Historically, Radial Basis Function (RBF) networks (Park & Sandberg, 1991) and spline-based models (Schumaker, 2007) capitalized on this to achieve functional approximation. Recently, KANs (Liu et al., 2025) revitalized this paradigm by replacing fixed linear weights with learnable B-splines, demonstrating superior parameter efficiency and interpretability over MLPs. However, KANs remain structurally conventional: they stack discrete layers of basis functions, where each layer requires its own instantiated parameter set, bottlenecking representation capacity by instantiated depth. Instead of assigning static bases to each layer, our RBN maintains a global Learnable Basis Dictionary. By dynamically composing these shared bases, RBN retains the non-linear expressivity of KANs while decoupling capacity from layer count.

Conditional-Computation Models for Dynamic Adaptation Adjusting network parameters or pathways in response to input context has been explored through dynamic routing and meta-learning (Bengio et al., 2013). Historically, early conditional execution algorithms like HyperNetworks (Ha et al., 2017) capitalized on this to achieve input-dependent functional adaptation. Recently, MoEs (Dai et al., 2024; Fedus et al., 2022; Chen et al., 2022; Cai et al., 2025) revitalized this by routing inputs to a sparse subset of specialized modules, demonstrating superior capacity scaling over dense networks. However, MoEs route representations to discrete and statically instantiated experts, requiring a dedicated parameter set per layer to scale functional capacity. Instead of routing to static experts (Fedus et al., 2022) or generating full weight matrices (Ha et al., 2017), our RBN synthesizes functional transformations via dynamic composition of shared basis profiles, retaining the adaptability of conditional models while eliminating the need for layer-wise parameter inflation.

Recursive Models for Weight-Shared Scaling The desire to decouple functional depth from parameter depth has long motivated weight-sharing architectures. RNNs (Sherstinsky, 2020) unroll a shared transformation over temporal or depth sequences. More recently, Universal Transformers (Dehghani et al., 2019) and Deep Equilibrium Models (DEQs) (Bai et al., 2019) tie weights across layers to support iterative computation. However, these approaches reuse the same transformation

parameters across recursive steps, which can constrain step-specific functional specialization and introduce optimization difficulties in deep recursive computation, including vanishing or exploding gradients (Pascanu et al., 2013). In contrast, RBN retains a shared parameter reservoir while explicitly synthesizing step-specific functional coefficients through input-conditioned composition over the shared basis dictionary. This enables heterogeneous functional transformations across recursive steps without introducing layer-specific parameters.

B Theoretical Proofs and Complexity Analysis

B.1 Proof of Theorem 1

This section provides the formal proof for Theorem 1. The core objective is to demonstrate that an RBN with a shared basis dictionary can approximate deep functional compositions while fully decoupling the parameter budget from the functional depth L. We decompose the total approximation error into two components using the triangle inequality. Let F^* denote the ideal compositional function where the intermediate coefficients $c^*_{l,t}$ are exactly specified (unconstrained by the dictionary). The total error is bounded by:

$$\|F - F^{T,K}_{\mathrm{RBN}}\|_\infty \leq \underbrace{\|F - F^*\|_\infty}_{\text{Basis Truncation Error}} + \underbrace{\|F^* - F^{T,K}_{\mathrm{RBN}}\|_\infty}_{\text{Dictionary Composition Error}} . \tag{7}$$

Bounding the Basis Truncation Error The first term, $\|F - F^*\|_\infty$, measures the capacity of the chosen basis set to span the target function space assuming optimal coefficients. Classical approximation theory establishes that finite basis expansions progressively reduce approximation error as the basis resolution increases, with the convergence rate determined by the regularity of the target function and the approximation properties of the basis family (Pinkus, 1999; Schumaker, 2007). Accordingly, for the basis approximation considered in Theorem 1, we define the maximum basis approximation error $\mathcal{E}_{\mathrm{basis}}(T)$ as:

$$\mathcal{E}_{\mathrm{basis}}(T) = \sup_{1\leq l\leq L} \inf_{\mathbf{c}_l\in\mathbb{R}^T} \|f_l - \textstyle\sum_{t=1}^T c_{l,t}\phi_t\|_\infty. \tag{8}$$

This term measures the worst-case approximation error of the finite basis expansion across the required functional transformations. The resulting basis truncation error is bounded as

$$\|F - F^*\|_\infty \leq C_1 \mathcal{E}_{\mathrm{basis}}(T), \tag{9}$$

for some constant $C_1 > 0$ depending on the target function, basis functions, and composition stability. This bound isolates the approximation error associated with the finite basis expansion from the error introduced by dictionary composition.

Bounding the Dictionary Composition Error The second term, $\|F^* - F^{T,K}_{\mathrm{RBN}}\|_\infty$, bounds the error from our functional synthesis in the dictionary. Instead of allowing arbitrary coefficients $c^*_{l,t}$, RBN constrains the coefficients to be a combination of profiles drawn from a shared dictionary $\mathbf{W} \in \mathbb{R}^{K\times T}$. Specifically, for any step l and basis t, the synthesized coefficient is $c_{l,t} = \mathbf{w}_t^\top \mathbf{s}_t^l$, where $\mathbf{s}_t^l \in \mathbb{R}^K$denotes the normalized compositional weights produced by the composer. Since the softmax normalization ensures $s_k \geq 0$ and $\sum_{k=1}^K s_k = 1$, we define the maximum composition error $\mathcal{E}_{\mathrm{com}}(K)$ as:

$$\mathcal{E}_{\mathrm{com}}(K) = \sup_{1\leq l\leq L, 1\leq t\leq T} \inf_{\mathbf{s}\in\mathbb{R}^K} |c^*_{l,t} - \mathbf{w}_t^\top \mathbf{s}|. \tag{10}$$

This term represents the worst-case dictionary composition error when approximating the ideal coefficients within the convex hull of the candidate dictionary profiles, analogous to quantization error in signal processing (Gray & Neuhoff, 2002). Assuming a Lipschitz continuous dependence of F^* on its intermediate coefficients, the total composition error is proportional to this maximum error for some Lipschitz constant $C_2 > 0$ associated with the considered compositional mapping F^*:

$$\|F^* - F^{T,K}_{\mathrm{RBN}}\|_\infty \leq C_2 \mathcal{E}_{\mathrm{com}}(K). \tag{11}$$

Relation to Compositional Space The approximation bound above is independent of the compositional space in Remark 1. Under top-1 composition, the T basis functions independently select from K candidate coefficient profiles, yielding up to K^T discrete joint routing configurations. Therefore,

when $K^T \gg L$, the available routing configurations are substantially more numerous than the L transformation steps required by the recursive composition. This compositional space provides diversity for step-specific functional synthesis, whereas $\mathcal{E}_{\text{com}}(K)$ measures how accurately the shared dictionary can realize the coefficients required by the target transformations. In particular, the K^T counting argument does not itself determine $\mathcal{E}_{\text{com}}(K)$.

Parameter Complexity Analysis Finally, we evaluate the parameter budget required to achieve this bound. The total parameter count is determined by three shared components:

- *Dynamic Composer:* The learnable projection matrix $\pi \in \mathbb{R}^{N \times KT}$.
- *Learnable Basis Dictionary:* $\mathbf{W} \in \mathbb{R}^{K \times T}$.
- *Basis Functions:* The parameters defining $\{\phi_t\}_{t=1}^{T}$.

The dominant term scales as $\mathcal{O}(N^2TK)$. None of these components are indexed by the depth variable l. Because the parameters are shared across the recursive depth unrolling (Eq.6), the parameter complexity remains bounded by $\mathcal{O}(N^2TK)$ regardless of how large L becomes. This completes the proof that RBN achieves functional depth scaling decoupled from parameter expansion.

B.2 Computational Complexity and Capacity Trade-off

In this section, we provide a formal analysis of the parameter and computational complexity of our RBN. While RBN drastically compresses the parameter budget, it incurs a higher computational cost (FLOPs) per functional transformation step compared to linear projections. This reflects a deliberate architectural trade-off: shifting the expressivity bottleneck from static memory footprint to dynamic computational cycles. Tab.2 summarizes the asymptotic complexities for an input/output dimension N, depth L, basis size T, and dictionary size K (for MoE, E denotes the number of experts, and we assume that the expert is realized by MLP).

Architecture	Parameter Budget	Computational Cost (FLOPs)	Compositional Space
MLP (LeCun et al., 2015)	$\mathcal{O}(N^2 \cdot L)$	$\mathcal{O}(N^2 \cdot L)$	$\mathcal{O}(L)$
RNN (Sherstinsky, 2020)	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2 \cdot L)$	$\mathcal{O}(1)$
MoE (Top-1) (Fedus et al., 2022)	$\mathcal{O}(N^2 \cdot E \cdot L)$	$\mathcal{O}(N^2 \cdot L)$	$\mathcal{O}(E^L)$
KAN (Liu et al., 2025)	$\mathcal{O}(N^2 \cdot T \cdot L)$	$\mathcal{O}(N^2 \cdot T \cdot L)$	$\mathcal{O}(L)$
RBN (*top-K*)	$\mathcal{O}(N^2 \cdot T \cdot K)$	$\mathcal{O}(N^2 \cdot T \cdot K \cdot L)$	$\mathcal{O}(K^T)$
RBN (*top-1)**	$\mathcal{O}(N^2 \cdot T \cdot K)$	$\mathcal{O}(N^2 \cdot T \cdot L)$	$\mathcal{O}(K^T)$

Table 2: Theoretical comparison of complexities. RBN trades per-step computational FLOPs for parameter efficiency and an exponentially large compositional space independent of layer depth. * The RBN (top-1) is a variant that utilizes a top-k activation as MoEs compared in App.E.4 that will effectively reduce computation.

Parameter vs. Compute Decoupling. As shown in Tab.2, traditional architectures (MLP, KAN) couple their parameter budget to depth L. MoE decouples capacity from compute, but its parameter budget still scales linearly with depth and expert count E. In stark contrast, RBN's parameter budget $\mathcal{O}(N^2TK)$ is strictly independent of L. The computational cost scales as $\mathcal{O}(N^2TKL)$ due to the dynamic synthesis of T basis functions from K dictionary profiles at each step.

Exponential Return on Computational Investment. While the per-step FLOPs in RBN are amplified by a factor of TK relative to a standard MLP, this computational investment yields a combinatorial expansion up to K^T discrete routing configurations under Top-1 composition. Unlike RNNs, which apply a static monolithic transformation across steps ($\mathcal{O}(1)$ compositional space), RBN utilizes computation to navigate a vast, dynamic compositional space using a tiny parameter budget. Furthermore, if we utilize the top-k activation as MoEs (Fedus et al., 2022), i.e., RBN (top-1), the computation will be further reduced while maintaining the compositional space. Experiment is presented in App.E.4.

Practical Hardware Efficiency. We benchmark a parameter-matched Top-1 ViR and ViT-base under identical hardware to evaluate the wall-clock efficiency of the current RBN implementation. Evaluation settings use two NVIDIA H100 GPUs (80GB), BF16 precision, 224×224 inputs, and a

Model	Param. (M)	FLOPs (G)	Throughput (img/s)	Latency (ms/batch)
ViT-base	85.88	16.87	20,010	12.79
Parameter-matched top-1 ViR ($L = 8$)	86.32	14.86	7,128	35.91

Table 3: **Wall-clock efficiency under matched hardware settings.** Encoder-only measurements are conducted on two NVIDIA H100 80GB GPUs using BF16 precision, 224×224 inputs, and a global batch size of 256. Latency denotes the wall-clock time per global batch.

global batch size of 256. As shown in Tab.3, ViT-base achieves 20,010 images/s with 12.79 ms batch latency, whereas ViR at $L = 8$ achieves 7,128 images/s with 35.91 ms latency, despite requiring fewer theoretical FLOPs (14.86G vs. 16.87G). Under the same 200-epoch MAE pre-training recipe, ViR requires 14.35 hours compared with 9.77 hours for ViT-base. These measurements show that the current implementation of dynamic functional synthesis is less hardware-efficient than the highly optimized dense operations in ViT, despite its favorable theoretical FLOP count. We have discussed this in the limitations (App.G).

C Complementary Strategies for Elastic Computation

The decoupling of functional depth from parameter depth makes computation in RBN naturally adjustable without changing the shared parameterization. Several complementary strategies further turn this structural flexibility into **elastic computation**: stochastic depth sampling allows a single model to operate effectively across different inference depths, top-k activation adjusts the computational cost of functional composition within each recursive step, and adaptive halting allocates different numbers of recursive steps to different inputs. These strategies vary computation across training depth, per-step composition, and input-dependent execution while preserving the same shared RBN parameter reservoir. These strategies are not required components of RBN, but optional mechanisms explored in our experiments: stochastic depth sampling is used in the test-time depth scaling analysis (Sec.3.3) and ViR pre-training (Sec.3.2), top-k activation is adopted in ViR (Sec.3.2) and further examined through the sparse-composition ablation (App.E.4), and adaptive halting is evaluated in the test-time depth adaptation analysis (Sec.3.3).

C.1 Stochastic Depth Sampling

RBN exposes the recursive depth L as a variable without additional transformation parameters. To train a shared RBN cell that remains effective across different recursive steps, we designed a stochastic depth sampling method, where a single depth is sampled for each mini-batch, and the shared cell is unrolled for the corresponding number of steps. This strategy allows the same parameter reservoir to be optimized under multiple functional depths, thus optimizing the representation in different unrolled depths. It has two sampling strategies in experiments.

Uniform Depth Sampling. It samples the unrolling depth from a discrete uniform distribution:

$$L_{\text{train}} \sim \mathcal{U}(L_{\min}, L_{\max}), \tag{12}$$

where $\mathcal{U}(a, b)$ denotes the discrete uniform distribution over the integer set $\{a, a+1, \ldots, b\}$. Thus, every admissible recursive depth is sampled with equal probability. We use this strategy in the test-time depth scaling analyses (Sec.3.3), where we expose RBN uniformly to a prescribed depth range to examine its behavior when evaluated at different L_{test}, including depths outside the training range.

Curriculum Depth Sampling. It smoothly expands the depth sampling interval as training progresses via a curriculum mechanism. For each batch at epoch e, the unrolling depth is uniformly sampled from a dynamically bounded interval:

$$L_{\text{train}} \sim \mathcal{U}(L_{\text{min}}, L_{\text{up}}^{(e)}). \tag{13}$$

The dynamic upper bound $L_{\text{up}}^{(e)}$ grows linearly from minimum depth L_{min} to the absolute maximum depth L_{max} over the E_{curr} curriculum epochs:

$$L_{\text{up}}^{(e)} = \lfloor L_{\text{min}} + (L_{\text{max}} - L_{\text{min}}) \cdot \min(1, \tfrac{e}{E_{\text{curr}}}) \rfloor. \tag{14}$$

This curriculum mechanism forces the Dynamic Composer to first master shallow, macroscopic functional mappings before progressively tackling deep, complex residual refinements. Ultimately, it enables the model to learn step-agnostic compositional rules, allowing the single pre-trained foundation model to be elastically unrolled to varying inference depths. We adopt this curriculum strategy for ViR pre-training (Sec.3.2) to improve the learned scaling ability.

C.2 TOP-k ACTIVATION

RBN is able to reduce the computational cost of dynamic composition through top-k activation, analogous to sparse routing in MoEs (Fedus et al., 2022). For each basis function t, the Dynamic Composer predicts a compositional state $\mathbf{s}_t^l \in \mathbb{R}^K$ over the K candidate coefficient profiles at recursive step l. Top-k activation retains the k profiles with the highest scores and normalizes them over the selected set. The coefficient of the t-th basis function at recursive step l is then synthesized as

$$\hat{s}_{j,t}^l = \begin{cases} \frac{s_{j,t}^l}{\sum_{m \in \mathcal{I}_{t,k}^l} s_{m,t}^l}, & j \in \mathcal{I}_{t,k}^l, \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where $\mathcal{I}_{t,k}^l$ is the indices of the top-k profiles. Therefore, the synthesized coefficient is $c_{l,t} = \mathbf{w}_t^\top \hat{\mathbf{s}}_t^l$.

The sparsification is independently applied to the dictionary profiles of each basis function and does not alter the basis set or recursive unrolling mechanism. Setting $k = K$ recovers dense functional composition, while $k = 1$ yields top-1 composition. Accordingly, the dictionary-composition cost scales with the number of active profiles k rather than the full dictionary size K. We analyze the computation-accuracy trade-off across values of k in the additional ablation experiments (App.E.4).

C.3 ADAPTIVE HALTING

To make functional depth input-adaptive, an ACT-style halting mechanism (Graves, 2016; Dehghani et al., 2019) can be integrated into the recursive unrolling process of our RBNs. At each recursive step l, a lightweight halting head takes the current representation $\mathbf{h}^l$ and predicts a halting probability

$$p_l = \sigma_{\text{sigmoid}}(g(\mathbf{h}^l)), \tag{16}$$

where $g(\cdot)$ denotes the learnable halting head and $\sigma_{\text{sigmoid}}(\cdot)$ is the sigmoid function. Following ACT, recursive computation terminates once the accumulated halting probability reaches the prescribed threshold, or when the maximum depth $L_{\max}$ is reached. The halting module is jointly optimized with RBN using the standard ACT ponder loss,

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{ponder}} \mathcal{L}_{\text{ponder}}, \tag{17}$$

which balances task performance against the amount of recursive computation. Importantly, adaptive halting does not modify the shared RBN cell or introduce step-specific transformation parameters; it only determines how many functional transformation steps are executed for each input. Therefore, RBN converts functional depth from a manually controlled inference variable into an input-adaptive execution variable. In our test-time depth adaptation analysis, we set $L_{\max} = 24$ and $\lambda_{\text{ponder}} = 0.001$ (Sec.3.3).

D DETAILED ARCHITECTURE AND EXPERIMENTAL SETUPS

In this section, we provide comprehensive formulations, architectural specifications, and hyperparameter configurations to ensure the full reproducibility of our empirical studies.

D.1 FORMULATIONS OF THE OCTAGON BENCHMARK

As shown in Fig.3a, the Octagon benchmark defines a spatially heterogeneous target function $F : [-1, 1]^2 \to \mathbb{R}$ over eight disjoint angular regions partitioned via polar coordinates (r, θ), where

$r = \sqrt{x_1^2 + x_2^2}$ and $\theta = \text{atan2}(x_2, x_1) \in [-\pi, \pi]$:

$$F(x_1, x_2) = \begin{cases} \sin(20\pi x_1)\cos(20\pi x_2), & \theta \in [0, \frac{\pi}{4}) \quad (\mathcal{R}_1) \\ \sin(12\pi x_1 x_2), & \theta \in [\frac{\pi}{4}, \frac{\pi}{2}) \quad (\mathcal{R}_2) \\ \sin(16\pi u)\,\text{sgn}(\cos(10\pi v)), & \theta \in [\frac{\pi}{2}, \frac{3\pi}{4}) \quad (\mathcal{R}_3) \\ 2e^{-4r}\cos(6\pi r), & \theta \in [\frac{3\pi}{4}, \pi] \quad (\mathcal{R}_4) \\ \text{sgn}(x_1 x_2 - 0.15)|x_1 x_2|^{0.4}, & \theta \in [-\pi, -\frac{3\pi}{4}) \quad (\mathcal{R}_5) \\ (-1)^{\lfloor 7x_1 \rfloor + \lfloor 7x_2 \rfloor}, & \theta \in [-\frac{3\pi}{4}, -\frac{\pi}{2}) \quad (\mathcal{R}_6) \\ e^{-3r^2}\sin(10\pi x_1)\cos(10\pi x_2), & \theta \in [-\frac{\pi}{2}, -\frac{\pi}{4}) \quad (\mathcal{R}_7) \\ 3x_1^2 x_2 - x_2^3, & \theta \in [-\frac{\pi}{4}, 0) \quad (\mathcal{R}_8) \end{cases} \tag{18}$$

where $u = \frac{\sqrt{2}}{2}(x_1 + x_2)$ and $v = \frac{\sqrt{2}}{2}(x_1 - x_2)$ in $\mathcal{R}_3$ denote coordinates under a 45° rotation. Each region is designed to expose a distinct approximation-theoretic challenge:

- $\mathcal{R}_1$: Axis-aligned high-frequency oscillations (20π), targeting the spectral bias of smooth function approximators that struggle to represent rapidly varying patterns along coordinate directions.
- $\mathcal{R}_2$: Non-separable multiplicative interaction $\sin(12\pi x_1 x_2)$, which cannot be decomposed into a sum $f(x_1) + g(x_2)$, challenging architectures that rely on additive univariate decompositions.
- $\mathcal{R}_3$: Rotated high-frequency oscillations combined with sign discontinuities, compounding the spectral bias challenge with non-axis-aligned structure and hard boundaries.
- $\mathcal{R}_4$: Radially decaying oscillations $2e^{-4r}\cos(6\pi r)$, requiring simultaneous modeling of a smooth global envelope and local high-frequency variation as a function of radius.
- $\mathcal{R}_5$: Hyperbolic discontinuity combined with a fractional power law $|x_1 x_2|^{0.4}$, introducing a sign boundary defined by the hyperbola $x_1 x_2 = 0.15$ alongside non-integer exponent nonlinearity.
- $\mathcal{R}_6$: High-frequency checkerboard pattern defined by the parity of $\lfloor 7x_1 \rfloor + \lfloor 7x_2 \rfloor$, producing axis-aligned discontinuities at a spatial frequency of 7, which is particularly hostile to smooth basis approximators.
- $\mathcal{R}_7$: Gabor-modulated envelope $e^{-3r^2}\sin(10\pi x_1)\cos(10\pi x_2)$, combining a radially decaying Gaussian with high-frequency sinusoidal modulation, requiring localized nonlinear basis functions to capture the amplitude decay.
- $\mathcal{R}_8$: Pure polynomial cross-term $3x_1^2 x_2 - x_2^3$, which corresponds to the imaginary part of $(x_1 + ix_2)^3$ and contains no separable univariate components.

Together, these eight regions are coupled into a single piecewise function F defined over a compact 2D domain, creating a unified approximation target that is locally heterogeneous in both smoothness and functional form within each region. This design ensures that no single inductive bias or architectural prior can uniformly address all regions simultaneously, making the benchmark a sensitive diagnostic of a model's functional coverage across diverse approximation-theoretic challenges.

D.2 Configurations in Toy Study

We provide additional implementation details for the Octagon benchmark to supplement the experimental settings described in Sec.3.1.

Baseline Architectures, Configurations, and Scaling Sweep To map the parameter expressivity (Fig.4), we evaluate all architectures across functional depths $L \in \{2, 4, 12\}$ and exponentially scale their hidden feature widths (N). This exhaustive grid search ensures that RBN's performance advantages stem from structural superiority rather than insufficient baseline scaling. The specific architectural implementations and capacity alignment strategies are as follows:

- *MLP (LeCun et al., 2015)* represents the standard paradigm of static parameter stacking. Our implementation utilizes dense linear projections with Sigmoid Linear Unit (SiLU) activations.
- *RNN (Sherstinsky, 2020)* represents the conventional weight-sharing recursive model. It unrolls a single shared hidden state transition, $h_t = \text{Tanh}(\mathbf{W}_x\, x_t + \mathbf{W}_h\, h_{t-1})$. To maximize its representation power, we augment the standard RNN with residual connections across steps and inject a learnable depth embedding into the input at each step, matching the contextual awareness of RBN. Despite these enhancements, it applies the exact same monolithic weight matrices at each depth.

- *KAN (Liu et al., 2025)* represents the basis-driven models. Each KAN layer computes the sum of a base linear projection (with SiLU) and a parameterized B-spline transformation. To ensure strict equivalence with RBN's basis capacity, the KAN baseline is instantiated with an identical grid resolution (9 grid points) and spline order ($k = 3$), yielding an equivalent basis size of $T = 12$.
- *MoE (Fedus et al., 2022)* represents the conditional computation models. To mirror the dictionary size ($K = 8$) of our RBN, the MoE baseline is configured with $E = 8$ distinct linear experts per layer. It employs a noisy Top-2 routing mechanism, injecting Gaussian noise into the router logits during training to encourage exploration. To prevent routing collapse, its optimization is augmented with a routing load-balancing auxiliary loss.

Specifically, to generate the comprehensive capacity scaling curves, we independently scale the hidden feature widths N for each architecture. This capacity-alignment strategy ensures that models with fundamentally different parameter densities (e.g., dense MLPs vs. basis-expanded KANs) are evaluated under equivalent parameter budgets and FLOPs, spanning a spectrum from highly compressed to heavily over-parameterized regimes. The detailed optimization settings and scaling strategies are summarized in Tab.4.

Hyperparameter	**Value**
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Peak learning rate	1×10^{-3}
Weight decay	1×10^{-5}
Batch size	2048
Learning rate schedule	Cosine annealing
Training epochs	1800
Loss function	Mean Squared Error (MSE)
Functional depths sweep (L)	$\{2, 4, 12\}$
Feature widths sweep (N)	$\{16, 32, 64, 128, 256, 512, 1024\}$ for RBN
RBN basis size (T)	12 (9 grid points + 3 spline orders)
RBN dictionary size (K)	8

Table 4: **Training Hyper-parameters for the Octagon Benchmark.** Most optimization settings are shared across all architectures to prevent confounding factors, but the hidden feature widths (N) are independently scaled for each model to ensure a fair and rigorous capacity comparison under aligned parameter budgets.

Optimization and Stability. While all models are optimized via Mean Squared Error (MSE), their regularization requirements differ significantly. RBN maintains routing stability naturally through its Dynamic Composer and is optimized purely via MSE. Conversely, as noted above, the MoE baseline suffers from severe routing collapse on this piecewise fitting task and strictly requires the auxiliary load-balancing loss ($\lambda_{\text{aux}} = 0.01$) to remain competitive. Finally, to rigorously test the inherent forward and backward stability of these recursive mechanisms, we do not apply any gradient clipping. This allows all models to optimize freely, exposing the true vanishing or exploding gradient behaviors inherent to their specific depth scaling. All Octagon benchmark experiments were conducted on a single NVIDIA A100 GPU (40GB memory).

D.3 VISUAL RBN ARCHITECTURE

Building upon the RBN architecture described in Sec.2.2, we instantiate **Visual RBN (ViR)** for vision representation learning under the MAE (He et al., 2022) (Fig.9). The key design principle is to replace the conventional stack of L independent Transformer blocks with a single shared ViR Cell that is recursively unrolled for L steps, realizing parameter-free depth scaling in the high-dimensional vision domain.

ViR Cell Each ViR Cell consists of two sub-modules connected via residual streams with pre-norm layer normalization: **1)** a standard Multi-Head Self-Attention (MHSA) module that captures global spatial dependencies among tokenized patches, and **2)** an RBN cell that replaces the original Feed-Forward Network (FFN) in Transformers (Vaswani et al., 2017), synthesizing functional

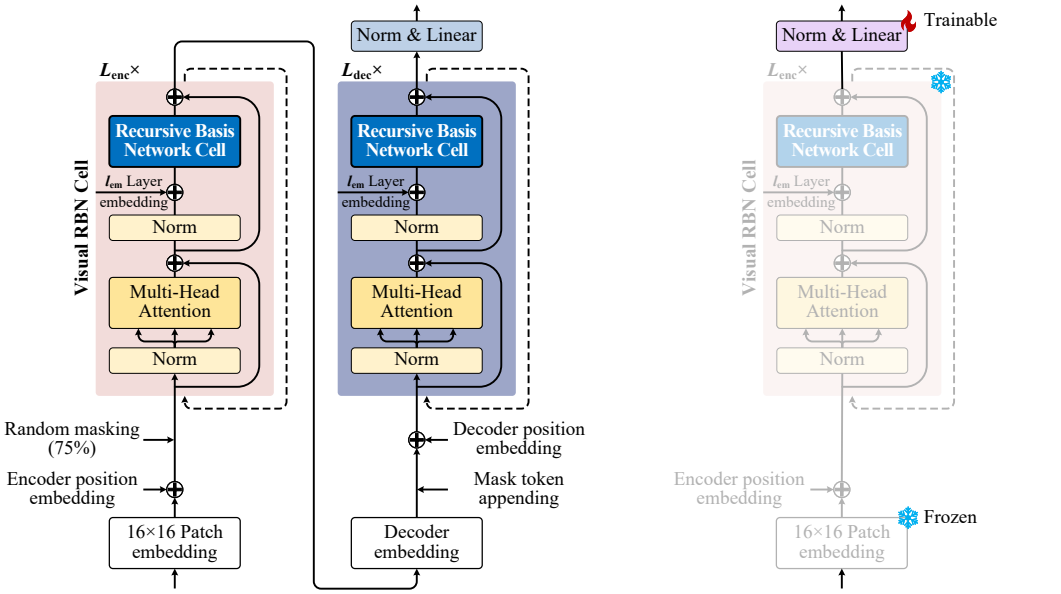


Figure 9: **Visual RBN (ViR) for MAE Pre-training and Linear Probing. a)** The encoder and decoder each consist of a single shared ViR Cell which comprises a Multi-Head Self-Attention and a RBN cell, and the ViR Cells are recursively unrolled for L_{enc} and L_{dec} steps with a layer embedding injected at each step. The L_{enc} is sampled stochastically during training ($L_{\text{enc}} \sim \mathcal{U}(L_{\min}, L_{\max})$) to learn depth scaling ability. **b)** For downstream evaluation, the encoder is frozen and a linear classifier is trained on top of the learned representations at varying inference depths.

transformations via dynamic basis composition over parameterized B-spline bases. For computational efficiency, ViR Cell utilizes a top-1 activation strategy (as in MoEs (Fedus et al., 2022), App.C.2), activating only the most relevant candidate profile per basis. To enable distinct functional transformations at each recursive step under shared parameters, a layer embedding l_{em} is injected into the ViR Cell input at step l, conditioning the Dynamic Composer on the current depth index, consistent with the recursive unrolling formulation in Eq.6. The forward pass at step l is:

$$\begin{aligned} \mathbf{h} &= \mathbf{x} + \text{MHSA}(\text{Norm}(\mathbf{x})), \\ \mathbf{x}' &= \mathbf{h} + \text{RBN}(\text{Norm}(\mathbf{h}) + l_{\text{em}}). \end{aligned} \tag{19}$$

ViR Encoder in MAE Images are divided into 16×16 patches, linearly projected to D-dimensional tokens, and augmented with fixed sinusoidal position embeddings (He et al., 2022). During pre-training, 75% of patch tokens are randomly masked and discarded. The remaining visible tokens together with a `[CLS]` token are processed by the shared ViR Cell unrolled for L_{enc} steps, followed by a layer normalization. To enable elastic computation, the ViR Encoder is trained with the stochastic depth sampling strategy. This forces the ViR Cell to learn depth-agnostic compositional rules rather than overfitting to a specific depth, enabling unrolling for varying depths in downstream tasks.

ViR Decoder in MAE The decoder is an independent lightweight ViR Cell with a smaller embedding dimension and dictionary size, decoupled from the encoder. The encoder output is projected to the decoder dimension, mask tokens are appended, and decoder position embeddings are added. The decoder ViR cell is unrolled for a fixed L_{dec} steps, followed by a linear prediction head that reconstructs the normalized pixel values of masked patches.

Model Details. Tab.5 summarizes the architectural hyper-parameters for the MAE-ViT baselines and our proposed ViR model. The architectural configurations of ViR are designed around two core principles to operationalize the elastic computation paradigm: **1)** *Unified Parameter Reservoir:* Unlike standard scaling laws that mandate instantiating discrete models to match different computational budgets (e.g., Tiny, Base, Large), we deploy a **single foundation model** with a fixed budget of 130.95M parameters. By recursively sharing the ViR Cell, we eliminate the need for layer-wise

Model	N_{enc}	L_{enc}	H	T	K	N_{dec}	L_{dec}	Param. (M)
ViT-tiny	192	12	3	–	–	128	8	5.54
ViT-base	768	12	12	–	–	512	8	85.88
ViT-large	1024	24	16	–	–	512	8	303.40
ViR	768	$L_{\text{enc}} \in [1, 8]$	12	12	18	256	2	130.95

Table 5: ViR and MAE-ViT model configurations. N_{enc}: encoder embedding dimension (feature width), L_{enc}: encoder unrolling steps (depth), H: attention heads, T: basis size, K: dictionary size, N_{dec}: decoder embedding dimension (feature width), L_{dec}: decoder unrolling steps (depth). The ViR model utilizes elastic unrolling ($L_{\text{enc}} \in [1, 8]$) rather than a fixed layer count.

parameter inflation (i.e., stacking independent FFN layers). This allows the network to maintain a high-capacity embedding dimension ($N_{\text{enc}} = 768$) while relying on dynamic functional synthesis, rather than parameter bloat, to handle varying task complexities. **2)** *Elastic Compute via Dynamic Unrolling:* We transition the encoder depth L_{enc} from a rigid architectural constraint into an elastic inference variable ($L_{\text{enc}} \in [1, 8]$) with the curriculum mechanism (Eq.13 with E_{curr}=80, App.C.1). The basis size ($T = 12$) and dictionary size ($K = 18$) are set to enlarge the $\mathcal{O}(K^T)$ compositional space while preserving computational efficiency (FLOPs). Furthermore, consistent with the MAE design philosophy (He et al., 2022), the decoder is intentionally kept lightweight ($N_{\text{dec}} = 256$, fixed shallow unrolling $L_{\text{dec}} = 2$). This architectural asymmetry enables the heavy representational burden to remain on the dynamically unrolled encoder, while the decoder handles pixel reconstruction.

D.4 Configurations in Application Experiment

For reproducibility and a fair comparison, we adhere to the Masked Autoencoder (MAE) (He et al., 2022) training recipe without architecture-specific hyper-parameter tuning. This enables any performance gains to be attributable to our proposed functional synthesis.

Datasets We utilize ImageNet-100 (IN-100) (Tian et al., 2020), a widely adopted 100-class subset of ImageNet-1K comprising approximately $130,000$ training images and $5,000$ validation images, as a computationally efficient proxy for self-supervised pre-training. To assess the quality and transferability of the learned representations, we evaluate the pre-trained models on four distinct image classification benchmarks: **1)** IN-100 for in-domain generalization; **2)** SUN-397 (Xiao et al., 2010) ($19,850$ training and $19,850$ testing images across 397 classes) for cross-domain scene recognition; **3)** CIFAR-100 (Krizhevsky et al., 2009) ($50,000$ training and $10,000$ testing images across 100 classes) to test robustness on low-resolution images; and **4)** Food-101 (Bossard et al., 2014) ($75,750$ training and $25,250$ testing images across 101 classes) for fine-grained visual categorization.

Evaluation Protocols Following self-supervised learning practices, we adopt Linear Probing as our evaluation protocol. After pre-training, the weights of the ViR encoder cell are frozen. We extract the class-token features from the final encoder step at specific inference unrolling depths (e.g., $L \in \{1, 2, 4, 8\}$) and train a single linear classifier from scratch on the downstream training sets. Performance is reported using Top-1 and Top-5 accuracy metrics on the respective validation sets, isolating the quality of the raw representations synthesized at different compute budgets.

Hyperparameter Settings During pre-training, the ViR encoder and decoder cells are unrolled for L_{enc} and L_{dec} steps, respectively, matching the architectural configurations (Tab.5). For each batch, the encoder unrolling depth is uniformly sampled: $L_{\text{enc}} \sim \mathcal{U}(1, 8)$ via the stochastic depth sampling. We adhere to the MAE (He et al., 2022) recipe without architecture-specific tuning. Models are trained from scratch for 200 epochs with a patch size of 16×16 and a 75% mask ratio. Pre-training hyper-parameters are summarized in Tab.6. For downstream linear probing, the linear classifiers are trained using the LARS optimizer with a batch size of 1024, a cosine learning rate decay, and 10 epochs of warmup.

Comparison Setup. To establish a rigorous baseline, we benchmark our **single ViR foundation model** against three standard Vision Transformers (MAE-ViT) (Dosovitskiy et al., 2021) of varying discrete capacities: Tiny (1.08G FLOPs), Base (16.87G FLOPs), and Large (59.70G FLOPs). To eliminate confounding factors from training duration, all three MAE-ViT baselines are re-trained

Hyperparameter	Value
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Base learning rate	1×10^{-4}
Weight decay	1×10^{-6}
Optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$
Batch size	256
Learning rate schedule	Cosine decay
Warmup epochs	10
Training epochs	200
Augmentation	RandomResizedCrop ($[0.2, 1.0]$), RandomHorizontalFlip

Table 6: Pre-training hyper-parameters for ViR under the MAE framework.

from scratch under the exact same MAE recipe as our ViR model. Crucially, our comparison abandons the traditional "parameter-matching" paradigm. Instead, we evaluate the elastic compute-to-accuracy trade-off. During downstream evaluation, the identically pre-trained ViR model is elastically unrolled to varying inference depths ($L \in \{1, 2, 4, 8\}$). We then compare the performance of these specific ViR inference unrollings against the discretely sized ViT models at comparable FLOP budgets. This perfectly controlled setup enables the observed performance gains to be attributable to the structural advantages of our functional synthesis rather than layer-wise parameter inflation. All pre-training was conducted on two NVIDIA H100 GPUs (80GB memory each). Downstream linear probing evaluations were conducted on a single NVIDIA A100 GPU (40GB memory).

E Additional Experiments and Ablations

E.1 Extensive Functional Approximation

To supplement the Octagon Benchmark presented in the manuscript (Sec.3.1), we further evaluate the approximation capabilities of RBN across a broader and more rigorous set of 16 mathematical functions. As detailed in Tab.7, this extended benchmark is structured into four topological challenges: discontinuous topologies, localized and multi-scale oscillations, spatial heterogeneity, and high-dimensional spaces. All models are evaluated under identical conditions, including a fixed functional depth ($L = 4$), consistent optimization settings (AdamW), and a similar parameter budget, ensuring a fair comparison of their expressivity. The empirical results reveal several key insights into the advantages of our functional synthesis:

1) *Modeling Discontinuous Topologies.* Traditional baselines inherently impose global smoothness priors, causing them to fail at sharp boundaries (e.g., Sign Square Wave) (Sitzmann et al., 2020). While KANs (Liu et al., 2025) improve upon this via local B-splines, their performance still degrades at severe discontinuities (e.g., Sawtooth Wave) due to static parameters. In contrast, RBN consistently achieves the lowest RMSE, outperforming baselines by orders of magnitude (e.g., 1.10×10^{-4} on Absolute Kink). By dynamically predicting compositional states, RBN specializes step-like bases where the topology demands it.

2) *Adapting Spectral Bias in Multi-Scale Oscillations.* When modeling localized and multi-scale oscillations, standard dense networks suffer from severe spectral bias (Rahaman et al., 2019), favoring macroscopic trends while failing to resolve nested high frequencies (e.g., Nested Multi-scale) (Tancik et al., 2020). RBN overcomes this frequency entanglement. By allowing the Dynamic Composer to activate specialized oscillatory bases on demand, RBN accurately captures localized spikes and damped high-frequency vibrations that elude static architectures.

3) *Navigating Spatial Heterogeneity.* When governing dynamics fundamentally shift across spatial domains (e.g., 2D Smooth-Osc Boundary), MLPs and monolithic MoEs struggle with catastrophic interference (Jacobs et al., 1991), attempting to fit conflicting topologies with a shared set of static weights. RBN's Dynamic Composer shifts the composing probabilities based on input coordinates, synthesizing different functional spaces tailored to the local topological region without suffering from cross-region interference.

4) *Robustness in High-Dimensional Manifolds.* Scaling approximation to $d = 50$ will introduce the curse of dimensionality (Bellman, 1966), where dense architectures succumb to background noise

Name	Functions / Source	MLP	KAN	MoE	RNN	RBN (ours)
1. Discontinuous Topologies						
Absolute Kink	$\lvert x\rvert \cdot \cos(5\pi x)$	8.46×10^{-3}	2.42×10^{-4}	4.12×10^{-1}	2.61×10^{-1}	$\mathbf{1.10 \times 10^{-4}}$
Sign Square Wave	$\mathrm{sgn}(\sin(10\pi x))$	6.45×10^{-1}	5.68×10^{-2}	9.88×10^{-1}	8.76×10^{-1}	7.32×10^{-2}
Clipped Sine	$\max(\sin(8\pi x), 0)$	9.37×10^{-2}	1.76×10^{-3}	3.72×10^{-1}	3.00×10^{-1}	$\mathbf{9.26 \times 10^{-4}}$
Sawtooth Wave	$x - \lfloor x + 0.5 \rfloor$	3.24×10^{-2}	1.30×10^{-2}	6.33×10^{-2}	2.35×10^{-2}	$\mathbf{9.25 \times 10^{-3}}$
2. Localized and Multi-scale Oscillations						
Quadratic Chirp	$\sin(20\pi x^2)$	4.41×10^{-1}	6.74×10^{-3}	6.68×10^{-1}	5.53×10^{-1}	$\mathbf{5.56 \times 10^{-3}}$
Damped High-Freq	$e^{-3\lvert x\rvert} \sin(40\pi x)$	2.86×10^{-1}	1.19×10^{-2}	2.85×10^{-1}	2.86×10^{-1}	$\mathbf{4.72 \times 10^{-3}}$
Nested Multi-scale	$\sin(5\pi x + 0.5\cos(30\pi x))$	2.38×10^{-1}	2.89×10^{-3}	6.83×10^{-1}	3.79×10^{-1}	$\mathbf{1.41 \times 10^{-3}}$
Localized Spike	$e^{-100x^2} + 0.1\sin(4\pi x)$	4.94×10^{-4}	1.24×10^{-4}	1.36×10^{-1}	8.44×10^{-3}	$\mathbf{4.41 \times 10^{-5}}$
3. Spatial Heterogeneity						
1D Piecewise Freq	$\begin{cases}\sin(4\pi x), & x < 0\\ \sin(20\pi x), & x \geq 0\end{cases}$	4.18×10^{-1}	4.06×10^{-3}	6.91×10^{-1}	4.73×10^{-1}	$\mathbf{1.31 \times 10^{-3}}$
2D Smooth-Osc Boundary	$\begin{cases}\cos(10\pi x_1)\cos(10\pi x_2), & x_1 < 0\\ x_1^2 - x_2^2, & x_1 \geq 0\end{cases}$	3.52×10^{-1}	1.50×10^{-1}	3.85×10^{-1}	3.52×10^{-1}	$\mathbf{5.34 \times 10^{-2}}$
2D Concentric Ripples	$\sin(15\pi\sqrt{x_1^2 + x_2^2})e^{-(x_1^2+x_2^2)}$	4.17×10^{-1}	2.83×10^{-1}	4.16×10^{-1}	4.17×10^{-1}	$\mathbf{6.59 \times 10^{-2}}$
Localized Noise Patch	$x_1 + x_2 + \mathbb{I}_{\lvert x_1\rvert<0.5, \lvert x_2\rvert<0.5} \cdot 0.2\sin(50\pi x_1)$	7.01×10^{-2}	7.02×10^{-2}	7.10×10^{-2}	7.16×10^{-2}	$\mathbf{1.83 \times 10^{-2}}$
4. High-Dimensional Topologies ($d = 50$)						
High-D Additive Smooth	$\frac{1}{d}\sum_{i=1}^{d} \sin(\pi x_i)$	6.38×10^{-2}	7.47×10^{-4}	6.51×10^{-2}	6.38×10^{-2}	$\mathbf{5.46 \times 10^{-4}}$
High-D Ridge Function	$\cos(2\pi(\frac{1}{d}\sum_{i=1}^{d} x_i))$	8.59×10^{-3}	1.47×10^{-2}	2.89×10^{-2}	2.58×10^{-2}	$\mathbf{4.03 \times 10^{-3}}$
Sparse Interaction	$\sin(\pi x_1 x_2) + \cos(\pi x_3 x_4) + \frac{1}{d-4}\sum_{i=5}^{d} x_i$	3.06×10^{-1}	3.27×10^{-2}	3.47×10^{-1}	5.13×10^{-1}	$\mathbf{2.49 \times 10^{-2}}$
High-D Isotropic Gaussian	$e^{(-\frac{5}{d}\sum_{i=1}^{d} x_i^2)}$	3.62×10^{-2}	6.51×10^{-4}	3.99×10^{-2}	4.07×10^{-2}	$\mathbf{3.68 \times 10^{-4}}$

Table 7: **Extensive Functional Approximation across Different Topologies.** Performance comparison across four challenging functional categories: discontinuous, multi-scale, spatially heterogeneous, and high-dimensional topologies. All models share a consistent functional depth ($L = 4$) with similar parameter size and are optimized via AdamW for a fair assessment of structural expressivity. The evaluation metric is the RMSE, where a lower value means better approximation fidelity.

and variance explosion. RBN excels in identifying low-rank structures and isolating targeted features from massive linear backgrounds (e.g., achieving 2.49×10^{-2} on Sparse Interaction, compared to MLP's 3.06×10^{-1}). RBN preserves its high fidelity even in highly sparse high-dimensional spaces.

This evaluation verifies that RBN is not merely memorizing data, but learning a highly adaptive and combinatorial representation space. It demonstrates that the compositional space scales robustly across varying topological complexities, allowing RBN to flexibly synthesize diverse functional topologies beyond the capacity of static parameter allocation paradigms.

E.2 Implicit Neural Representations for Scientific Computing

To further validate the capacity of RBN for modeling complex physical manifolds, we evaluate our architecture on the PDEBench dataset (Takamoto et al., 2022) in implicit neural representations for scientific computing (Van Noorden & Perkel, 2023). Instead of employing highly engineered Neural Operators, we adopt the continuous coordinate-based network paradigm. This approach, which serves as the foundational backbone for Physics-Informed Neural Networks (PINNs) (Cuomo et al., 2022), tasks the model to map continuous spatiotemporal coordinates to the corresponding physical scalar fields, i.e., $f : (t, \mathbf{x}) \to u$. Therefore, we can compare RBN against standard baselines under purely data-driven training conditions.

Experimental Setup. We select three challenging physical systems characterized by distinct topological difficulties:

- **1D Burgers' Equation ($\nu = 0.001$):** Features extreme physical singularities where smooth waves collapse into sharp, non-differentiable shockwaves.
- **2D Diffusion-Reaction (Diff-React):** Features highly chaotic, multi-scale Turing patterns driven by non-linear chemical kinetics, testing the model's capacity for localized high-frequency spatial heterogeneity.
- **2D Shallow Water Equation (SWE):** Features complex radial wave reflections, interference, and surface elevation interactions in a 2D spatial domain.

We compare PINN-RBN against PINN-MLP (the default baseline) and PINN-KAN. Evaluation metrics include Root Mean Square Error (RMSE) and Normalized RMSE (nRMSE).

PDE (Takamoto et al., 2022)	**PINN-MLP**		**PINN-KAN**		**PINN-RBN (ours)**	
	RMSE	nRMSE	RMSE	nRMSE	RMSE	nRMSE
1D Burgers'	3.32×10^{-1}	5.96×10^{-1}	8.31×10^{-2}	1.39×10^{-1}	$\mathbf{7.96 \times 10^{-2}}$	$\mathbf{1.32 \times 10^{-1}}$
2D Diff-React	1.97×10^{-1}	$1.51 \times 10^{+0}$	1.98×10^{-1}	$1.53 \times 10^{+0}$	$\mathbf{1.34 \times 10^{-1}}$	$\mathbf{1.01 \times 10^{+0}}$
2D SWE	1.39×10^{-2}	1.34×10^{-2}	6.76×10^{-3}	6.50×10^{-3}	$\mathbf{5.63 \times 10^{-3}}$	$\mathbf{5.42 \times 10^{-3}}$

Table 8: **Implicit Neural Representations for Scientific Computing.** Performance comparison of coordinate-based network backbones on PDEBench (Takamoto et al., 2022), including 1D Burgers' Equation $\nu = 0.001$ (Burgers'), 2D Diffusion-Reaction Equation (Diff-React), 2D Shallow Water Equation (SWE). We evaluate the raw topological expressivity of different network primitives by mapping continuous spatiotemporal coordinates to the physical field. Evaluation metrics are RMSE and Normalized RMSE (nRMSE) that lower score means better.

Results and Analysis. As demonstrated in Tab.8, PINN-RBN consistently outperforms standard baseline architectures across all physical systems. Classical MLPs inherently suffer from spectral bias, leading to severe over-smoothing at shockwave boundaries (e.g., yielding a high RMSE of 3.32×10^{-1} on 1D Burgers') and failing to capture fine-grained wave interactions. While KANs mitigate this issue partially via local B-splines (significantly improving performance on Burgers' and SWE), their structural reliance on static knot allocation exposes a critical vulnerability when confronted with highly chaotic, unaligned spatial heterogeneity. This is evident in the 2D Diff-React task: the multi-scale Turing patterns overwhelm KAN's fixed grid resolution, causing its performance to saturate and even marginally underperform the MLP (1.98×10^{-1} vs. 1.97×10^{-1}).

In contrast, RBN treats the continuous PDE field as a dynamic composition of localized basis functions rather than relying on static weights or fixed knots. By shifting its composing probabilities based on input coordinates, RBN achieves highly adaptive spatial decoupling. It accurately activates step-like basis profiles precisely at the shockwave edges of Burgers' equation, while flexibly deploying high-frequency oscillatory bases to resolve the complex spatial domains of the Diff-React system (slashing the RMSE to 1.34×10^{-1}). This empirical evidence confirms that RBN provides a superior, parameter-efficient foundational primitive for modeling complex physical manifolds.

E.3 Comparison with Weight-Shared Recursive Models

To isolate the contribution of dynamic functional synthesis from recursive parameter sharing, we compare RBN with two weight-shared recursive baselines on the Octagon Benchmark: a standard RNN (Sherstinsky, 2020) and a Universal Transformer with Adaptive Computation Time (UT-ACT) (Dehghani et al., 2019; Graves, 2016). The RNN comparison evaluates RBN against conventional fixed-depth recursive weight sharing, whereas the UT-ACT comparison further controls for adaptive computation. We equip RBN with the same ACT mechanism (RBN-ACT, as illustrated in App.C.3), enabling the effect of dynamic functional synthesis to be examined separately from the halting strategy. All methods follow the same Octagon evaluation protocol, while their parameter counts and computational costs are reported explicitly in Tab.9.

Comparison with Recursive Weight Sharing. Across all evaluated depths, RBN achieves lower RMSE than the RNN while using fewer parameters and FLOPs. For example, at $L = 4$, RBN reduces the RMSE from 0.3801 to 0.2519, while requiring 0.0743M parameters compared with 0.1984M for the RNN. A similar trend holds at $L = 2$ and $L = 12$. These results show that the advantage of RBN cannot be attributed to recursive parameter sharing alone. Instead, dynamically composing the shared basis dictionary allows the recursively shared cell to instantiate different functional transformations across depth.

Comparison under Adaptive Computation. Under adaptive computation, RBN-ACT outperforms UT-ACT across all evaluated maximum depths. Since both methods employ shared recursive parameters and ACT-based depth control, the remaining advantage of RBN-ACT stems from its ability to synthesize heterogeneous functional transformations across recursive steps through input-conditioned basis composition. Meanwhile, compared with fixed-depth RBN, RBN-ACT reduces the executed computation while further improving approximation accuracy. For example, at $L_{\max} = 4$, RBN-ACT achieves an RMSE of 0.2232, compared with 0.2519 for fixed-depth RBN, while executing fewer recursive steps on average. This indicates that ACT complements dynamic functional synthesis by terminating computation near an input-appropriate functional depth, avoiding both insufficient and unnecessary recursive transformations.

Method	Max Depth	FLOPs	Param.	RMSE
RNN	2	0.2629	0.1984	0.4056
RBN	2	0.1372	0.0757	0.2929
UT-ACT	2	0.2641	0.1654	0.3525
RBN-ACT	2	0.2029	0.1097	**0.2916**
RNN	4	0.5302	0.1984	0.3801
RBN	4	0.2744	0.0743	0.2519
UT-ACT	4	0.5277	0.1654	0.3244
RBN-ACT	4	0.3979	0.1097	**0.2232**
RNN	12	1.5890	0.1984	0.3840
RBN	12	0.8230	0.0743	0.2732
UT-ACT	12	0.9233	0.1654	0.3232
RBN-ACT	12	0.6483	0.1097	**0.2549**

Table 9: **Comparison with weight-shared recursive baselines on the Octagon Benchmark.** RNN provides a conventional fixed-depth recursive weight-sharing baseline, while Universal Transformer (UT) with Adaptive Computation Time (ACT) provides a shared-parameter adaptive-depth baseline. RBN-ACT applies the same adaptive halting mechanism to RBN. For fixed-depth RNN and RBN, "Max Depth" denotes the executed recursive depth. For ACT variants, it denotes the maximum allowed recursive depth. Lower RMSE is better.

E.4 Dense vs. Sparse Composition

Given the structural resemblance between the Dynamic Composer and the gating mechanisms in MoE (Fedus et al., 2022), a scientific question arises: how does sparsity affect the functional synthesis capacity and the computational cost? To explore this, we vary the number of activated profiles (top-k) per unrolling step as described in App.C.2 from $k = 1$ to $k = 8$ (dense composition) with 5 rounds. Additionally, we investigate whether the auxiliary load-balancing loss ($\mathcal{L}_{\text{aux}}$), which is essential in MoE architectures, is beneficial in our RBN. As shown in Fig.10, the sparsity method in MoEs is able to be seamlessly integrated into RBNs as an effective efficiency patch. There are two important observations:

1) *Effective FLOP Reduction via top-k:* By simply adopting top-k routing, the computational cost scales down linearly. We observe that dense composition ($k = 8$) is suboptimal due to the accumulation of composition noise, which also exists in MoEs (Shazeer et al., 2017; Fedus et al., 2022). A sparse setting of $k = 2$ or 3 emerges as a sweet spot. At $k = 2$, the model not only reduces FLOPs by approximately 70% compared to the dense baseline but also achieves its lowest RMSE. This validates that RBN can leverage established sparsity techniques for practical deployment.

2) *Incompatibility of Uniform Load Balancing $\mathcal{L}_{aux}$.* In standard MoE paradigms, enforcing uniform utilization is mandatory to prevent expert collapse. However, when applied to our continuous basis synthesis, enforcing $\mathcal{L}_{\text{aux}}$ degrades approximation fidelity compared to unconstrained routing. This suggests a fundamental difference in basis-based functional synthesis that for complex topological mapping, allowing basis profiles to freely specialize and dominate distinct local features (e.g., high-frequency transients or smooth backgrounds) is mathematically more favorable than forced uniform activation (Tancik et al., 2020).

These exploratory findings elegantly resolve the trade-off between computational complexity and representational capacity. While top-k sparsity is a standard MoE technique, its unconstrained integration into RBN provides a definitive practical solution. By adopting a Top-2 routing patch without artificial load balancing, it can improve computational efficiency, slashing the FLOPs overhead by approximately 70%, while preserving the network's compositional expressivity ($\mathcal{O}(K^T)$).

E.5 With vs. Without Residual Stream

A core objective of our RBNs is to enable parameter-free depth scaling, conceptually allowing for infinite functional unrolling. However, recursively applying pure nonlinear transformations exposes the network to the degradation problem and vanishing/exploding gradients commonly observed in ultra-deep or recurrent architectures (He et al., 2016; Pascanu et al., 2013). As shown in Fig.11, to

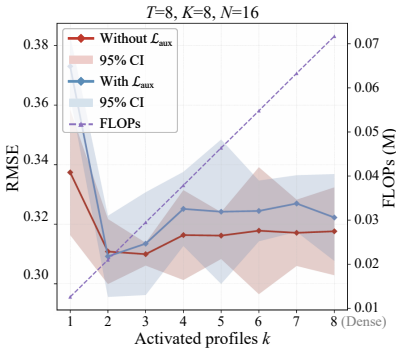


Figure 10: **Ablation of Sparse Composition.** The left axis shows RMSE (with 95% CI) for models trained with and without auxiliary load-balancing loss ($\mathcal{L}_{\text{aux}}$) (Fedus et al., 2022). The right axis illustrates FLOPs (M). Optimal performance is achieved at $k = 2$, reducing FLOPs by $\sim 70\%$ compared to dense composition ($k = 8$) and improving performance.

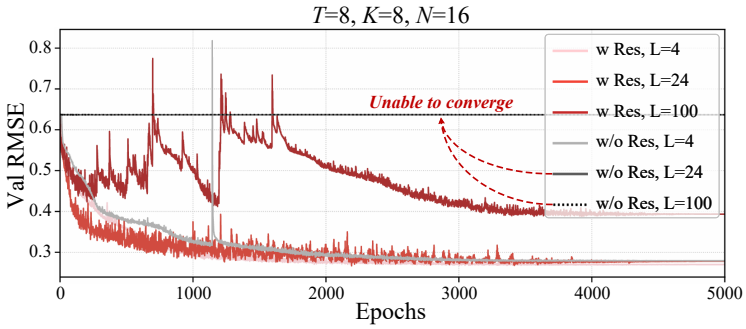


Figure 11: **Ablation of Residual Stream.** Validation RMSE trajectories over $5,000$ epochs for RBNs with (red lines) and without (grey lines) residual connections across varying functional depths ($L \in \{4, 24, 100\}$). While shallow networks ($L = 4$) converge reliably in both settings, removing the residual stream causes catastrophic optimization collapse ("Unable to converge") at extended depths ($L = 24, L = 100$). Integrating the residual stream ensures robust forward stability and enables convergence even at extreme unrolling ($L = 100$). However, the observable performance degradation at $L = 100$ relative to $L = 4$ reinforces the necessity of elastic computation to avoid the accumulating variance of excessively deep recursive chains.

evaluate the structural necessity of the residual stream formulation (Eq.6), we analyze the validation RMSE convergence across varying functional depths ($L \in \{4, 24, 100\}$) for models with and without the residual connection. To observe the convergence behavior in greater depth, we extended the training to $5,000$ epochs.

1) *Value of the Residual Stream for Depth Unrolling.* While the non-residual variant converges at a shallow depth ($L = 4$), it suffers from catastrophic optimization collapse at extended depths. For $L = 24$ and extreme unrolling at $L = 100$, the network without the residual stream fails to converge, exhibiting stagnant and severely degraded error trajectories. This empirical observation aligns with the degradation phenomenon identified in classic deep networks (He et al., 2016). Integrating the residual stream resolves this optimization bottleneck. Even under the extreme recursive depth of $L = 100$, the residual-equipped RBN avoids complete collapse, overcoming initial training turbulence to achieve optimization convergence.

2) *Performance Degradation at Extreme Depths.* We observe that while the residual stream enables forward stability for convergence, it does not immunize the network against performance degradation at extreme depths. As observed, the final converged RMSE of the $L = 100$ model is notably higher than that of its shallower $L = 4$ counterpart. This progressive degradation is a natural consequence of accumulating variance and the heightened optimization complexity inherent to excessive depth, as discovered in ResNet (He et al., 2016). This limitation reinforces the necessity of elastic computation. Rather than blindly unrolling to infinite depths, practitioners must allocate the optimal functional depth to maximize both computational efficiency and representation accuracy.

E.6 With vs. Without Layer Embedding

To evaluate the necessity of the layer embedding (l_{em}) in our RBN, we conduct an ablation study scaling the unrolling depth $L \in \{4, 24, 100\}$ for the RBNs with and without the layer embedding across 5 independent random seeds. As shown in Fig.12, we visualize the Difference RMSE $\text{RMSE}_{\text{Diff}} = \text{RMSE}_{\text{w/o } l_{\text{em}}} - \text{RMSE}_{\text{w } l_{\text{em}}}$. Under this metric, a positive value indicates that adding layer embedding improves performance (i.e., lowers the error).

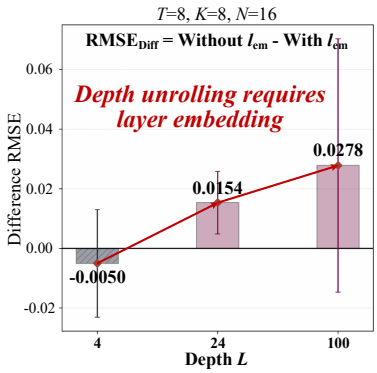


Figure 12: **Ablation of Layer Embedding (l_{em}).** A positive value indicates that incorporating l_{em} reduces the overall approximation error. While the layer embedding exhibits a negligible, slightly negative impact at shallow unrolling ($L = 4$), its performance gain amplifies as depth increases ($L = 24$ and 100). Error bars represent the 95% CI across 5 independent runs.

The results reveal a depth-dependent behavioral shift: **1)** *Shallow Unrolling ($L = 4$): Marginal to No Benefit.* At a shallow unrolling depth, the model performance is roughly equivalent regardless of the presence of layer embedding. In fact, removing l_{em} yields a negligible improvement ($\text{RMSE}_{\text{Diff}} = -0.0050$). This occurs because, for very short recurrent sequences, the hidden states themselves retain sufficient capacity to implicitly encode the current unrolling step. Adding a layer embedding causes slight parameter overhead and optimization noise, which may marginally interfere with the fitting of simple, shallow trajectories. **2)** *Deep Unrolling ($L \geq 24$): Layer Embedding is Indispensable.* As the unrolling depth increases, the performance gain from layer embedding becomes strikingly prominent. At $L = 24$, the inclusion of l_{em} reduces the RMSE by an average of 0.0154, and this advantage further amplifies to 0.0278 at the extreme depth of $L = 100$, validating that depth unrolling requires layer embedding.

The justification for this phenomenon lies in the state confusion inherent to recursive architectures (Dehghani et al., 2019). Without positional awareness (l_{em}), the Dynamic Composer must rely on the intermediate feature representations to determine its routing state $\mathbf{S}^l$. At massive depths (e.g., $L = 100$), the features will undergo homogenization, leaving the composer unable to distinguish whether it is operating at the early stages of functional synthesis or the late stages of residual refinement. The layer embedding enables the Dynamic Composer to execute depth-aware composing strategies, thereby preventing the unrolled optimization from collapsing and unlocking the potential of progressive functional synthesis.

F Interpretability of RBNs

While standard DNNs (e.g., dense MLPs) operate as opaque black boxes, basis-based models like KANs (Liu et al., 2025) have pioneered a "white-box" approach by replacing scalar weights with visualizable spline functions (Du et al., 2019). Our RBNs inherit this interpretability from KANs, but elevate it from a static, layer-wise mapping to a dynamic recursive process. By operationalizing functional synthesis, RBNs enable two dimensions of transparency: **1)** *Functional Interpretability* (what functional transformations are learned) and **2)** *Depth Interpretability* (how these transformations are assembled across steps).

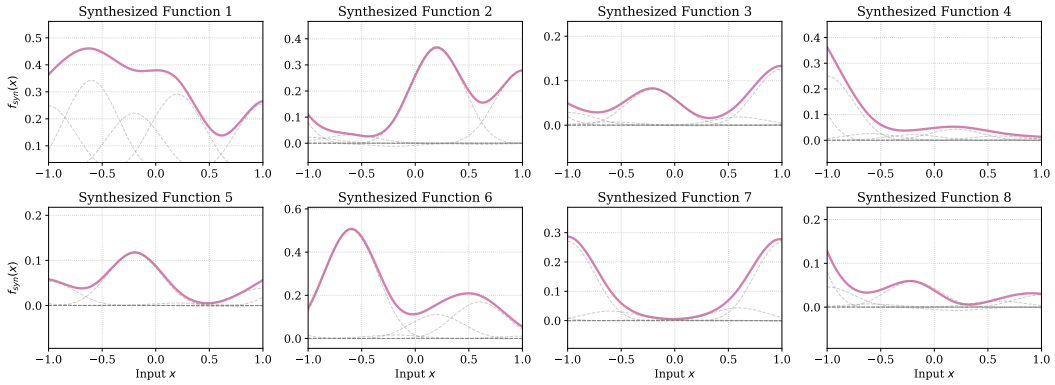


Figure 13: **Visualization of the Compositional Space.** Eight 1D functional profiles dynamically synthesized from the same trained basis dictionary $\mathbf{W}$. The solid pink lines depict the final assembled functions $f_{\text{syn}}(x)$, while the gray dashed lines show the underlying scaled B-spline bases ($c_t\phi_t(x)$). By varying the compositional states $\mathbf{S}$ while keeping the trained dictionary $\mathbf{W}$ fixed, the same parameter reservoir can synthesize both high-frequency oscillatory profiles (e.g., Functions 3 and 6) and smooth, low-frequency envelopes (e.g., Functions 4 and 7), illustrating the functional diversity accessible through composition over a shared dictionary.

F.1 Functional Interpretability: Exploring the Compositional Space

Inheriting the transparent nature of KANs, the structural basis functions $\phi_t(\cdot)$ in RBN are visualizable. Groundbreakingly, RBN departs from static function approximation by decoupling the dictionary $\mathbf{W}$ from fixed instantiated layers. Instead, the Dynamic Composer predicts independent routing states $s^l_{k,t}$ for each basis t. This means the network does not hold static functions, while it provides a dynamic composition mechanism that unlocks an exponentially large compositional space. For any given input at depth l, the synthesized 1D function can be extracted:

$$f_{\text{syn}}(x) = \sum_{t=1}^{T} c_{l,t}\phi_t(x) = \sum_{t=1}^{T} \left(\sum_{k=1}^{K} w_{k,t} s^l_{k,t} \right) \phi_t(x). \tag{20}$$

To validate the expressivity of this white-box synthesis, we extract the finalized dictionary $\mathbf{W} \in \mathbb{R}^{K\times T}$ after training on the Octagon Benchmark. To simulate the Dynamic Composer's response to different input contexts, we inject distinct, randomly sampled compositional states $\mathbf{S}$. We then plot the resulting dynamically assembled functions $f_{\text{syn}}(x)$.

As shown in Fig. 13, the same trained dictionary parameters $\mathbf{W}$ can be recomposed into substantially different 1D nonlinear profiles through different compositional states $\mathbf{S}$. The solid curves represent the synthesized functions, while the dashed curves reveal their underlying weighted B-spline components ($c_{l,t}\phi_t(x)$). We observe a broad repertoire of functional behaviors, ranging from high-frequency oscillatory profiles to smooth decaying and low-frequency envelopes. This controlled visualization demonstrates that functional diversity can arise from composition over a fixed parameter reservoir, without introducing additional transformation parameters.

A property of RBN is its adherence to the Kolmogorov-Arnold representation theorem (Kolmogorov, 1963) that decomposes complex topologies into univariate operations. Even though the Octagon Benchmark presents a highly heterogeneous 2D input space, the RBN operates via element-wise 1D projections. The transparent superposition in Fig.13 confirms that RBN solves complex multi-dimensional topologies not by memorizing opaque multidimensional manifolds, but by dynamically synthesizing suitable functions. This visualization illustrates the compositional mechanism underlying transformation-level heterogeneity: although the basis dictionary $\mathbf{W}$ remains fixed, different compositional states can synthesize substantially different functional transformations without introducing additional transformation parameters.

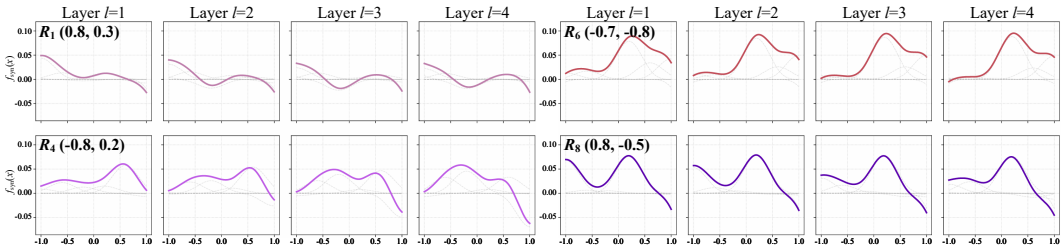


Figure 14: **Visualization of Functional Morphing.** By extracting the dynamic routing probabilities at each recursive layer l and combining them with the shared basis dictionary, we reconstruct the functions synthesized by the network for four distinct topological regions. The gray dashed lines reveal the underlying weighted B-spline components, while the solid colored curves represent the final composed function $f^l_{\text{syn}}(x)$. This layer-wise tracking demonstrates two key properties of RBNs: 1) *Adaptivity*, where the model dynamically constructs completely different geometric profiles based on the input; and 2) *Progressivity*, where the macroscopic shape is established in early layers ($l \leq 2$) and refined via residual fine-tuning in later layers ($l \geq 3$).

F.2 Depth Interpretability: Extraction of Functional Morphing

Beyond the static functional interpretability of KANs, RBN provides transparency into the evolution of the model's functional depth. In standard MLPs, tracking feature evolution across layers involves interpreting opaque, high-dimensional weight matrices. In contrast, because the compositional states $\mathbf{S}^l \in \mathbb{R}^{K\times T}$ dictate the routing probabilities at each recursive step, we are able to reconstruct the exact mathematical operations applied at any depth. By combining these routing states with the shared basis dictionary, we project the opaque concept of "layer depth" into a human-readable morphing of 1D functional curves.

To demonstrate this white-box property, Fig.14 extracts the dynamically synthesized functions $f^l_{\text{syn}}(x)$ and their underlying weighted basis components across unrolling steps $l \in \{1, \ldots, 4\}$. We track inputs originating from four distinct regions of the Octagon benchmark ($\mathcal{R}_1, \mathcal{R}_6, \mathcal{R}_4$, and $\mathcal{R}_8$). These trajectories provide direct evidence for transformation-level heterogeneity: although the basis dictionary and RBN cell are shared across recursive steps, the evolving compositional states $\mathbf{S}^l$ instantiate distinct functional transformations f^l_{syn}. The resulting functional trajectories further reveal two complementary behaviors:

Adaptive Functional Synthesis for Regions. Instead of applying a uniform transformation across the entire input space in the existing DNNs (Liu et al., 2025), the empirical results prove that RBN acts as an adaptive function synthesizer. The visualization shows that the synthesized curves drastically differ depending on the topological region. For high-frequency, complex regions like $\mathcal{R}_1$ and $\mathcal{R}_6$, the Dynamic Composer actively activates oscillatory, multi-peak basis primitives to capture intense local variance. Conversely, for the smooth polynomial region $\mathcal{R}_8$, the network synthesizes a stable, low-frequency geometric envelope. This demonstrates that RBN decouples spatial reasoning, composing functionally appropriate mathematical operations based on the local structural demands of the input. This step-by-step tracking shows that recursive depth in RBN is not merely the repeated application of a fixed functional mapping. Instead, the shared parameter reservoir is progressively recomposed into different transformations across steps, providing an interpretable realization of the functional heterogeneity enabled by dynamic functional synthesis.

Progressive Residual Refinement in Functional Morphing. By tracing the functions across depth, a step-wise developmental law emerges: a. Early-stage structural establishment. During the initial recursive steps ($l \leq 2$), the network focuses on constructing the macroscopic backbone of the function. The gray dashed lines reveal active reorganization of the underlying B-spline primitives to establish the primary peaks and valleys. Notably, for simpler regions like $\mathcal{R}_8$, this optimal geometry is identified and converges almost immediately at $l = 1$. b. Late-stage residual fine-tuning. As the depth increases ($l \geq 3$), the global shape of the synthesized functions gradually stabilizes. Instead of drastic morphological shifts, the routing mechanism transitions to fine-tuning the local residual amplitudes (e.g., the slight amplitude adjustments of the secondary peaks in $\mathcal{R}_4$ and $\mathcal{R}_6$ at $l = 4$).

This step-by-step functional tracking mathematically proves that RBN dynamically synthesizes its transformations progressively. Rather than blindly applying rigid transformations across redundant

layers, RBN iteratively sculpts the required mathematical topography on the fly, validating our elastic computation in a white-box manner.

F.3 Discussion on the Interpretability of RBNs

The empirical visualizations presented in the preceding sections highlight two fundamental theoretical advantages of the RBN architecture that extend beyond mere visual transparency. By structurally enforcing 1D functional combinations, RBN redefines model interpretability from two perspectives:

Analytic Transparency vs. Post-hoc Approximation. A major limitation in standard DNNs is that interpretability relies heavily on post-hoc explainer methods (e.g., SHAP, Integrated Gradients, or attention heatmaps) (Du et al., 2019; Lundberg & Lee, 2017). These methods provide only local approximations of feature importance, which are often mathematically flawed, fragile, or inconsistent with the model's true internal logic. In contrast, RBN enables analytic transparency inherited from KANs. Because the network's non-linear operations are formulated as additive compositions of splines weighted by composing probabilities, the exact contribution of any basis function t at any depth l is able to be analytically computed. This ensures that our extracted curves are not heuristic guesses, but the exact mathematical readouts of the network's localized reasoning process. RBN eliminates the need for external explainers by being inherently self-explanatory.

Towards Dynamic Symbolic Discovery. RBNs preserve and expand upon the symbolic regression capabilities of KANs. Because the network synthesizes 1D curves, the trained basis functions can theoretically be mapped to known mathematical formulas (e.g., sine waves, polynomials, or exponentials). This allows RBN to act as a dynamic equation generator. Instead of fitting a single, static formula to the entire dataset, the network selects and combines these symbolic primitives on the fly. It essentially "writes" custom mathematical formulas tailored to different local input regions. This dynamic routing opens exciting possibilities for AI for Science (Van Noorden & Perkel, 2023), where the model could potentially discover and output varying physical laws across different functions or depth states.

G Limitations and Future Directions

RBN provides a modular formulation of dynamic functional synthesis whose compositional mechanism is not restricted to a particular basis family or architectural backbone. While the current results demonstrate the potential of this formulation across functional approximation and visual representation learning, several limitations remain in its present implementation and evaluation. Here, we discuss these limitations and outline future directions for improving the systems efficiency, functional primitives, interpretability, and architectural generality of RBNs.

G.1 Limitations

Hardware Efficiency. Although RBN decouples functional depth from parameter depth and provides favorable parameter and theoretical FLOP trade-offs, these advantages are challenging to translate into equivalent wall-clock efficiency. As reported in App.B.2, ViR exhibits lower throughput and higher inference latency than highly optimized dense Transformer operations under comparable computational budgets. This gap reflects an important implementation-level limitation: basis evaluation and dynamic composition are realized through general-purpose deep-learning operators and do not yet benefit from the same degree of hardware and kernel optimization as standard dense GEMM operations (Dao, 2024). Furthermore, while recursive parameter sharing keeps the transformation parameter budget independent of L, increasing the executed recursive depth still increases the runtime and activation-memory cost during training.

Basis Instantiation. The current RBN architecture is instantiated with a single finite family of localized B-spline (Schumaker, 2007) bases defined on a fixed grid. While this provides an effective and interpretable functional parameterization, it also imposes a specific inductive bias on the transformations that can be synthesized efficiently. Functional structures that are less compatible with localized spline primitives, such as those requiring strong global continuity or high-frequency oscillatory variations, may require a substantially denser basis expansion to achieve accurate ap-

proximation (Tancik et al., 2020). Consequently, the expressivity of dynamic functional synthesis is still constrained by the functional coverage of the underlying basis set.

G.2 Future Directions

Hardware-Aware Systems Optimization. The repeated use of a shared basis dictionary provides opportunities for parameter reuse and reduced memory traffic. Future work will investigate fused kernels for basis evaluation and composition, more efficient reuse of shared dictionary parameters, and implementations that reduce kernel-launch and intermediate-memory overhead (Dao, 2024). Detailed accelerator profiling will further help characterize arithmetic intensity, Tensor Core utilization, and cache behavior, and determine whether hardware-aware implementations can translate the structural parameter sharing of RBN into improved wall-clock efficiency.

Heterogeneous Basis Integration. The Learnable Basis Dictionary is structurally agnostic to the choice of basis functions. While our implementation utilizes B-splines for their local support and smoothness properties (Schumaker, 2007), the framework naturally accommodates heterogeneous basis sets as domain-specific inductive biases. For instance, combining Fourier features, which provide a natural representation for high-frequency structure (Tancik et al., 2020), with orthogonal polynomial bases (Boyd, 2001) could allow the Dynamic Composer to synthesize hybrid functions that combine complementary functional characteristics. Such joint composition may enable RBN to synthesize richer functional structures across different domains.

Basis-Level Interpretability Analysis. RBN's functional decomposition provides a structured interface for analyzing the transformations synthesized by the model. Beyond the functional visualizations considered in this work, future studies could analyze the Dynamic Composer's routing states $\mathbf{S}$ through causal intervention (Meng et al., 2022) and feature attribution (Sundararajan et al., 2017). Such analyses could characterize how individual basis functions and their compositions contribute to model predictions and how these contributions evolve across recursive depth, providing a more systematic understanding of the learned functional composition process.

Broader Architectural Instantiations. ViR demonstrates that dynamic functional synthesis is able to be incorporated into a Transformer by replacing static channel-mixing transformations with recursively synthesized functions. A natural next step is to investigate whether the same principle generalizes to other architectural families and iterative computational processes. For convolutional models, RBN cells could operate over local receptive fields to combine dynamic functional synthesis with spatial inductive biases (Li et al., 2021). For diffusion models, the recursive formulation could provide a shared functional parameter reservoir across iterative denoising steps, with timestep conditioning guiding the Dynamic Composer to synthesize step-specific transformations (Ho et al., 2020). Evaluating these extensions would help establish the generality of dynamic functional synthesis beyond the settings considered in this work.