CORE-KAN: CONTINUOUS VISION KERNELS WITH KOLMOGOROV-ARNOLD NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional convolutional kernels are typically defined on fixed discrete grids, limiting their ability to accommodate heterogeneous local structures. Existing adaptive operators improve flexibility, but often couple geometric scale variation with content-dependent filtering, while incurring high computational cost from per-location kernel generation. To decouple geometric scale adaptation from content-dependent filtering while avoiding expensive per-location kernel generation, we propose Continuous Relative-scale KAN (Core-KAN), a relative-scale-conditioned continuous convolution operator. In detail, Core-KAN maps input features into a compact latent basis space and uses a lightweight scale controller to predict local scales relative to an exponential moving average reference. A KAN-based generator represents depth-wise kernel bases as continuous coordinate functions, allowing the operator to synthesize spatial filters at arbitrary resolutions rather than being confined to a fixed lattice. Instead of synthesizing independent kernels at every location, it constructs a compact bank of scale-conditioned kernel responses and interpolates them according to the predicted local scale map. An independent mixing controller further combines the interpolated basis responses based on local content, explicitly decoupling geometric scale adaptation from content-dependent filtering. Together with lightweight pointwise projections, this design forms a low-rank dynamic convolution that scales efficiently with kernel size and can be readily integrated into hierarchical vision backbones. Extensive experiments across three representative computer vision tasks demonstrate that Core-KAN consistently outperforms strong convolutional and dynamic-kernel baselines while introducing only marginal parameter and computational overhead. Core-KAN provides an efficient and general framework for continuous, scale-adaptive convolution across diverse vision tasks.

1 INTRODUCTION

Local aggregation is a fundamental mechanism for building visual representations. Convolution remains one of the most reliable choices because a small spatially shared kernel provides translation equivariance, computational efficiency, and a strong locality bias Krizhevsky et al. (2012); Simonyan & Zisserman (2015); He et al. (2016); Liu et al. (2022); Woo et al. (2023). Yet the same property that makes convolution efficient also limits its adaptivity: a fixed discrete kernel profile is applied to textures, object boundaries, thin structures, and homogeneous regions alike. Modern vision backbones therefore face a recurring tension between the efficiency of shared convolutional operators and the need for location-specific spatial reasoning.

A direct way to increase adaptivity is to generate or select a different kernel at each location. However, fully position-specific kernels weaken the parameter sharing that makes convolution transferable, and dense kernel synthesis is expensive in both computation and memory. Existing approaches address parts of this tension. Large-kernel and multi-branch networks enlarge or diversify receptive fields, but still operate over finitely parameterized filters Szegedy et al. (2015); Li et al. (2019); Ding et al. (2022); Liu et al. (2023); Ding et al. (2024). Conditional and dynamic convolutions improve content adaptivity by routing among learned components or modulating kernel weights Yang et al. (2019); Chen et al. (2020); Ma et al. (2020); Li et al. (2022); Jia et al. (2016); Zhou et al. (2021). Deformable convolutions adapt sampling locations, but still attach weights to a finite set of displaced points Dai et al. (2017); Zhu et al. (2019); Wang et al. (2023); Xiong et al. (2024). Continuous

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.

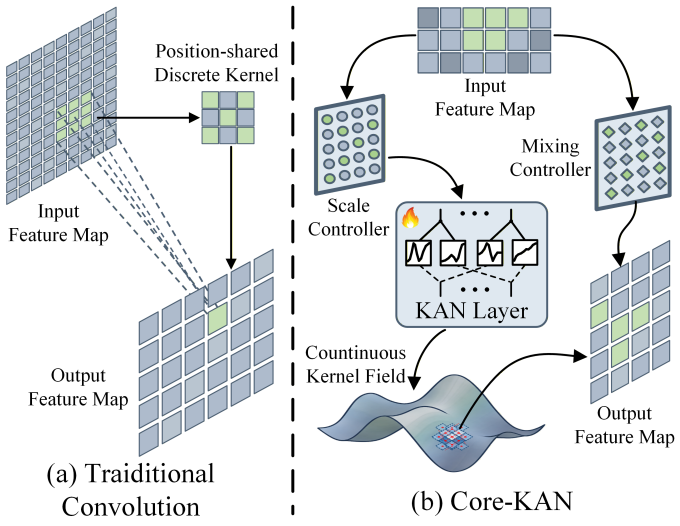

Figure 1: Conceptual comparison between conventional convolution and Core-KAN. (a) Conventional convolution shares a fixed discrete kernel across locations. (b) Core-KAN conditions a shared KAN-parameterized continuous kernel field on local relative scales and independently mixes basis responses according to content, enabling location-specific aggregation on a fixed sampling grid. Implementation details are omitted for clarity.

convolutional representations learn coordinate-to-weight functions, but their kernel fields are typically shared at the layer level rather than conditioned densely on local structure Wang et al. (2018); Wu et al. (2019); Romero et al. (2022b;a). Thus, an efficient operator that provides dense spatial adaptation through a shared continuous kernel family remains underexplored.

Spatial adaptation involves both adjusting kernel geometry and selecting content-dependent filtering patterns. Motivated by this distinction, we propose Continuous Relative-scale KAN (Core-KAN), a convolution operator that decouples these two roles. As illustrated in Figure 1, Core-KAN parameterizes shared continuous kernel fields with a Kolmogorov-Arnold Network (KAN) (Liu et al., 2025). A scale controller adapts kernel profiles through relative-scale coordinate queries on a fixed sampling grid, while an independent mixing controller combines basis responses according to local content. A compact support-and-interpolate scheme enables location-specific adaptation without explicitly generating a kernel at every position. These components provide a structured approach to spatially adaptive convolution within a low-rank basis space.

Our contributions are threefold. First, we propose Core-KAN, a relative-scale-conditioned continuous convolution operator that parameterizes a shared coordinate-to-kernel field with a KAN, enabling smooth location-specific adaptation on a fixed sampling grid. Second, we develop an efficient support-and-interpolate scheme with a dual-control mechanism that decouples geometric scale adaptation from content-dependent basis composition. Finally, extensive evaluations on ImageNet-1K, COCO, and ADE20K demonstrate that Core-KAN is an effective plug-and-play operator with consistent performance gains and interpretable adaptive behavior.

2 Related Work

2.1 Adaptive Convolutional Operators

Adaptive ConvNets enlarge the receptive field, select among filters, or generate input-dependent weights. Selective Kernel Networks and large-kernel models provide multiple or wider spatial supports Li et al. (2019); Ding et al. (2022); Liu et al. (2023); Ding et al. (2024). Recent efficient backbones further use selective large kernels, multi-scale kernels, anisotropic strip kernels, and partial-channel processing Li et al. (2023b); Cai et al. (2024); Yuan et al. (2026); Huang et al. (2026). RefConv reparameterizes filters from pretrained kernels, while SCConv reduces spatial and

channel redundancy Cai et al. (2025); Li et al. (2023a). These designs improve context modeling or efficiency, but their filters remain finite tensors or branches.

Conditional and dynamic convolution instead adapts learned components from the input. CondConv and Dynamic Convolution combine experts Yang et al. (2019); Chen et al. (2020); ODConv attends over spatial, channel, and kernel dimensions Li et al. (2022); KernelWarehouse assembles kernels from shared cells Li & Yao (2024); and FDConv modulates frequency-grouped kernel parameters Chen et al. (2025). Spatially varying filters are also produced by Dynamic Filter Networks, DDF, pixel-adaptive convolution, and involution Jia et al. (2016); Zhou et al. (2021); Su et al. (2019); Li et al. (2021).

2.2 Sampling Adaptation and Continuous Kernels

Deformable convolution approaches spatial adaptivity from the sampling side. By predicting offsets and modulation weights, they allow the operator to collect evidence from irregular positions around each location Dai et al. (2017); Zhu et al. (2019); Xiong et al. (2024). This mechanism is powerful when object geometry suggests that the sampling grid itself should move. Core-KAN follows a different principle. It keeps the regular convolutional grid intact and adapts the kernel function evaluated on that grid. Thus, the operator preserves the implementation structure and locality bias of convolution while allowing the filter profile to vary continuously.

Continuous kernel methods provide another route to parameter efficiency by representing weights as coordinate-conditioned functions rather than independent lattice parameters. Prior work has shown that coordinate-to-weight mappings can compactly describe kernels for point clouds, long sequences, or large spatial supports Wang et al. (2018); Romero et al. (2022b;a); Kim & Park (2023). However, these continuous kernels are usually learned as layer-level or globally controlled functions. They are not primarily designed for dense, location-wise kernel adaptation inside standard 2D visual backbones.

2.3 KANs for Visual Modeling

Kolmogorov-Arnold Networks replace fixed scalar weights and activations with learnable univariate edge functions, giving them a different functional parameterization from conventional multilayer perceptrons Liu et al. (2025). This makes the kernel transformation traceable as combinations of one-dimensional functions. Unlike MLPs, which apply high-dimensional nonlinear changes, each input coordinate's effect on the output can be understood through the shape of its corresponding edge function. This gives the kernel evolution explicit geometric meaning. Existing visual uses of KANs mainly treat them as feature transformation modules, inserting KAN style nonlinear mappings into local or tokenized visual representations Bodner et al. (2024); Li et al. (2025). In that setting, the KAN directly transforms image features.

Subsequent studies have broadened this paradigm to other building blocks of visual architectures. For instance, KAT replaces MLP blocks in Transformers with scalable rational-function KAN layers Yang & Wang (2025), whereas KAC incorporates a KAN-based classifier for continual visual learning Hu et al. (2025). Despite their emerging potential across vision tasks, leveraging the continuous functional parameterization of KANs to implement efficient, spatially adaptive convolution operators remains largely unexplored.

3 Method

We propose **Core-KAN**, a relative-scale-conditioned continuous convolution operator that decouples geometric scale adaptation from content-dependent basis composition. As illustrated in Figure 2, a shared Kolmogorov-Arnold Network (KAN) generates kernel bases from scale-transformed coordinates. A scale controller determines local queries relative to an operator-specific exponential moving average (EMA) reference, while an independent mixing controller combines basis responses according to local content. A compact support-and-interpolate scheme implements spatial adaptation without explicitly synthesizing a kernel at every position.

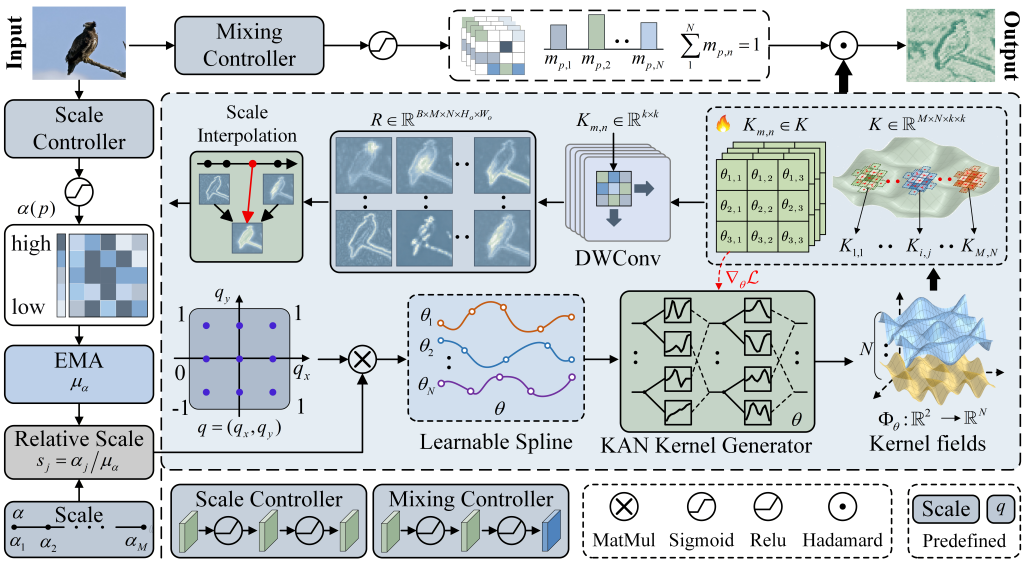

Figure 2: Overview of Core-KAN. The scale controller predicts a dense scale field $\alpha(p)$, and an operator-specific EMA reference μ_α converts absolute scale supports into relative supports. These supports transform a predefined coordinate grid q before it is evaluated by the KAN kernel generator. The learnable spline parameters θ define N shared continuous kernel fields, from which M sets of fixed-size kernels are sampled. Depthwise convolution produces a response bank $\mathcal{R}$, and each location interpolates its two neighboring support responses.

3.1 Operator Formulation

Let $X \in \mathbb{R}^{B\times C_{\text{in}}\times H\times W}$ and $Y \in \mathbb{R}^{B\times C_{\text{out}}\times H_o\times W_o}$ denote the input and output features. Core-KAN first projects the input into a compact N-dimensional basis space:

$$V = P_{\text{in}}(X), \qquad V \in \mathbb{R}^{B\times N\times H\times W}, \tag{1}$$

where P_{in} is a 1×1 convolution. For each sample b, basis n, and output location p, the operator computes a scale-adapted response $\widetilde{R}_{b,n}(p)$, modulates it with a content-dependent weight $m_b(p,n)$, and applies a pointwise output projection:

$$Y_{b,c}(p) = \sum_{n=1}^{N} W^{\text{out}}_{c,n}\, m_b(p,n)\, \widetilde{R}_{b,n}(p), \tag{2}$$

where W^{out} is the weight matrix of P_{out}. The scale branch controls the spatial profile of each basis response, whereas the mixing branch controls its contribution to the output.

For clarity, the spatial equations below use unit-stride indexing. General stride, padding, and output resolution follow the standard convolutional convention.

3.2 Decoupled Scale and Mixing Controllers

Both controllers operate on the original input X. Each contains two lightweight 3×3 convolution-GroupNorm-ReLU blocks followed by a 1×1 prediction layer.

Dense scale prediction. The scale controller C_α predicts a bounded scalar at every spatial location:

$$\begin{aligned} \widehat{\alpha}_b(p) &= \sigma(C_\alpha(X_b)(p))\,, \\ \alpha_b(p) &= \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})\,\widehat{\alpha}_b(p), \end{aligned} \tag{3}$$

where σ is the sigmoid function and $0 < \alpha_{\min} < \alpha_{\max}$.

To calibrate scale predictions across changing feature statistics, each Core-KAN layer maintains a non-learnable EMA reference. At training iteration t, it is updated by

$$\begin{aligned}\overline{\alpha}^{(t)} &= \frac{1}{BHW}\sum_{b=1}^{B}\sum_{p}\alpha_b^{(t)}(p),\\ \mu_\alpha^{(t)} &= \mathrm{clip}_{[\alpha_{\min},\alpha_{\max}]}\Big(\rho\mu_\alpha^{(t-1)} + (1-\rho)\,\mathrm{sg}\Big(\overline{\alpha}^{(t)}\Big)\Big),\end{aligned} \tag{4}$$

where ρ is the EMA momentum and $\mathrm{sg}(\cdot)$ denotes stop-gradient. The stored reference is fixed at inference time. The local relative scale is

$$s_b(p) = \frac{\alpha_b(p)}{\mu_\alpha}, \tag{5}$$

which expresses the prediction relative to the typical scale of the current operator.

Content-dependent mixing. The independent mixing controller C_m predicts a simplex distribution over the N latent bases:

$$\begin{aligned}m_b(p,:) &= \mathrm{softmax}(C_m(X_b)(p))\,,\\ m_b(p,n) \geq 0, &\qquad \sum_{n=1}^{N} m_b(p,n) = 1.\end{aligned} \tag{6}$$

These weights control content-dependent basis composition without determining the relative scale used to query the kernel field.

3.3 KAN-Parameterized Continuous Kernel Fields

Relative-scale supports. We use $M \geq 2$ uniformly spaced absolute supports and normalize them with the same EMA reference as the dense scale map:

$$\alpha_j = \alpha_{\min} + \frac{j-1}{M-1}\,(\alpha_{\max} - \alpha_{\min})\,, \qquad s_j = \frac{\alpha_j}{\mu_\alpha}, \quad j = 1,\dots,M. \tag{7}$$

Thus, local scales and support values share the same operator-calibrated coordinate system.

Continuous coordinate-to-kernel mapping. Let $\mathcal{G}_k = \{\delta_r\}_{r=1}^{k^2}$ denote the offsets of a regular $k\times k$ convolutional grid, and let $q_r = (q_{x,r}, q_{y,r}) \in [-1,1]^2$ be the normalized coordinate associated with δ_r. A shared KAN maps continuous coordinates to N scalar kernel fields:

$$\Phi_\theta : \mathbb{R}^2 \to \mathbb{R}^N, \tag{8}$$

where θ denotes its learnable parameters. The generator formulation and kernel-normalization properties are detailed in Appendix A.1.

For a relative scale s, the unnormalized n-th basis kernel is obtained by evaluating Φ_θ at the transformed coordinates:

$$G_n(s) = \mathrm{reshape}_{k\times k}\Big(\{[\Phi_\theta(sq_r)]_n\}_{r=1}^{k^2}\Big)\,. \tag{9}$$

Each sampled kernel is standardized over its spatial entries and rescaled to a Kaiming-compatible magnitude:

$$K_n(s) = \sqrt{\frac{2}{k^2}}\,\frac{G_n(s) - \mathrm{mean}(G_n(s))}{\mathrm{std}(G_n(s)) + \varepsilon}. \tag{10}$$

Sampling at the M relative supports yields the kernel bank

$$\mathcal{K} = \{K_{j,n} = K_n(s_j)\}_{\substack{j=1,\dots,M\\ n=1,\dots,N}} \in \mathbb{R}^{M\times N\times k\times k}. \tag{11}$$

These kernels are samples from shared continuous fields rather than independently stored filters. Varying s changes the queried coordinates and hence the kernel weights; the discrete sampling offsets δ_r remain fixed.

3.4 SUPPORT-AND-INTERPOLATE READOUT

Core-KAN evaluates the kernel field at the M supports and implements dense scale adaptation by interpolating their responses.

Support-sampled response bank. Each support kernel is applied depthwise to the projected feature V:

$$R_{b,j,n}(p) = \sum_{r=1}^{k^2} K_{j,n}(\delta_r)\, V_{b,n}(p + \delta_r). \tag{12}$$

Stacking the responses gives

$$\mathcal{R} \in \mathbb{R}^{B\times M\times N\times H_o\times W_o}. \tag{13}$$

The bank requires M shared depthwise convolutions in the compact basis space.

Pixel-wise scale interpolation. If the convolution changes spatial resolution, the scale map is bilinearly resized to (H_o, W_o). We reuse $\alpha_b(p)$ and $s_b(p)$ for the resized values. For neighboring supports satisfying $\alpha_j \le \alpha_b(p) \le \alpha_{j+1}$, equivalently $s_j \le s_b(p) \le s_{j+1}$, the interpolation coefficient is

$$\lambda_{b,p} = \frac{\alpha_b(p) - \alpha_j}{\alpha_{j+1} - \alpha_j} = \frac{s_b(p) - s_j}{s_{j+1} - s_j}, \qquad \lambda_{b,p} \in [0, 1]. \tag{14}$$

The equality follows from the shared positive reference μ_α. The scale-adapted response is

$$\widetilde{R}_{b,n}(p) = (1 - \lambda_{b,p})\, R_{b,j,n}(p) + \lambda_{b,p} R_{b,j+1,n}(p). \tag{15}$$

Out-of-range values are clamped to the nearest boundary; at an exact support, interpolation returns that support response.

By linearity in the kernel weights, this readout exactly applies the interpolated local kernel $\widetilde{K}_{b,n,p} = (1 - \lambda_{b,p})K_{j,n} + \lambda_{b,p}K_{j+1,n}$ without materializing it at every position. This kernel is a piecewise-linear approximation to the direct continuous query $K_n(s_b(p))$, rather than an exact evaluation in general. Appendix A.2 derives this identity; Appendix A.3 analyzes interpolation error and scale gradients under stated regularity assumptions. Increasing M gives denser scale sampling at a higher response-bank cost.

3.5 CONTENT MIXING AND LOW-RANK INTERPRETATION

Core-KAN applies the independent content weights to the scale-adapted responses:

$$Z_{b,n}(p) = m_b(p, n)\, \widetilde{R}_{b,n}(p). \tag{16}$$

When required, the mixing map is bilinearly resized to (H_o, W_o) and renormalized over n. The output is $Y = P_{\text{out}}(Z)$, as in Equation equation 2.

Let W^{in} and W^{out} denote the input and output projection matrices. Combining the projections, interpolated kernels, and content weights gives the effective location-dependent kernel:

$$W^{\text{eff}}_{b,c,c',p}(\delta) = \sum_{n=1}^{N} W^{\text{out}}_{c,n}\, m_b(p, n)\, \widetilde{K}_{b,n,p}(\delta)\, W^{\text{in}}_{n,c'}. \tag{17}$$

Here, $s_b(p)$ controls the spatial profile of $\widetilde{K}_{b,n,p}$, while $m_b(p, n)$ controls basis composition. The pointwise projections share channel-mixing factors across locations, confining spatially adaptive computation to the N-dimensional basis space. Appendix A.4 gives the corresponding channel-rank bound and computational and memory cost analysis.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Core-KAN configuration. Unless otherwise specified, Core-KAN uses a fixed 3×3 sampling grid, $N = 16$ kernel bases, and $M = 8$ uniformly spaced absolute scale supports. The scale

Table 1: ImageNet-1K validation results. Reported or reproduced training recipes differ across methods, so the table is intended as a reference comparison. **Bold** and underlined values indicate the best and second-best accuracy, respectively.

Method	Params (M) ↓	Top-1 (%) ↑	Top-5 (%) ↑
ResNet-50 [CVPR'16]	25.56	78.44	94.24
DY-Conv [CVPR'20]	100.88	79.00	94.27
ODConv [ICLR'22]	90.67	78.52	94.01
SCConv [CVPR'23]	17.69	79.89	94.76
RefConv [TNNLS'25]	36.97	79.91	94.61
PartialNet [AAAI'26]	18.00	80.61	95.13
KernelWarehouse [ICML'24]	102.02	81.05	95.21
FDConv [CVPR'25]	29.20	80.36	95.02
Core-KAN (Ours)	26.61	**81.45**	**95.68**

Table 2: COCO val2017 results using Mask R-CNN under the standard $1\times$ schedule (12 epochs). **Bold** and underlined values denote the best and second-best results, respectively; ties are marked equally.

Method	Object detection (box)						Instance segmentation (mask)					
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50 [CVPR'16]	37.9	58.7	41.2	21.6	41.5	49.3	34.5	55.6	36.8	15.9	37.1	50.4
DY-Conv [CVPR'20]	39.2	60.3	42.5	23.0	42.9	51.4	34.7	56.0	37.1	16.4	36.9	51.1
ODConv [ICLR'22]	40.1	61.5	43.6	24.0	43.6	52.3	36.7	58.5	39.6	18.6	39.0	52.8
KernelWarehouse [ICML'24]	42.4	**65.4**	46.3	**27.2**	46.2	54.6	38.9	62.0	41.5	**22.7**	42.6	53.1
FDConv [CVPR'25]	42.5	64.8	46.2	26.4	47.0	54.9	38.3	61.8	41.0	19.6	42.4	54.3
Core-KAN (Ours)	**43.0**	65.1	**47.0**	27.1	**47.1**	**55.8**	**39.5**	**62.1**	**42.4**	20.5	**42.8**	**57.2**

range is $[0.5, 2.0]$, with scale-reference EMA momentum $\rho = 0.99$. Both controllers have hidden width 32. The generator uses two KAN layers with hidden width 32, grid size 7, and spline order 3. Appendix B.1 details backbone integration, and Appendix B.2 summarizes the compute environment and reporting conventions.

Training and evaluation. Core-KAN is trained on ImageNet-1K Russakovsky et al. (2015) for 300 epochs with AdamW, a global batch size of 4,096, learning rate 4×10^{-3}, 20 warm-up epochs, cosine decay, and single-crop 224×224 validation. COCO Lin et al. (2014) uses Mask R-CNN He et al. (2017) with FPN Lin et al. (2017) for 12 and 36 epochs ($1\times$ and $3\times$). ADE20K Zhou et al. (2017) uses 160K iterations. Within each downstream benchmark, reproduced backbones share the data pipeline, schedule, task head, and evaluation protocol. Full recipes, including the different augmentations used by the two COCO schedules, appear in Appendices B.3–B.5.

4.2 Main Results

Table 1 compares Core-KAN with dynamic convolution operators and recent efficient backbones on ImageNet-1K. Core-KAN achieves 81.45% top-1 and 95.68% top-5 accuracy with 26.61M parameters. Compared with ResNet-50, it delivers relative improvements of 3.84% and 1.53% in top-1 and top-5 accuracy, respectively, with a parameter overhead of only 4.11%. Core-KAN also provides relative gains of 0.49% in both accuracy metrics over KernelWarehouse while using 73.92% fewer parameters.

On COCO, Core-KAN consistently improves Mask R-CNN under both training schedules. As shown in Table 2, Core-KAN achieves 43.0 AP^{box} and 39.5 AP^{mask}, corresponding to relative improvements of 13.46% and 14.49% over ResNet-50, respectively. Compared with the strongest competing methods, Core-KAN further improves box AP over FDConv by 1.18% and mask AP over KernelWarehouse by 1.54%. The improvements are consistent across object scales: relative to ResNet-50, Core-KAN achieves gains of 25.46%, 13.49%, and 13.18% in AP^{box}_S, AP^{box}_M, and

Table 3: COCO val2017 results using Mask R-CNN under the standard 3× schedule (36 epochs). **Bold** and underlined values denote the best and second-best results, respectively; ties are marked equally.

Method	Object detection (box)						Instance segmentation (mask)					
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50 [CVPR'16]	40.9	61.3	44.8	24.4	44.6	52.3	37.1	58.3	39.9	18.4	39.8	52.9
DY-Conv [CVPR'20]	41.9	63.0	45.7	25.8	45.6	53.8	36.9	58.5	39.7	18.8	39.4	53.2
ODConv [ICLR'22]	42.6	64.0	46.6	27.0	46.3	54.8	38.7	60.6	42.0	21.4	41.2	54.7
KernelWarehouse [ICML'24]	45.6	67.5	49.8	29.8	49.4	**59.0**	41.5	63.0	44.8	24.2	43.9	58.5
FDConv [CVPR'25]	45.7	67.3	50.4	30.5	49.7	58.4	41.7	62.5	45.0	24.0	44.2	58.7
Core-KAN (Ours)	**46.2**	**67.8**	**50.5**	**31.3**	**49.8**	58.9	**42.2**	**63.8**	**45.1**	**24.8**	**44.5**	**59.2**

Table 4: ADE20K validation results with a ResNet-50 backbone. Params denotes the complete segmentation model. **Bold** and underlined values denote the best and second-best results, respectively.

Method	Params (M) ↓	mIoU (%) ↑	mAcc (%) ↑
ResNet-50 [CVPR'16]	66.00	40.00	49.61
DY-Conv [CVPR'20]	140.00	41.20	51.05
ODConv [ICLR'22]	131.00	42.36	52.51
KernelWarehouse [ICML'24]	141.00	43.20	53.30
FDConv [CVPR'25]	70.00	43.50	53.63
Core-KAN (Ours)	67.50	**44.19**	**54.38**

AP_L^{box}, respectively, together with corresponding gains of 28.93%, 15.36%, and 13.49% for mask prediction. Under the longer 3× schedule, Core-KAN obtains 46.2 AP^{box} and 42.2 AP^{mask} (Table 3), yielding relative improvements of 12.96% and 13.75% over ResNet-50, respectively. It also outperforms FDConv, the strongest competing method in terms of overall AP under this schedule, by 1.09% in box AP and 1.20% in mask AP. Core-KAN ranks first on five of the six box metrics and all six mask metrics. The only exception is AP_L^{box}, where its result is only 0.17% lower than that of KernelWarehouse (58.9 versus 59.0). These results demonstrate that the advantages of Core-KAN remain consistent under longer training and across detection and instance-segmentation tasks. Appendix C.3 further analyzes custom localization gains across relative-area and contour-complexity groups.

On ADE20K, Core-KAN achieves 44.19% mIoU and 54.38% mAcc with 67.50M parameters (Table 4). These results represent relative improvements of 10.47% and 9.61% over ResNet-50, respectively, with a parameter overhead of only 3.57% in the complete segmentation model. Core-KAN further improves mIoU and mAcc over FDConv by 1.59% and 1.40%, respectively, while using 2.27% fewer parameters.

4.3 Continuous Kernels and Interpolation

Figure 3 examines whether the KAN represents a continuous kernel family and, critically, whether this representation provides interpretable kernel evolution. Queries at increasing relative scales change both the signs and spatial arrangements of the normalized weights while retaining a fixed 3 × 3 sampling grid. The ordered, basis-specific PCA trajectories reveal that each basis function captures a distinct and interpretable pattern of scale-dependent kernel transformations. Dense queries form ordered, basis-specific PCA trajectories, demonstrating that the KAN learns semantically meaningful continuous trajectories rather than arbitrary discrete lookup tables.

Figure 4 compares support-based interpolation with direct per-location continuous-kernel evaluation. The approximation error decreases consistently as the number of supports increases across all scale quantiles and summary statistics. The default setting of $M = 8$ substantially reduces the error relative to smaller support sets, while $M = 16$ provides further improvement at additional response-bank cost. Thus, eight supports offer a practical trade-off between approximation fidelity and computational efficiency. Appendix C.2 reports complementary Stage 3 controller diagnostics.

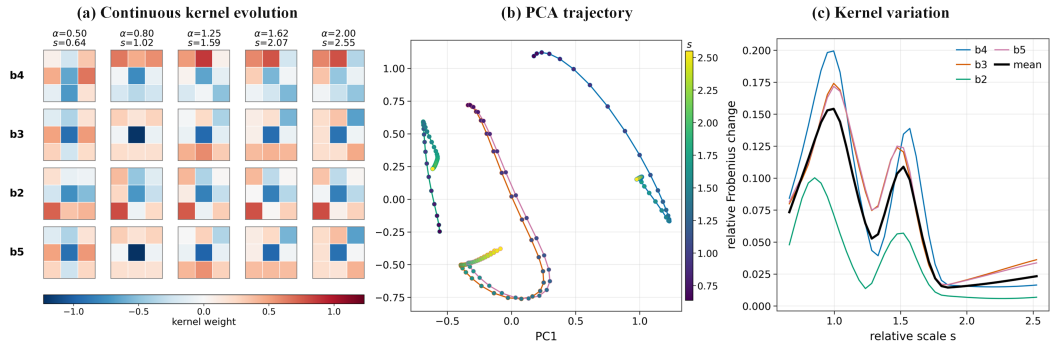

Figure 3: Evolution of learned 3×3 kernels in the last Core-KAN block. (a) Four basis fields queried at five relative scales. (b) PCA trajectories of densely sampled kernels. (c) Relative Frobenius change between adjacent queries. Smooth, basis-specific trajectories indicate a continuous kernel field rather than a discrete lookup table.

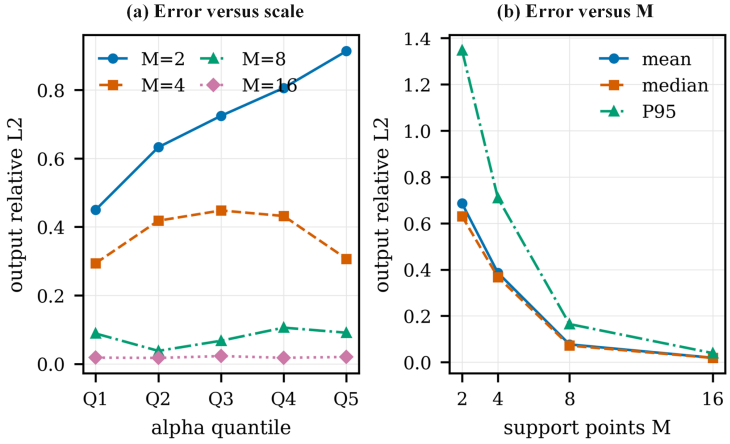

Figure 4: Interpolation fidelity to direct continuous-kernel evaluation. (a) Relative output error across five scale quantiles for different support counts M. (b) Mean, median, and 95th-percentile error. Denser supports consistently reduce the approximation error.

Locations closer to ground-truth boundaries exhibit larger absolute log-relative scale offsets and modestly more concentrated basis mixtures.

5 Conclusion

Core-KAN introduces spatial adaptation through a shared continuous kernel field. Relative-scale queries vary kernel profiles on a fixed sampling grid, an independent controller mixes basis responses according to content, and support-based interpolation avoids explicit kernel synthesis at every location. Experiments on ImageNet-1K, COCO, and ADE20K show improvements over ResNet-50 with modest parameter overhead. Kernel and interpolation analyses characterize the continuous formulation and its approximation, while supplementary statistics describe controller behavior near boundaries. A remaining limitation is that response-bank computation and storage grow with the number of scale supports; reducing this cost while maintaining interpolation fidelity is a direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript language polishing, formula typesetting, and figure caption refinement. We did not use generative AI tools for the development of the core methodology, theoretical or mathematical claims, experimental design or implementation, or result analysis and interpretation; other research tasks requiring disclosure, such as synthetic data generation and qualitative data analysis, are not applicable to this work. Additionally, we used generative AI tools for preliminary literature-search assistance. We reviewed all AI-assisted work and manually cross-checked AI-polished text against our original arguments. All experiment-related descriptions were independently verified by two authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study evaluates CoreKAN on publicly available benchmark dataset. No new data are collected from human participants, and no experiments involving human subjects or deployed decision-making systems are conducted. The datasets are used for model training, validation, and evaluation within the experimental protocols described in this paper.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure the reproducibility of our work. The complete method specification, including the KAN-parameterized kernel field formulation (Section 3.3), support-and-interpolate readout (Section 3.4), and the exact local-kernel equivalence proof (Appendix A.2), are provided in the main text and appendices. All architectural details, hyperparameters, optimization settings, and data augmentation protocols for ImageNet-1K, COCO, and ADE20K are specified in Appendices B.1–B.5. Model training uses a fixed random seed with distributed-rank offsets. We will release the source code, configuration files, and pretrained checkpoints upon paper acceptance to facilitate replication of all reported results. The datasets used are publicly available and referenced in the paper with their standard preprocessing pipelines.

REFERENCES

Alexander Dylan Bodner, Antonio Santiago Tepsich, Jack Natan Spolski, and Santiago Pourteau. Convolutional kolmogorov–arnold networks. *arXiv preprint arXiv:2406.13155*, 2024.

Xinhao Cai, Qiuxia Lai, Yuwei Wang, Wenguan Wang, Zeren Sun, and Yazhou Yao. Poly kernel inception network for remote sensing detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27706–27716, 2024.

Zhicheng Cai, Xiaohan Ding, Qiu Shen, and Xun Cao. RefConv: Reparameterized refocusing convolution for powerful ConvNets. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):11617–11631, 2025.

Linwei Chen, Lin Gu, Liang Li, Chenggang Yan, and Ying Fu. Frequency dynamic convolution for dense image prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30178–30188, 2025.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 764–773, 2017.

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975, 2022.

Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. UniRepLKNet: A universal perception large-kernel ConvNet for audio, video, point cloud, time-series and image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5513–5524, 2024.

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, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969, 2017.

Yusong Hu, Zichen Liang, Fei Yang, Qibin Hou, Xialei Liu, and Ming-Ming Cheng. KAC: Kolmogorov–arnold classifier for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15297–15307, 2025.

Haiduo Huang, Tian Xia, Wenzhe Zhao, and Pengju Ren. PartialNet: Compute less, perform better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21930–21938, 2026.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Sanghyeon Kim and Eunbyung Park. SMPConv: Self-moving point representations for continuous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10289–10299, 2023.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Chao Li and Anbang Yao. KernelWarehouse: Rethinking the design of dynamic convolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29201–29221. PMLR, 2024.

Chao Li, Aojun Zhou, and Anbang Yao. Omni-dimensional dynamic convolution. In *International Conference on Learning Representations*, 2022.

Chenxin Li, Xinyu Liu, Wuyang Li, Cheng Wang, Hengyu Liu, Yifan Liu, Zhen Chen, and Yixuan Yuan. U-KAN makes strong backbone for medical image segmentation and generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(5):4652–4660, 2025.

Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12321–12330, 2021.

Jiafeng Li, Ying Wen, and Lianghua He. SCConv: Spatial and channel reconstruction convolution for feature redundancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6153–6162, 2023a.

Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 510–519, 2019.

Yuxuan Li, Qibin Hou, Zhaohui Zheng, Ming-Ming Cheng, Jian Yang, and Xiang Li. Large selective kernel network for remote sensing object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16794–16805, 2023b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125, 2017.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Zhangyang Wang. More ConvNets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In *International Conference on Learning Representations*, 2023.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. WeightNet: Revisiting the design space of weight networks. In *Computer Vision – ECCV 2020*, volume 12360 of *Lecture Notes in Computer Science*, pp. 776–792. Springer, 2020.

David W. Romero, Robert-Jan Bruintjes, Erik J. Bekkers, Jakub M. Tomczak, Mark Hoogendoorn, and Jan C. van Gemert. FlexConv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a.

David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11166–11175, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2589–2597, 2018.

Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. InternImage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14408–14419, 2023.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16133–16142, 2023.

Wenxuan Wu, Zhongang Qi, and Fuxin Li. PointConv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

Yuwen Xiong, Zhiqi Li, Yuntao Chen, Feng Wang, Xizhou Zhu, Jiapeng Luo, Wenhai Wang, Tong Lu, Hongsheng Li, Yu Qiao, Lewei Lu, Jie Zhou, and Jifeng Dai. Efficient deformable ConvNets: Rethinking dynamic and sparse operator for vision applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5652–5661, 2024.

Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. CondConv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Xingyi Yang and Xinchao Wang. Kolmogorov–arnold transformer. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xinbin Yuan, Zhaohui Zheng, Yuxuan Li, Xialei Liu, Li Liu, Xiang Li, Qibin Hou, and Ming-Ming Cheng. Strip R-CNN: Large strip convolution for remote sensing object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12259–12267, 2026.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 633–641, 2017.

Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6647–6656, 2021.

Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable ConvNets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9308–9316, 2019.

A Additional Method Details

This appendix develops the formulation in Section 3. Appendix A.1 specifies the generator and the role of kernel normalization; Appendix A.2 derives the response-interpolation identity. Appendices A.3 and A.4 then analyze interpolation accuracy, scale sensitivity, channel factorization, and computational cost.

A.1 KAN Layer Formulation and Kernel Normalization

Shared coordinate-to-kernel generator. The generator $\Phi_\theta : \mathbb{R}^2 \to \mathbb{R}^N$ maps scale-transformed coordinates to N kernel-field values. For a KAN with L layers, let d_ℓ denote the width of layer ℓ, with $d_0 = 2$ and $d_L = N$. Its layerwise computation is

$$z_v^{(\ell+1)} = \sum_{u=1}^{d_\ell} \phi_{v,u}^{(\ell)}\left(z_u^{(\ell)}\right), \qquad z^{(0)} = x, \tag{18}$$

where $\ell = 0, \ldots, L-1$, $v = 1, \ldots, d_{\ell+1}$, and $\phi_{v,u}^{(\ell)}$ is a learnable univariate edge function parameterized by splines. The output is $\Phi_\theta(x) = z^{(L)}$. The default two-layer generator has widths $2 \to 32 \to N$, with $N = 16$, spline grid size 7, and spline order 3.

For grid coordinate q_r and relative scale s, the input is $x = sq_r$. Output component n supplies the unnormalized weight at the fixed convolutional offset δ_r for basis n. Evaluating the k^2 coordinates and reshaping the outputs gives $G_n(s)$ in Equation (9); Equation (10) then produces $K_n(s)$. Generator parameters are shared across supports and spatial locations within an operator. Consequently, local adaptation changes the queried kernel profile without changing the offsets used to sample the feature map.

Normalization and interpolation. Spatial centering makes each support kernel zero-sum, a property retained by a linear interpolation of support kernels. The normalization sets the scale of each sampled kernel, but an interpolated kernel need not have the same norm. For example, if two vectorized support kernels a and b have equal norm c, then

$$\|(1-\lambda)a + \lambda b\|_2^2 = c^2 - \lambda(1-\lambda)\|a-b\|_2^2. \tag{19}$$

Thus, changes in both kernel shape and response magnitude can occur between supports. The numerical stabilizer in Equation (10) can also make support norms slightly unequal. The readout interpolates the already normalized support responses; it does not apply another normalization to the resulting local kernel. This ordering is essential to the exact identity derived next.

A.2 Response Interpolation and Local-Kernel Equivalence

Figure 5 illustrates how neighboring support responses are combined. We distinguish the exact equivalence to applying an interpolated kernel from the approximation to directly querying the continuous kernel at every location.

Direct query and interpolation coordinates. For sample b, basis n, and output location p, the direct response is

$$R_{b,n}^\star(p) = \sum_{r=1}^{k^2} K_n(s_b(p))\,(\delta_r) V_{b,n}(p+\delta_r). \tag{20}$$

Core-KAN instead evaluates the generator at a shared support set. Since the local prediction and supports use the same positive reference μ_α,

$$\frac{s_b(p) - s_j}{s_{j+1} - s_j} = \frac{\alpha_b(p)/\mu_\alpha - \alpha_j/\mu_\alpha}{\alpha_{j+1}/\mu_\alpha - \alpha_j/\mu_\alpha} = \frac{\alpha_b(p) - \alpha_j}{\alpha_{j+1} - \alpha_j}.$$

Absolute and relative coordinates therefore yield the same interpolation coefficient. The reference still affects the sampled kernels through $s_j = \alpha_j/\mu_\alpha$.

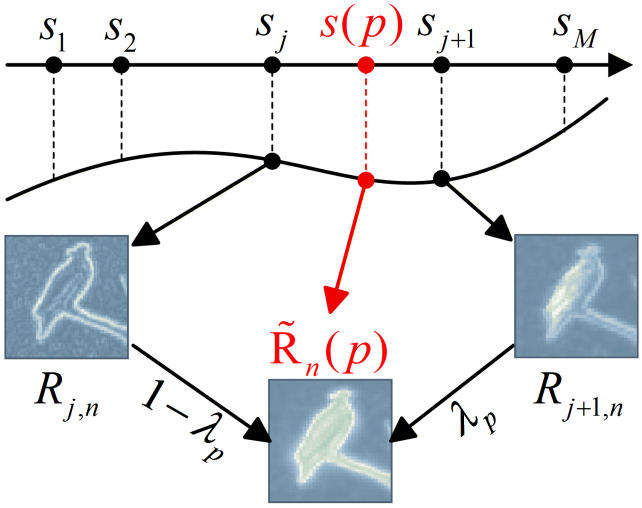

Figure 5: Response-space interpolation at location p. For $s_j \le s(p) \le s_{j+1}$, the support responses $R_{j,n}(p)$ and $R_{j+1,n}(p)$ are combined with weights $1-\lambda_p$ and λ_p. This combination exactly applies the interpolated local kernel; agreement with a direct continuous-kernel query is generally approximate. Batch indices are omitted.

Exact local-kernel identity. Define the interpolated kernel as

$$\widetilde{K}_{b,n,p} = (1-\lambda_{b,p})K_{j,n} + \lambda_{b,p}K_{j+1,n}. \tag{21}$$

Substituting Equation (12) into Equation (15) gives

$$\begin{aligned}\widetilde{R}_{b,n}(p) &= (1-\lambda_{b,p})R_{b,j,n}(p) + \lambda_{b,p}R_{b,j+1,n}(p)\\ &= \sum_{r=1}^{k^2}\left[(1-\lambda_{b,p})K_{j,n}(\delta_r) + \lambda_{b,p}K_{j+1,n}(\delta_r)\right]V_{b,n}(p+\delta_r)\\ &= \sum_{r=1}^{k^2}\widetilde{K}_{b,n,p}(\delta_r)V_{b,n}(p+\delta_r).\end{aligned} \tag{22}$$

The identity follows from linearity in the kernel weights, so a separate kernel tensor need not be materialized at every location. It holds conditional on an input and its predicted controls; the complete operator remains input-dependent. An additional nonlinear normalization of $\widetilde{K}_{b,n,p}$ would generally change this identity.

Relation to direct evaluation. The normalized trajectory $K_n(s)$ can be nonlinear in s, so $\widetilde{K}_{b,n,p}$ generally differs from $K_n(s_b(p))$. The responses coincide at support locations and on intervals where the normalized trajectory is affine. Between supports, the implemented readout is a piecewise-linear approximation. Figure 4 measures the observed approximation error, and Appendix A.3 gives a conditional error bound.

A.3 Interpolation Error and Scale Sensitivity

Kernel and response error. Fix an operator, its parameters, and its reference $\mu_\alpha > 0$. On a support interval $[s_j, s_{j+1}]$, let $h_j = s_{j+1} - s_j$ and $k_n(s) = \mathrm{vec}(K_n(s))$. Suppose the normalized trajectory is twice continuously differentiable and satisfies $\sup_{s\in[s_j,s_{j+1}]} \|k_n''(s)\|_2 \le L_{j,n}$. Its linear interpolant obeys

$$\|k_n(s) - \widetilde{k}_n(s)\|_2 \le \frac{L_{j,n}}{2}(s-s_j)(s_{j+1}-s) \le \frac{L_{j,n}h_j^2}{8}. \tag{23}$$

For completeness, define $g_s(t) = (t-s_j)(s_{j+1}-s)/h_j$ for $t \le s$, and $g_s(t) = (s-s_j)(s_{j+1}-t)/h_j$ for $t \ge s$. The interpolation remainder is $-\int_{s_j}^{s_{j+1}} g_s(t)k_n''(t)\,dt$. Since $g_s \ge 0$ and its integral is $(s-s_j)(s_{j+1}-s)/2$, taking norms gives Equation (23).

Let $v_{b,n,p} \in \mathbb{R}^{k^2}$ collect one projected feature patch. The Cauchy–Schwarz inequality gives

$$|R^{\star}_{b,n}(p) - \widetilde{R}_{b,n}(p)| \le \frac{L_{j,n}h_j^2}{8}\|v_{b,n,p}\|_2. \tag{24}$$

With the same mixing weights and output projection, let $e_n = \widetilde{R}_{b,n}(p) - R^{\star}_{b,n}(p)$. The output difference is $W^{\mathrm{out}}\,\mathrm{diag}(m_b(p,:))e$, whose norm is at most $\|W^{\mathrm{out}}\|_2\|e\|_2$, because $0 \le m_b(p,n) \le 1$.

Uniform absolute supports give $h_j = (\alpha_{\max} - \alpha_{\min})/((M-1)\mu_\alpha)$. For fixed parameters and a uniform curvature bound, the upper bound therefore scales as $O((M-1)^{-2})$. The assumption concerns the normalized trajectory, whose curvature also depends on the spatial normalization.

Local scale sensitivity. Holding the response bank and reference fixed, the derivative inside a support interval is

$$\frac{\partial \widetilde{R}_{b,n}(p)}{\partial \alpha_b(p)} = \frac{R_{b,j+1,n}(p) - R_{b,j,n}(p)}{\alpha_{j+1} - \alpha_j}. \tag{25}$$

The scale controller thus receives an interpolation gradient whenever neighboring responses differ. The readout is continuous at support points, although its slope can change; clamping gives zero interpolation gradient outside the support range. This is a conditional readout derivative, distinct from the full derivative with respect to X. The stop-gradient EMA update likewise does not prevent the generator from receiving gradients through the support convolutions.

A.4 Low-Rank Structure and Computational Cost

Conditional channel factorization. For a fixed input, position, and offset, Equation (17) can be written as

$$\begin{aligned} W^{\mathrm{eff}}_{b,p}(\delta) &= W^{\mathrm{out}} D_{b,p}(\delta) W^{\mathrm{in}}, \\ D_{b,p}(\delta) &= \mathrm{diag}\Big(m_b(p,n)\widetilde{K}_{b,n,p}(\delta)\Big)_{n=1}^{N}. \end{aligned} \tag{26}$$

Consequently, $\mathrm{rank}(W^{\mathrm{eff}}_{b,p}(\delta)) \le \min(C_{\mathrm{in}}, C_{\mathrm{out}}, N)$. The spatially varying kernel acts through shared channel factors and an N-dimensional latent space.

Filtering, projection, and control costs. Let $S_{\mathrm{in}} = HW$ and $S_{\mathrm{out}} = H_oW_o$. The input projection, support convolutions, and output projection require the following leading multiply–accumulate counts:

$$BS_{\mathrm{in}}C_{\mathrm{in}}N + BS_{\mathrm{out}}MNk^2 + BS_{\mathrm{out}}NC_{\mathrm{out}}. \tag{27}$$

Interpolation and content weighting add $O(BS_{\mathrm{out}}N)$ operations. The generator, controllers, normalization, activations, and resizing contribute additional costs. For example, if both controllers' two 3×3 convolutions operate at input resolution with hidden width d, their convolutional terms together require

$$BS_{\mathrm{in}}\left(18C_{\mathrm{in}}d + 18d^2 + d(N+1)\right) \tag{28}$$

multiply–accumulate operations, excluding normalization and activations.

Kernel reuse and response storage. The shared generator is queried at Mk^2 coordinates per operator bank, returning all N basis values at each query. Direct evaluation at every output location would instead require $BS_{\mathrm{out}}k^2$ coordinate queries. The sampled kernels contain MNk^2 entries, while the explicit response bank contains $BMNS_{\mathrm{out}}$ entries. Thus, response-bank computation and storage both grow linearly with M.

B EXPERIMENTAL DETAILS

This appendix supplements Section 4.1 with the reported architecture, compute environment, and task-specific recipes. For the reproduced downstream comparisons, the task head, data pipeline, schedule, and evaluation protocol are shared within each benchmark. The ImageNet comparison retains the reference-comparison scope stated in Table 1, where training recipes differ across entries.

B.1 ARCHITECTURE AND CORE-KAN CONFIGURATION

Backbone integration. Core-KAN is integrated into ResNet-50 by replacing the 3×3 bottleneck convolution (`conv2`) at selected locations in each of the four residual stages. The stem, residual downsampling paths, normalization layers, and classification head retain the standard ResNet-50 design. Detection and segmentation use the corresponding ImageNet-1K pretrained backbone, with the feature pyramid, where applicable, and task-specific heads otherwise unchanged.

Shared operator settings. The default operator uses a fixed 3×3 sampling grid, $N = 16$ bases, $M = 8$ uniformly spaced absolute supports, and $[\alpha_{\min}, \alpha_{\max}] = [0.5, 2.0]$. The scale reference has EMA momentum $\rho = 0.99$ and is fixed at inference. Each controller has hidden width 32. The two-layer generator has hidden width 32, spline grid size 7, and spline order 3. These defaults are shared across the three tasks. Appendix A.1 describes the generator, and Appendix A.2 specifies the interpolation readout.

B.2 COMPUTE ENVIRONMENT AND REPORTING

The reported training runs use one node with eight NVIDIA A100 GPUs (40 GB each), with one distributed process per GPU. The ImageNet-1K configuration uses random seed 0 with distributed-rank offsets. Results are point estimates. The instance-level and controller analyses in Appendix C characterize the evaluated predictions and representations.

B.3 IMAGENET-1K CLASSIFICATION

Optimization. Core-KAN is trained for 300 epochs with AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.05. The global batch size is 4,096. The base learning rate of 5×10^{-4} at batch size 512 is linearly scaled to 4×10^{-3}. Training uses 20 warm-up epochs, cosine decay, gradient clipping at norm 5, and float16 automatic mixed precision.

Augmentation and evaluation. Training uses 224×224 images with bicubic interpolation, color jitter 0.4, RandAugment (`rand-m9-mstd0.5-inc1`), random erasing probability 0.25, Mixup parameter 0.8, CutMix parameter 1.0, and label smoothing 0.1. Batch-level mixing is applied with probability 1.0, with a Mixup/CutMix switch probability of 0.5. Validation uses a single 224×224 center crop and reports top-1 and top-5 accuracy as in Table 1.

B.4 COCO DETECTION AND INSTANCE SEGMENTATION

Task model and input processing. COCO experiments use Mask R-CNN with FPN and the corresponding ImageNet-1K pretrained backbone. The stem and first residual stage are frozen; batch-normalization statistics remain in evaluation mode during fine-tuning. Images use ImageNet normalization and are padded to a multiple of 32. Each GPU processes two images, for a global batch size of 16.

Schedules and augmentation. The $1\times$ configuration trains for 12 epochs. Images retain their aspect ratio when resized to fit within 1333×800 and are horizontally flipped with probability 0.5. The $3\times$ configuration trains for 36 epochs with short-side targets from 480 to 800 in increments of 32. An alternative augmentation branch first resizes the short side to 400, 500, or 600, applies a 384×600 random crop, and then performs the final multi-scale resize.

Optimization and checkpoint selection. Both configurations use SGD with learning rate 0.02, momentum 0.9, weight decay 10^{-4}, and gradient clipping at ℓ_2 norm 5. Linear warm-up lasts

500 iterations and starts at 10^{-3} of the target learning rate. The learning rate is multiplied by 0.1 at epochs 8 and 11 for $1\times$, and at epochs 27 and 33 for $3\times$. Official box and mask metrics are evaluated on `val2017` after every epoch, with checkpoint selection based on box mAP. Tables 2 and 3 report the corresponding task metrics; Appendix C defines the separate descriptive diagnostics.

B.5 ADE20K Semantic Segmentation

Optimization and validation. ADE20K experiments use the ResNet-50 backbone and synchronized batch normalization. The training configuration specifies 24 crops. Training lasts 160,000 iterations with SGD, learning rate 0.01, momentum 0.9, and weight decay 5×10^{-4}. Linear warm-up lasts 1,500 iterations and starts at 10^{-6} of the target learning rate, followed by polynomial decay with power 1.0 to zero. Validation occurs every 16,000 iterations, and the checkpoint with the highest mIoU is selected.

Augmentation and inference. Training uses 512×512 crops. Images are resized with a random ratio in $[0.5, 2.0]$ relative to a 2048×512 reference, cropped with maximum single-category occupancy 0.75, horizontally flipped with probability 0.5, and augmented with photometric distortion. Evaluation preserves aspect ratio at the same reference scale and uses whole-image inference without test-time flipping. Table 4 reports validation mIoU and mean class accuracy; its parameter count covers the complete segmentation model.

C COCO Localization and Controller Diagnostics

The COCO results in Section 4.2 summarize task performance across the evaluation set. We complement them with two analyses: localization gains for ground-truth instance groups, and controller behavior at different distances from annotated boundaries. The first concerns predictions, while the second examines the scale and mixing outputs defined in Section 3.2. Figure 6 brings these measurements together without treating them as the same diagnostic.

C.1 Ground-Truth Partitions and Localization Metrics

Instance attributes. For instance i, let A_i, P_i, and A_i^{img} denote its ground-truth mask area, mask perimeter, and image area. Relative-area quartiles Q1–Q4 partition $a_i = A_i/A_i^{\mathrm{img}}$ in ascending order. They describe image occupancy and are distinct from the official COCO small, medium, and large categories. Contour-complexity quartiles B1–B4 instead use

$$C_i = \frac{P_i^2}{4\pi A_i}. \tag{29}$$

At a fixed area, a longer perimeter increases C_i. For an ideal planar shape the ratio is invariant to uniform scaling, although rasterized measurements also depend on resolution and the perimeter estimator. The reported partitions contain approximately 9,084 instances per quartile. Q and B are separate marginal partitions of the same set, so they do not independently control size, contour complexity, and category composition.

Per-instance localization. Let g_i and c_i be the ground-truth box and category. For model f, let $\mathcal{D}^{(f)}_{I_i,c_i}$ contain its retained boxes of category c_i in the same image. The best-matched box score is

$$u_i^{(f)} = \max\Big(\{\mathrm{IoU}_{\mathrm{box}}(g_i, d) : d \in \mathcal{D}^{(f)}_{I_i,c_i}\} \cup \{0\}\Big). \tag{30}$$

The zero gives no localization credit when no same-class prediction remains. This per-instance maximum allows a prediction to be considered for more than one ground-truth instance; it is not a global one-to-one assignment. It measures the quality of available localization candidates rather than directly penalizing all false positives.

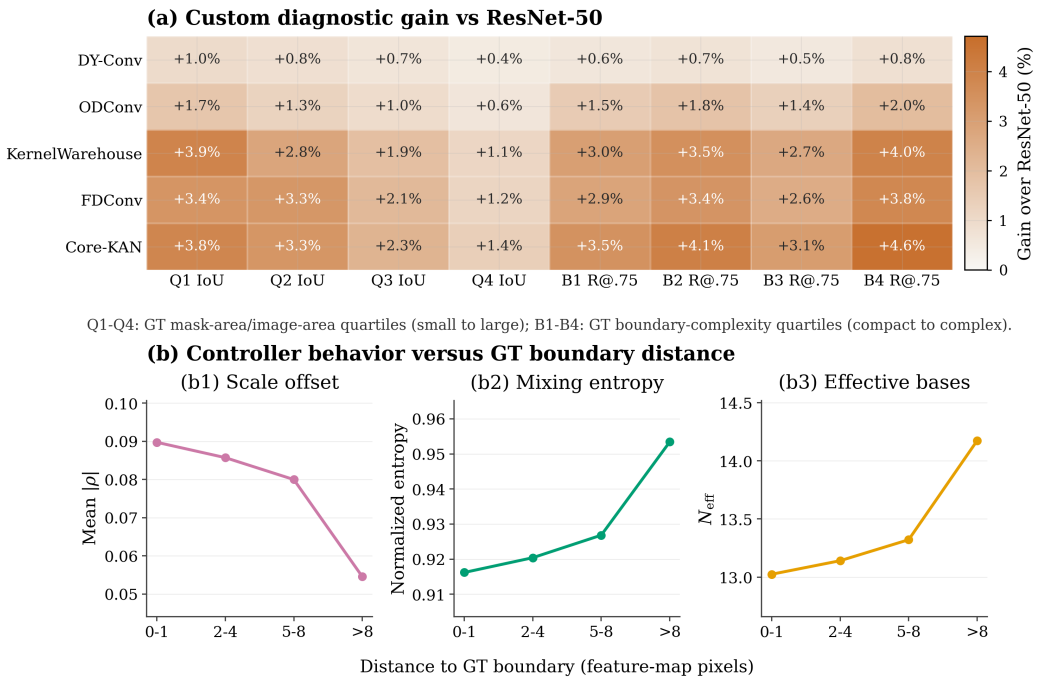

Figure 6: COCO localization and controller diagnostics. (a) Gains over ResNet-50 in mean best-matched same-class box IoU for relative-area quartiles Q1–Q4, and GT-instance box Recall@0.75 for contour-complexity quartiles B1–B4. All annotated gains are percentage-point differences, although the two column groups use different metrics. (b) Stage 3 controller summaries by distance to a ground-truth boundary at feature-map resolution: mean absolute log-relative scale offset, normalized mixing entropy, and effective number of bases. The symbol ρ in panel (b1) denotes $\delta_\alpha = \log(\alpha/\mu_\alpha)$.

For relative-area group Q_q and contour group B_r, respectively, the two diagnostics are

$$\begin{aligned} J_{Q_q}^{(f)} &= \frac{1}{|Q_q|} \sum_{i \in Q_q} u_i^{(f)}, \\ R_{B_r}^{(f)}(0.75) &= \frac{1}{|B_r|} \sum_{i \in B_r} \mathbf{1}\left[u_i^{(f)} \geq 0.75\right]. \end{aligned} \tag{31}$$

The first averages box IoU; the second measures the fraction of instances with a same-class box meeting a fixed localization threshold. The B groups are defined from masks, but their reported score still evaluates boxes, not predicted-mask boundaries. For either metric $T \in [0, 1]$, the plotted gain is $100(T^{(f)} - T^{(\text{ResNet-50})})$ percentage points.

C.2 Controller Statistics

For readability, omit the batch index. We compute the log-relative scale offset and two summaries of the simplex-normalized mixing weights:

$$\begin{aligned} \delta_\alpha(p) &= \log\left(\frac{\alpha(p)}{\mu_\alpha}\right) = \log s(p), \\ H_{\text{norm}}(p) &= -\frac{1}{\log N} \sum_{n=1}^{N} m(p,n) \log m(p,n), \\ N_{\text{eff}}(p) &= \exp\left(-\sum_{n=1}^{N} m(p,n) \log m(p,n)\right). \end{aligned} \tag{32}$$

Zero log-relative offset corresponds to the stored scale reference. Its absolute value measures deviation magnitude and discards direction, so a larger $|\delta_\alpha|$ does not imply a larger s. As in Section 3.3, this scale changes the queried kernel profile on a fixed sampling grid, rather than directly specifying support size or receptive-field extent.

The analysis uses Stage 3 outputs at their native resolution and groups locations into the reported boundary-distance bands 0–1, 2–4, 5–8, and more than 8 feature-map pixels. From the closest to the farthest band, mean absolute log-relative scale offset decreases from 0.0897 to 0.0546, normalized entropy increases from 0.9162 to 0.9534, and effective count increases from 13.02 to 14.17 out of $N = 16$. Locations closer to boundaries therefore show larger scale deviations and modestly more concentrated mixtures.

C.3 Instance-Level Findings

Relative-area groups. Core-KAN improves mean best-matched box IoU over ResNet-50 by 3.8, 3.3, 2.3, and 1.4 percentage points across Q1–Q4. At the displayed precision, it ties FDConv in Q2 and leads Q3 and Q4. KernelWarehouse is slightly higher in Q1, with a 3.9-point gain compared with 3.8 points. Thus, localization improves across all four groups, with a larger gain for the smallest relative-area quartile than for the largest, although Core-KAN does not lead every group.

Contour-complexity groups. The Recall@0.75 gains are 3.5, 4.1, 3.1, and 4.6 points across B1–B4. Core-KAN leads each group, exceeding the strongest competing value in the corresponding column by 0.5, 0.6, 0.4, and 0.6 points. The largest gain occurs in B4: more instances in the highest contour-complexity group have a same-class box satisfying the localization threshold. The gains are not monotonic across B1–B4, because B3 improves less than B1 and B2. Across the eight columns, Core-KAN has the highest or joint-highest gain in seven groups. These results complement the aggregate COCO evaluation by showing gains across multiple instance subsets.

C.4 Connection to the Dual-Control Formulation

The two diagnostics connect to different aspects of Core-KAN. Instance-level scores show how prediction gains are distributed across object groups. Boundary-distance statistics characterize where the learned controls vary: nearby locations have larger deviations from the reference scale and somewhat less diffuse basis weights. This combination is consistent with adapting local filter profiles and basis composition to heterogeneous structures.

The connection follows the factorization in Equation (17). Scale predictions select the interpolated spatial kernels, while mixing predictions weight their contributions before the output projection. Both influence the effective filter. Their separate parameterized paths define architectural roles without requiring statistical independence, and the descriptive measurements do not isolate either branch's causal contribution to task performance.

Mixture concentration also differs from computational sparsity. The support-response bank is computed before content weighting, so a smaller N_{eff} does not reduce the number of support convolutions. Reuse comes from the shared supports and interpolation described in Appendix A.4. Together, the diagnostics characterize the evaluated models' localization and controller behavior, while the kernel and interpolation analyses in Section 4.3 examine the continuous representation and its approximation.