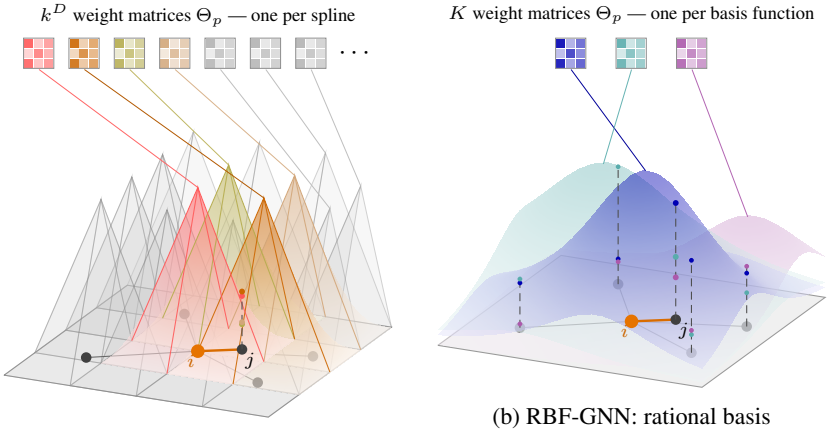
RBF-GNN: RATIONAL BASIS FUNCTIONS FOR PSEUDO-COORDINATE BASED GRAPH CONVOLUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose RBF-GNN, a new pseudo-coordinate based graph neural network architecture that takes into account Euclidean, spherical or angular coordinates and uses them to induce a powerful spatial inductive bias. Similar in architecture to SplineCNN, we improve upon the latter by replacing the less efficient sparse-activation based B-splines whose number grows exponentially with dimension by rational Padé basis functions. For effective training we propose a spline-subspace initialization and a variance-preserving weight rescaling. Experimentally, we evaluate on a number of popular neural network architectures that use SplineCNNs. We replace only the SplineCNNs with RBF-GNN. We achieve improved results, including on semantic keypoint matching, shape matching, event based camera computer vision tasks. We will make our implementation publicly available upon acceptance of the paper.



(a) SplineCNN: B-spline basis

(b) RBF-GNN: rational basis

Figure 1: RBF-GNN replaces the fixed B-spline basis of SplineCNN by a small set of global, learnable rational basis functions. Both panels show the pseudo-coordinate domain $[0, 1]^D$, $D = 2$ with the neighborhood of a node i: each neighbor j sits at its pseudo-coordinate $\mathbf{u}_{ij}$. Every basis function carries one weight matrix Θ_p (small squares), and the dashed stem over the highlighted edge (i, j) marks the basis values that weight its message. **(a)** SplineCNN has one weight matrix for each of the k^D B-splines on the knot grid of dimension D, but an edge activates only the 2^D splines around its pseudo-coordinate (colored); all other splines (faint) and their matrices (gray) receive no signal from this edge. **(b)** RBF-GNN uses K rational safe Padé basis functions (here $K = 3$), smooth, globally supported, learnable, with dense activations and K is independent of D.

1 INTRODUCTION

Many graph convolutions for geometric data define their kernel as a continuous function of pseudo-coordinates attached to the edges, and they differ mainly in how this function is represented. SplineCNN (Fey et al., 2018) expands it in a fixed tensor-product basis of B-splines, MoNet (Monti et al., 2017) in a mixture of learnable Gaussians, and more recent continuous-kernel operators (Ma

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.

et al., 2024) in a small multilayer perceptron of the pseudo-coordinates. This choice determines which kernels a layer can represent and how many parameters it requires.

SplineCNN has been especially powerful and has been adopted in a range of later architectures. Its basis functions are compactly supported B-splines on a regular grid over the unit cube. This gives the kernel an explicit spatial locality and makes the evaluation time independent of the kernel size, because only the few basis functions whose support contains the pseudo-coordinate of an edge have to be evaluated. The same grid, however, also determines the number of parameters. With k basis functions per axis in D pseudo-coordinate dimensions, the grid contains $K = k^D$ basis functions, and the layer has one weight matrix for each of them. For a fixed resolution per axis, the number of weight matrices therefore grows exponentially with D, independently of the complexity of the kernel that has to be learned. Because of the compact support, each edge moreover involves only a small subset of these weight matrices, so that each weight matrix is updated by only a fraction of the edges.

In this work, we replace the B-spline basis by K learnable rational functions, so that K becomes a hyperparameter that is independent of D. Each basis function is a ratio of two polynomials in the safe Padé form $B = P(t)/(1 + |Q(t)|)$ (Molina et al., 2020), whose denominator is at least one and therefore never vanishes, and both polynomials are expressed in the Chebyshev basis for better numerical conditioning. Unlike the B-splines, these functions are not compactly supported, so every edge involves all K weight matrices. Since rational functions with small random coefficients are nearly flat, we initialize the basis from the B-splines it replaces. When $K = k^D$, we fit the B-splines directly. Otherwise, we fit their leading principal components and rescale the kernel weights so that the initial message variance matches that of the B-spline layer.

In our experiments, we obtain performance improvements whenever we replace SplineCNN with RBF-GNN across a wide array of applications and deep learning architectures. Additionally, in many applications we can do so with significantly lower parameter counts.

In summary, our contributions are as follows:

- We propose RBF-GNN, a new pseudo-coordinate based GNN architecture with learnable activation functions. Our formulation is efficient and does not suffer from a blow-up in the number of activations.
- We propose practical ways to make RBF-GNN work, namely a special initialization scheme and a fast kernel implementation.
- We evaluate RBF-GNN across eight benchmarks whose pseudo-coordinate dimension ranges from two to six, spanning keypoint matching on PascalVOC (Everingham et al., 2010) and SPair-71k (Min et al., 2019) in two published matching architectures (Fey et al., 2020; Pourhadi & Swoboda, 2026), shape correspondence on FAUST (Bogo et al., 2014), event-camera recognition on N-Caltech101 (Orchard et al., 2015) and N-Cars (Sironi et al., 2018), superpixel classification (Dwivedi et al., 2023) and point-cloud part segmentation (Yi et al., 2016). We get consistent performance improvements by switching out our closest competitor SplineCNN to RBF-GNN with a matched or sometimes even lower parameter count and similar or lower compute costs.

2 Related work

Geometry-aware graph convolutions. Message-passing neural networks provide the general neighbourhood-aggregation view (Gilmer et al., 2017), while spatial operators additionally condition messages on edge geometry. ECC generates filters from edge attributes (Simonovsky & Komodakis, 2017); MoNet applies Gaussian mixtures to pseudo-coordinates (Monti et al., 2017); and FeaStNet learns feature-dependent assignments to weight matrices (Verma et al., 2018). Continuous filters have also been generated by neural networks for molecules (Schütt et al., 2017), point clouds (Wang et al., 2018; Wu et al., 2019), and general graphs equipped with positional encodings (Ma et al., 2024). Related point-cloud operators use Taylor-polynomial filters (Xu et al., 2018), learnable kernel points (Thomas et al., 2019), geometry-conditioned weight banks (Xu et al., 2021), or relative-position attention (Zhao et al., 2021). SplineCNN instead expands a continuous kernel in compactly supported tensor-product B-splines (Fey et al., 2018), an operator subsequently used for graph matching (Fey et al., 2020) and event streams (Schaefer et al., 2022). RBF-GNN is closest to SplineCNN: it preserves

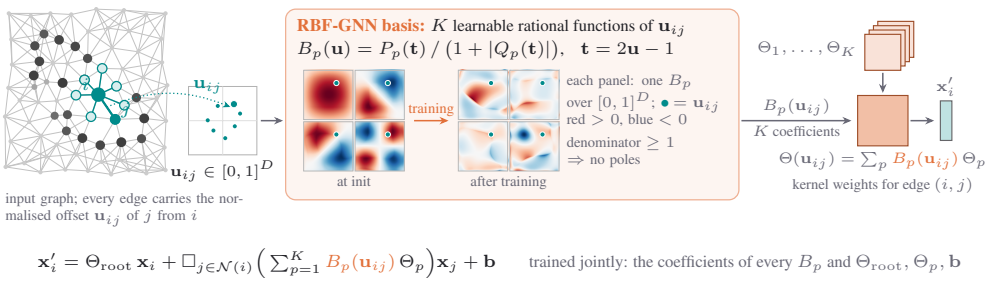


Figure 2: Overview of RBF-GNN. Each graph edge (j, i) carries a normalized pseudo-coordinate $\mathbf{u}_{ij}$. The K learnable safe rational basis functions are evaluated at $\mathbf{u}_{ij}$ and mix the weight matrices Θ_p into the edge-specific continuous kernel $\Theta(\mathbf{u}_{ij})$. The resulting neighbor messages are aggregated to produce $\mathbf{x}'_i$; the rational-basis coefficients and convolution weights are trained jointly.

its message-passing rule and matrix bank, but replaces the fixed grid basis by learnable non-separable rational functions whose number is independent of k^D.

Learned continuous and rational bases. Learnable functional parameterisations include spline time–frequency atoms (Balestriero et al., 2018), neural continuous kernels (Romero et al., 2022b;a), and coordinate networks based on Fourier features, periodic activations, or multiplicative filters (Tancik et al., 2020; Sitzmann et al., 2020; Fathony et al., 2021). Rational functions have been studied as trainable activations (Molina et al., 2020) and as efficient smooth approximators (Boullé et al., 2020). More recently, KAN places learned B-spline functions on network edges (Liu et al., 2025), while KAT replaces those splines by rational functions to improve efficiency and scalability (Yang & Wang, 2025). Rational-activation and KAN-family models modify neuron or feature-mixing nonlinearities, while neural coordinate models learn an entire continuous signal or kernel. In contrast, our rational functions are multivariate mixing coefficients of a matrix-valued graph kernel and are initialised to preserve both the SplineCNN kernel subspace and its output variance.

3 Method

RBF-GNN is a graph convolution whose continuous kernel is spanned by learnable rational basis functions of pseudo-coordinates. We first set up pseudo-coordinates and the continuous-kernel convolution operator, then introduce the rational basis in safe Padé form. We relate our construction to SplineCNN (Fey et al., 2018). Finally, we assemble the full network and describe the initialization that lets it train as reliably as the spline model it can replace. Figure 2 summarizes the complete pipeline.

Graphs and pseudo-coordinates. We consider graphs $G = (V, E)$ embedded in a coordinate space: every node i carries features $\mathbf{x}_i \in \mathbb{R}^{C_{\text{in}}}$ and a spatial position $\mathbf{z}_i \in \mathbb{R}^D$ (2-D image keypoints, vertices of a 3-D mesh, ...). The convolution below depends on *relative* positions. Because the kernel will be parameterised by basis functions living on a *fixed, bounded* domain, the offsets are affinely rescaled into the unit cube: with $\delta_{\max}$ the largest absolute offset component occurring in the graph, the *pseudo-coordinate* of edge (j, i) is

$$\mathbf{u}_{ij} \;=\; \frac{\mathbf{z}_j - \mathbf{z}_i}{2\,\delta_{\max}} \;+\; \tfrac{1}{2}\mathbf{1} \;\in\; [0,1]^D \tag{1}$$

The zero offset lands at the centre $\frac{1}{2}\mathbf{1}$, a neighbour to the left/below has $u_d \leq \frac{1}{2}$ in that dimension, and the largest offset touches the boundary. See Figure 3 for an illustration of the pseudo-coordinate construction. These pseudo-coordinates are the only way geometry enters the model. Polar or spherical encodings can be handled analoguously.

Angular and spherical coordinates: Rescaling periodic coordinates into the unit cube would introduce a seam at $0 \equiv 2\pi$ and, for spherical angles, singularities at the poles. Angular coordinates can be embedded as $(\cos\theta, \sin\theta)$ and directions as unit vectors $\mathbf{n} \in S^2$.

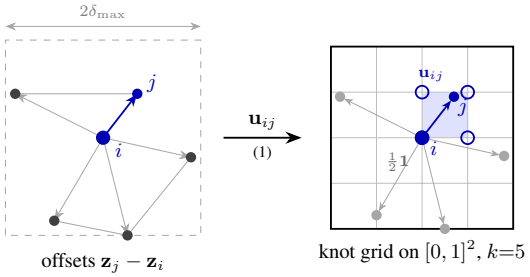


Figure 3: Construction of two dimensional pseudo-coordinates for D=2. **Left:** The convolution at node i represents each neighbour j by its spatial offset $\mathbf{z}_j - \mathbf{z}_i$. The dashed box marks the largest offset magnitude $\delta_{\max}$ in the graph. **Right:** The map in (1) rescales this box to the unit square. Node i lands at the centre $\frac{1}{2}\mathbf{1}$, node j lands at $\mathbf{u}_{ij}$ in blue, and the largest offset reaches the boundary.

Continuous convolution kernels. An image convolution assigns a separate weight matrix to each of the finitely many pixel offsets. On an irregular graph the offsets $\mathbf{u}_{ij}$ vary continuously, so we instead learn a *kernel function* $g : [0,1]^D \to \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ and evaluate it at the observed pseudo-coordinates. The kernel is parameterised as a linear combination $g(\mathbf{u}) = \sum_{p=1}^{K} B_p(\mathbf{u})\, \Theta_p$ of K scalar basis functions $B_p \colon [0,1]^D \to \mathbb{R}$ with learnable weight matrices $\Theta_p \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$, and a layer computes

$$\mathbf{x}'_i \;=\; \Theta_{\text{root}}\, \mathbf{x}_i \;+\; \operatorname*{\square}_{j \in \mathcal{N}(i)} \Big(\sum_{p=1}^{K} B_p(\mathbf{u}_{ij})\, \Theta_p \Big) \mathbf{x}_j \;+\; \mathbf{b}, \tag{2}$$

with a separate root weight for the node itself, a bias, and aggregation $\square \in \{\text{mean}, \text{add}, \max\}$ over the neighbourhood. This operator was introduced by SplineCNN (Fey et al., 2018). The difference lies in the choice of the basis functions B_p, which are fixed B-splines there (see the remark below) and learnable rational functions for RBF-GNN.

Rational safe Padé basis functions. In RBF-GNN the scalar basis functions $B_p \colon [0,1]^D \to \mathbb{R}$ of (2) are learnable rational functions. Each B_p is a quotient of two polynomials whose coefficients are network parameters. We build up the functional form in one dimension first. A *polynomial* of degree m with learnable coefficients $a_0, \ldots, a_m$ is

$$P(t) \;=\; \sum_{j=0}^{m} a_j\, t^j. \tag{3}$$

A *rational* function of degrees (m, n) is a quotient of two such polynomials,

$$R(t) \;=\; \frac{P(t)}{Q(t)} \;=\; \frac{\sum_{j=0}^{m} a_j\, t^j}{\sum_{j=0}^{n} b_j\, t^j}. \tag{4}$$

Training a free denominator is unstable, since it can become zero, making the evaluation undefined, but also being close to zero will lead to numerical overflow. We therefore use the *safe* Padé form (Molina et al., 2020), in which the denominator polynomial Q (without constant term) enters only through $1 + |Q(\cdot)| \geq 1$:

$$B(t) \;=\; \frac{P(t)}{1 + \big|Q(t)\big|} \;=\; \frac{\sum_{j=0}^{m} a_j\, t^j}{1 + \Big|\sum_{j=1}^{n} b_j\, t^j\Big|}, \tag{5}$$

which remains well defined and bounded for every value of the learnt coefficients a_j, b_j. Figure 4 contrasts the plain rational (4) with the safe form (5).

In practice we expand P and Q not in the monomial basis $1, t, t^2, \ldots$ but in Chebyshev polynomials of the first kind $\phi_0, \phi_1, \ldots$ (ϕ_0=1, ϕ_1=t, ϕ_{j+1}=$2t\phi_j - \phi_{j-1}$), i.e. $P(t) = \sum_j a_j\, \phi_j(t)$. This parametrization spans the same function class but is better conditioned on bounded domains. The map from monomial coefficients to polynomial values on an interval is exponentially ill-conditioned in the degree (Gautschi, 1979), so with monomials a small step in a high-order coefficient barely moves the function near $t = 0$ but changes it violently near the boundary, giving gradients on wildly different scales. The Chebyshev polynomials, by contrast, are orthogonal and uniformly bounded, $|\phi_j(t)| \leq 1$ on $[-1, 1]$, so every coefficient acts on a comparable scale and $|a_j|$ directly bounds the contribution of the j-th term.

Below we extend this one-dimensional basis function to multiple dimensions in two different ways.

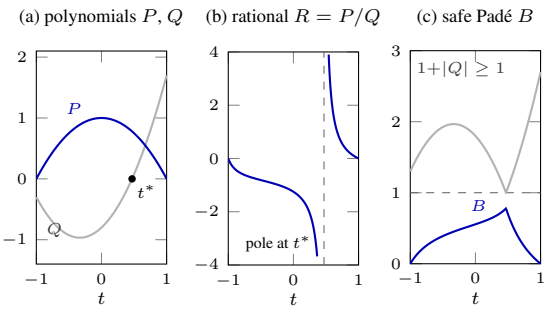


Figure 4: From polynomials to safe Padé basis functions, Eqs. (3)–(5), with the same numerator P and denominator Q in all three panels. **(a)** Two polynomials with learnable coefficients; Q has a root t^* inside the domain. **(b)** The plain rational function $R = P/Q$ diverges at t^*, and any root of Q can move into the domain during training. **(c)** In the safe form the denominator $1 + |Q|$ (gray) never drops below one (dashed), so B stays bounded and well defined for every setting of the coefficients; where Q vanishes, B simply equals P.

Product rational basis. Each of the K basis functions factorises over the coordinate dimensions into univariate safe Padé functions (5),

$$B_{\mathbf{p}}(\mathbf{u}) \;=\; \prod_{d=1}^{D} \frac{P_{d,p}(u_d)}{1 + \big|Q_{d,p}(u_d)\big|}, \tag{6}$$

where every factor has its own learnable polynomials $P_{d,p}$ and $Q_{d,p}$.

Multivariate rational basis. Each basis function $B_p \colon [0,1]^D \to \mathbb{R}$ is a single non-separable rational function of all coordinates jointly:

$$B_p(\mathbf{u}) = \frac{P_p(\mathbf{u})}{1 + |Q_p(\mathbf{u})|}, \quad P_p(\mathbf{u}) = \sum_{|\alpha| \leq m} a_{p,\alpha}\, u_1^{\alpha_1} \cdots u_D^{\alpha_D}, \quad Q_p(\mathbf{u}) = \sum_{1 \leq |\alpha| \leq n} b_{p,\alpha}\, u_1^{\alpha_1} \cdots u_D^{\alpha_D}, \tag{7}$$

with multi-indices $\alpha \in \mathbb{N}_0^D$ of total degree $|\alpha| \leq m$ and $|\alpha| \leq n$. As in the univariate case, the implementation uses the Chebyshev parametrization. Each monomial $u_1^{\alpha_1} \cdots u_D^{\alpha_D}$ is replaced by the product $\phi_{\alpha_1}(t_1) \cdots \phi_{\alpha_D}(t_D)$, spanning the same function class in better-conditioned form.

Original SplineCNN and B-spline basis. The original SplineCNN Fey et al. (2018) uses the operator (2), but chooses the basis functions B_p as B-splines, see Figure 1. This construction yields spatial localisation and a strong geometric inductive bias. However, in every coordinate at most 2 B-splines are active, which means in larger dimensions that all but 2^D of the $\Theta_{\mathbf{p}}$ receive zero gradient. This results in sparse, uneven updates and dense evaluation wasted on zeros. The basis size $K = k^D$ grows exponentially with the coordinate dimension D

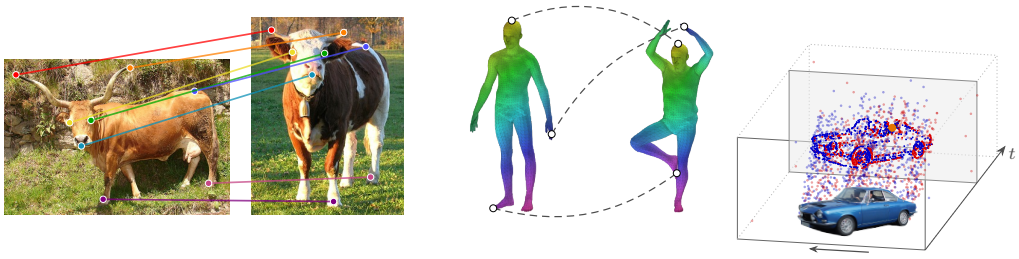
Initialization. Randomly initializing the basis functions works consistently worse. We observe -0.7 to -5.8 drops in accuracy, see the ablation in our experiments. Since we use small random numbers for initialization the resulting basis functions are approximately flat. First, spatial localization would need to be completely learned from scratch. Second, near-identical basis activations give the different Θ_p strongly correlated gradients. Therefore, we propose two initializations that allow for spatial localization from the beginning.

Spline fit. Fit each rational function to its B-spline by least squares so the layer starts as approximately SplineConv. We perform a stage-wise optimization to fit our learnable basis functions as follows:

1. We fit the numerator P with a closed-form least squares fit.
2. Next, we refine both P and Q with Adam (lr 0.01) for 500 steps.
3. Last, for further numerical accuracy we use L-BFGS for 100 iterations on float64.

We evaluate squared loss on a fixed grid of points, using 256 for 1D, 32^2 for 2D and 16^2 on 3D. We use small values for the denominator Q since the gradient of a constant zero Q would vanish. The fit is cached, so stacked layers share the same fit. The spline fit can be performed whenever the number of basis functions $K = k^d$ for some $k \in \mathbb{N}$.

PCA of splines. When $K \neq k^D$, we cannot fit to the splines. Instead, we fit the K functions to the K leading principal components of the k^D spline basis. In detail, this is done as follows: We



(a) semantic keypoint matching (b) shape correspondence (c) event-based vision

Figure 5: Tasks with pseudo-coordinate graphs used in our experiments. **(a)** Semantic keypoint matching on SPair-71k: keypoints of two images of the same category are graph nodes, and corresponding keypoints (same color, 8 of 20 shown) are matched across instances with different viewpoint and appearance. **(b)** Shape correspondence on FAUST: every vertex of a triangulated human scan is matched to its vertex on another person in a different pose; colors encode the ground-truth correspondence, and dashed arcs show example matches. **(c)** Event-based vision: object moving through a scene triggers a stream of events. The events of a short time window (gray plane) trace the contour of the car; each event is a graph node (orange) connected to its space-time neighbors, and the pseudo-coordinates are relative (x, y, t) offsets.

sample points on the same grid as above. We collect the values in a matrix, compute its SVD and the K leading left singular vectors $u_1, \dots, u_K$. We rescale each u_i to unit maximum and fix to a deterministic sign. Last, we fit each u_i with the same procedure as above for the spline fit.

Variance-preserving gain. The initialization of the kernel weights is calibrated for the spline basis. The entries of Θ_p are drawn i.i.d. zero-mean with variance proportional to $1/(KC_{\text{in}})$. Thus, the message $\sum_p B_p(\mathbf{u})\,\Theta_p\,\mathbf{x}_j$ along an edge has variance proportional to the basis energy $\sum_p B_p(\mathbf{u})^2$. For partition-of-unity basis like the spline the variance remains. However, the PCA-based initialization can result in larger activations and, without correction, to inflated output scale.

Adapting a technique from KAT (Yang & Wang, 2025), we therefore rescale the kernel weights by a suitable value $\sqrt{\alpha}$ after basis initialization. The gain compares the energy of the B-spline basis being replaced with that of the initialized rational basis,

$$\alpha \;=\; \mathbb{E}_{\mathbf{u}}\Big[\sum_{\mathbf{p}} N_{\mathbf{p}}(\mathbf{u})^2\Big] \Big/ \; \mathbb{E}_{\mathbf{u}}\Big[\sum_{p} B_p(\mathbf{u})^2\Big], \tag{8}$$

Table 1: Semantic keypoint matching on PascalVOC and SPair-71k. Accuracy (%) follows the protocol of each pipeline and is the mean ± std over 5 seeds, and best, second, and third mark the three highest means per column. Parameter counts exclude the VGG16 backbone and cover the whole DGMC model but only the NMT graph-convolution layers, which are larger because they use 648 channels instead of the 256 and 128 of DGMC. All other columns report the best test epoch.

	DGMC (Fey et al., 2020)			NMT (Pourhadi & Swoboda, 2026)		
Method	Params ↓	PascalVOC ↑	SPair-71k ↑	Params ↓	PascalVOC ↑	SPair-71k ↑
Spline baselines						
SplineCNN (K=25)	9.50M	73.00 ± 0.40	76.18 ± 0.64	28.17M	83.10 ± 0.36	82.10 ± 0.11
SplineCNN (K=9)	3.74M	72.78 ± 0.40	76.04 ± 0.43	10.84M	82.66 ± 0.30	81.54 ± 0.31
SplineCNN (K=4)	1.93M	71.48 ± 0.40	71.92 ± 0.38	5.42M	80.10 ± 0.46	77.79 ± 0.40
MLP learned-basis control						
MLP basis (K=4)	1.94M	72.76 ± 0.27	73.74 ± 0.71	5.42M	81.90 ± 0.27	80.40 ± 0.57
MLP basis (K=9)	3.74M	72.04 ± 0.73	74.34 ± 0.62	10.84M	82.12 ± 0.51	81.05 ± 0.32
RBF-GNN with a product basis						
RBF-GNN (K=25)	9.50M	72.88 ± 0.48	76.34 ± 0.42	28.17M	83.78 ± 0.35	82.08 ± 0.21
RBF-GNN (K=9)	3.74M	73.26 ± 0.18	76.24 ± 0.73	10.84M	82.85 ± 0.42	81.64 ± 0.35
RBF-GNN with a multivariate basis						
RBF-GNN (K=9), spline init	3.74M	74.12 ± 0.48	77.68 ± 0.35	10.84M	83.08 ± 0.26	82.11 ± 0.35
RBF-GNN (K=4), PCA+vp	1.94M	73.78 ± 0.41	77.44 ± 0.36	5.42M	82.42 ± 0.43	81.52 ± 0.22
RBF-GNN (K=6), PCA+vp	2.66M	74.06 ± 0.37	77.80 ± 0.66	7.59M	82.95 ± 0.42	81.71 ± 0.18
RBF-GNN (K=9), PCA+vp	3.74M	73.72 ± 0.19	77.50 ± 0.27	10.84M	82.88 ± 0.38	81.59 ± 0.14
RBF-GNN with a multivariate basis, degree control						
RBF-GNN (K=25), degrees $(5, 4)$	9.51M	73.08 ± 0.25	76.26 ± 0.23	28.17M	83.35 ± 0.38	82.01 ± 0.21

Table 2: Accuracy (%) for shape correspondence on FAUST. We could not fully reproduce the published numbers.

Method	Params ↓	Final ↑
Spline baselines		
SplineCNN (K=125), published	4.10M	*99.20*
SplineCNN (K=125), mean + clip	4.11M	98.71 ± 0.08
SplineCNN (K=125), add + clip	4.11M	97.23 ± 0.51
SplineCNN (K=125), add, no clip	4.11M	29.1 ± 50.3
SplineCNN (K=27)	2.30M	83.6 ± 15.6
SplineCNN (K=8)	1.95M	3.4 ± 3.3
MLP learned-basis control		
MLP basis (K=8)	1.96M	98.21 ± 1.11
RBF-GNN with a product basis		
RBF-GNN (K=27)	2.30M	99.24 ± 0.08
RBF-GNN with a multivariate basis		
RBF-GNN (K=4)	1.89M	99.38 ± 0.09
RBF-GNN (K=4), mean aggr.	1.89M	98.93 ± 0.10
RBF-GNN (K=8)	1.97M	99.41 ± 0.04

Table 3: Instance-average mIoU (%) for part segmentation on ShapeNet-Part with 6-D position and normal pseudo-coordinates.

Method	Params ↓	Final mIoU ↑
Spline baselines		
SplineCNN (K=729)	1.60M	65.12 ± 0.41
SplineCNN (K=64)	177.91k	60.02 ± 0.23
MLP learned-basis control		
MLP basis (K=8)	60.75k	62.80 ± 0.45
MLP basis (K=32)	116.88k	63.81 ± 0.46
MLP basis (K=64)	191.73k	64.41 ± 0.40
MLP basis (K=128)	341.43k	63.78 ± 0.17
MLP basis (K=256)	640.82k	64.05 ± 0.75
RBF-GNN with a multivariate basis		
RBF-GNN (K=8)	64.87k	61.88 ± 0.38
RBF-GNN (K=32)	137.43k	64.86 ± 0.18
RBF-GNN (K=64)	234.16k	64.68 ± 0.55
RBF-GNN (K=128)	427.63k	64.94 ± 0.70
RBF-GNN (K=256)	814.58k	64.69 ± 0.41

where $N_{\mathbf{p}}$ are the k^D tensor-product B-splines, B_p are our K rational basis functions as returned by the spline or PCA fit above, and both expectations are over uniform $\mathbf{u}$, evaluated on the fit grid. The initial message variance then matches that of SplineConv with the same K.

Efficient implementation. A naive implementation of RBF-GNN materialises the Chebyshev feature tensor before contracting it with the coefficients. More efficiently, we provide a triton kernel that never materializes this intermediate tensor that directly produces only K basis values from the D pseudo-coordinates. The polynomial sums are efficiently computed by a Horner-like scheme. The resulting kernel runs about 15 times faster than the naive PyTorch implementation and 3 faster than a compiled variant and consumes less memory. Training and inference with the Triton kernel for RBF-GNN is slightly faster overall than the PyG-implementation of SplineCNN.

4 Experiments

We evaluate RBF-GNN on eight benchmarks from five tasks, with pseudo-coordinates of two to six dimensions. In each experiment, RBF-GNN replaces only the SplineConv layers of the network. The tables compare the following methods.

SplineCNN: Fixed B-spline basis with $K = k^D$ functions (Fey et al., 2018).

MLP: Learned basis from a two-layer MLP (Schütt et al., 2017; Wu et al., 2019).

PointNet: PointNet-style layer on neighbor features and pseudo-coordinates, without a basis (Qi et al., 2017).

RBF-GNN with product basis: Products of univariate safe Padé functions (6).

RBF-GNN with multivariate basis: Non-separable rational functions of all coordinates (7).

Spline init and PCA+vp denote the spline fit and the variance-preserving PCA initialization of Section 3. Results are means and standard deviations over three seeds unless stated otherwise.

Semantic keypoint matching. We evaluate semantic keypoint matching in two host pipelines, DGMC (Fey et al., 2020) and the Normalized Matching Transformer (NMT) (Pourhadi & Swoboda, 2026), which both use a continuous-kernel graph convolution for geometric feature refinement. In each pipeline we replace only the SplineConv layers and keep the rest of the architecture, the backbone and the training procedure unchanged. Table 1 reports all configurations for both pipelines on PascalVOC-Keypoints (Everingham et al., 2010; Bourdev & Malik, 2009) and SPair-71k (Min et al., 2019).

In Table 1, RBF-GNN achieves the highest accuracy for both pipelines on both datasets. The multivariate basis is best under DGMC on both datasets and, by 0.01, under NMT on SPair-71k, and the product basis is best under NMT on PascalVOC with 83.78. The margin over the best spline baseline decreases from +1.12 and +1.62 under DGMC to +0.68 and +0.01 under the stronger NMT

Table 4: Test accuracy (%) for object recognition on N-Caltech101 with AEGNN.

Method	Aggr.	Params ↓	Final ↑
Spline baselines			
SplineCNN (K=8)	mean	77.68k	47.35 ± 0.15
SplineCNN (K=8)	max	77.68k	56.00 ± 0.92
PointNet learned-kernel control			
PointNet	max	55.68k	54.99 ± 0.84
RBF-GNN with a multivariate basis			
RBF-GNN (K=8)	mean	91.57k	55.05 ± 0.58
RBF-GNN (K=8)	max	91.57k	57.55 ± 0.42

Table 5: Test accuracy (%) for car recognition on N-Cars with AEGNN.

Method	Aggr.	Params ↓	Final ↑
Spline baselines			
SplineCNN (K=8)	mean	26.99k	87.83 ± 0.25
SplineCNN (K=8)	max	26.99k	89.81 ± 0.27
PointNet learned-kernel control			
PointNet	max	4.99k	87.17 ± 0.24
RBF-GNN with a multivariate basis			
RBF-GNN (K=4)	mean	21.10k	90.24 ± 0.39
RBF-GNN (K=8)	mean	40.88k	90.49 ± 0.05
RBF-GNN (K=8)	max	40.88k	91.00 ± 0.43

pipeline, so the accuracy gain depends on the host pipeline. The parameter savings are consistent. In three of the four columns, a rational basis matches or exceeds the widest spline grid with 2.6 to $4.9\times$ fewer parameters, and under NMT on PascalVOC it is within 0.02 points of that grid with $2.6\times$ fewer.

Shape correspondence. Dense shape correspondence on FAUST (Bogo et al., 2014) assigns every vertex of a held-out human mesh to a template vertex. We follow the SplineCNN protocol (Fey et al., 2018), report exact-match accuracy and compare with the published SplineCNN result (Table 2). Small multivariate rational bases achieve the best and most stable results with substantially fewer parameters than SplineCNN.

Event-based recognition. Both event-camera benchmarks use the seven-layer AEGNN recognition network (Schaefer et al., 2022), with one graph node per event and space-time pseudo-coordinates, and we replace only its SplineConv layers. We report test accuracy for multiclass object recognition on N-Caltech101 (Orchard et al., 2015), where we train with augmentation and a validation-based stopping criterion, and for real-world binary car-versus-background recognition on N-Cars (Sironi et al., 2018), where we train for 30 epochs. On N-Caltech101, RBF-GNN performs best with max aggregation and also improves substantially over SplineCNN under matched mean aggregation (Table 4). On N-Cars, RBF-GNN is the most accurate model under both aggregation rules, its mean-aggregation variant outperforms the max-aggregation SplineCNN, and its smaller configuration is more accurate than SplineCNN with fewer parameters (Table 5).

Superpixel graph classification. On MNIST (LeCun et al., 1998) superpixels, we follow the SplineCNN protocol (Fey et al., 2018) and use two continuous-kernel convolution layers on 2-D Cartesian pseudo-coordinates. On CIFAR-10 (Krizhevsky, 2009) superpixels from GNNBenchmark-Dataset (Dwivedi et al., 2023), we use a 5-D bilateral pseudo-coordinate that concatenates the relative 2-D centroid position with the RGB features. We report test accuracy on both datasets. On MNIST, RBF-GNN achieves the three highest accuracies and improves over SplineCNN at both matched and reduced basis sizes (Table 6). Wider bases do not help: K=16 and 25 stay within noise of K=9, and K=49, initialized on a k=7 spline grid, is slightly worse. On CIFAR-10, RBF-GNN attains the best accuracy at K=32 while using fewer parameters than the wider spline grid (Table 7).

Part segmentation. For point-cloud part segmentation on ShapeNet-Part (Yi et al., 2016), we use a fixed subset of 4,000 training and 1,333 test shapes. Relative positions and surface normals define 6-D pseudo-coordinates on a 16-nearest-neighbor graph, and we report instance-average mIoU. RBF-GNN nearly matches the large spline grid with an order-of-magnitude smaller model and outperforms the equal-width MLP control (Table 3). From K=32 on, accuracy plateaus within noise of the 729-function spline grid ($p \geq 0.27$) at 2 to $12\times$ fewer parameters, with the MLP control below it at every width from K=32 on. At the smallest width the MLP control is more accurate, so the advantage of RBF-GNN on this benchmark is parameter efficiency at larger widths, not a gain at every width.

Ablation study. *Basis size.* For the PCA+vp rows of Table 1, accuracy rises from K=1 to K=4 in all four columns, peaks at K=6 for DGMC and changes little beyond it for NMT. At matched

Table 6: Test accuracy (%) for MNIST superpixel classification.

Method	Params ↓	Final ↑
Spline baselines		
SplineCNN (K=25)	88.36k	97.41 ± 0.13
SplineCNN (K=9)	55.08k	96.49 ± 0.12
SplineCNN (K=4)	44.68k	94.83 ± 0.11
MLP learned-basis control		
MLP basis (K=4)	45.59k	97.23 ± 0.11
MLP basis (K=16)	72.11k	97.48 ± 0.07
MLP basis (K=25)	92.00k	97.32 ± 0.03
RBF-GNN with a multivariate basis		
RBF-GNN (K=4), PCA+vp	45.26k	97.79 ± 0.08
RBF-GNN (K=9), PCA+vp	56.38k	98.00 ± 0.16
RBF-GNN (K=9), spline init	56.38k	97.87 ± 0.06
RBF-GNN (K=16), PCA+vp	71.95k	97.80 ± 0.14
RBF-GNN (K=25), PCA+vp	91.96k	97.83 ± 0.11
RBF-GNN (K=49), PCA+vp	145.34k	97.71 ± 0.17

Table 7: Test accuracy (%) with 5-D bilateral pseudo-coordinates on CIFAR-10 superpixels.

Method	Params ↓	Final ↑
Spline baselines		
SplineCNN (K=243)	557.42k	58.73 ± 0.26
SplineCNN (K=32)	105.03k	57.55 ± 0.34
MLP learned-basis control		
MLP basis (K=8)	55.39k	59.67 ± 0.32
MLP basis (K=32)	109.96k	59.02 ± 0.10
MLP basis (K=64)	182.73k	57.94 ± 0.42
MLP basis (K=128)	328.27k	57.79 ± 0.07
RBF-GNN with a multivariate basis		
RBF-GNN (K=4)	46.45k	57.58 ± 0.23
RBF-GNN (K=8)	56.47k	58.97 ± 0.16
RBF-GNN (K=32)	116.62k	59.87 ± 0.40
RBF-GNN (K=64)	196.81k	59.51 ± 0.21
RBF-GNN (K=128)	357.19k	57.83 ± 0.30

K, the advantage over SplineCNN is concentrated at small bases. It ranges from $+0.42$ to $+5.52$ for $K \in \{4, 9\}$, and at K=25 it is negligible, between -0.02 and $+0.16$, except under NMT on PascalVOC with $+0.68$. On FAUST, the multivariate basis degrades sharply, with high variance across seeds, at K=16 and K=27 (Table 2), which points to a fragile PCA initialization rather than insufficient capacity. At 5-D and 6-D, in contrast, RBF-GNN improves up to K=32 and then saturates on ShapeNet-Part and declines on CIFAR-10 (Tables 7 and 3).

Basis form and polynomial degree. At K=9, the multivariate basis with spline init outperforms the product basis in all four columns of Table 1, but with higher polynomial degrees, $(8, 6)$ instead of $(5, 4)$. In the degree control, where both forms use $(5, 4)$ at K=25, they differ by at most 0.43 points. On FAUST, the product basis remains stable at K=27, whereas the multivariate basis degrades sharply (Table 2).

Initialization. The tables compare the two initializations only at K=9, where spline init outperforms PCA+vp by 0.18 to 0.52 points in all four columns of Table 1 and is 0.13 points worse on MNIST (Table 6). The two therefore perform similarly, and PCA+vp is required whenever $K \neq k^D$.

Aggregation. On FAUST, SplineCNN is most accurate with mean aggregation and unstable with add aggregation unless gradients are clipped, whereas RBF-GNN at K=4 is more accurate with add than with mean (Table 2). On both event-camera benchmarks, max aggregation outperforms mean aggregation for SplineCNN and RBF-GNN alike, and since RBF-GNN is more accurate under both rules, its gain is not explained by the choice of aggregation alone (Tables 4 and 5).

Structured and learned bases. In Table 1, the best rational basis in each column exceeds the best MLP control by $+1.36$, $+3.46$, $+1.66$ and $+1.06$ points. Unlike the margin over SplineCNN, this margin does not decrease systematically from DGMC to NMT, which indicates that the rational form itself matters and not only the fact that the basis is learned. On MNIST, the rational parametrization outperforms the equal-width MLP control at every width, whereas on CIFAR-10 and ShapeNet-Part the MLP control is more accurate only at the smallest shared width. On both event-camera benchmarks, the PointNet control uses fewer parameters than RBF-GNN but is less accurate.

5 Conclusion

We have proposed RBF-GNN, a drop-in replacement for splines in SplineCNN (Fey et al., 2018). Experiments show consistent improvements in performance across a wide range of methods where SplineCNN has been used before. We argue that hand-designed activation functions as used in SplineCNN are worse than learnable ones as used in RBF-GNN, which encode spatial relationships more flexibly but still encode an important inductive bias in contrast to a pure MLP basis.

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing, refactoring and debugging the experimental code. We have not used generative AI tools to generate synthetic data, to formulate mathematical claims or to assist with translation, and the writing of proofs as well as qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to polish the language and improve the readability of the manuscript. We have reviewed all AI-assisted work. AI-assisted code was verified and tested for correctness by the authors, and the reported results were checked against the logs of the corresponding runs. The authors take full responsibility for the integrity, accuracy, and entire content of the submission.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work proposes a basis for continuous-kernel graph convolutions and evaluates it on established benchmarks that are available to the research community. We collected no new data and conducted no experiments with human participants. Some of the benchmarks contain images or three-dimensional scans of people. We used them only as distributed by their creators and in accordance with their licences and terms of use, several of which restrict use to non-commercial research, and we made no attempt to identify any individual. The contribution is methodological and not tied to a particular application, so we do not foresee harmful uses beyond those that apply to graph learning methods in general.

REPRODUCIBILITY STATEMENT

An anonymised version of our code is provided as supplementary material and will be released publicly under an open-source licence upon acceptance. Section 3 specifies the rational basis, both initialisation procedures and the variance-preserving rescaling, and Section 4 states the task, the metric and the number of random seeds for every benchmark. All reported results are means and standard deviations over independent runs with different seeds. Apart from the replaced basis, every host architecture keeps its published configuration unless Section 4 states otherwise.

REFERENCES

Randall Balestriero, Romain Cosentino, Hervé Glotin, and Richard Baraniuk. Spline filters for end-to-end deep learning. In *International Conference on Machine Learning (ICML)*, pp. 364–373, 2018.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3794–3801, 2014.

Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 14243–14253, 2020.

Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1365–1372, 2009. doi: 10.1109/ICCV.2009.5459303.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24 (43):1–48, 2023.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88 (2):303–338, 2010. doi: 10.1007/s11263-009-0275-4.

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 869–877, 2018.

Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *International Conference on Learning Representations (ICLR)*, 2020.

Walter Gautschi. The condition of polynomials in power form. *Mathematics of Computation*, 33 (145):343–352, 1979. doi: 10.1090/S0025-5718-1979-0514830-6.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pp. 1263–1272, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

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

Liheng Ma, Soumyasundar Pal, Yitian Zhang, Jiaming Zhou, Yingxue Zhang, and Mark Coates. CKGConv: General graph convolution with continuous kernels. In *International Conference on Machine Learning (ICML)*, pp. 33902–33924, 2024.

Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5115–5124, 2017.

Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015. doi: 10.3389/fnins.2015.00437.

Abtin Pourhadi and Paul Swoboda. Normalized matching transformer. In *Pattern Recognition. International Conference on Pattern Recognition (ICPR)*, volume 16813 of *Lecture Notes in Computer Science*, pp. 339–353. Springer, 2026. doi: 10.1007/978-3-032-31583-0_23.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.

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

Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. AEGNN: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12371–12381, 2022.

Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3693–3702, 2017.

Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. HATS: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1731–1740, 2018.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7462–7473, 2020.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 7537–7547, 2020.

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6411–6420, 2019.

Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-steered graph convolutions for 3D shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2598–2606, 2018.

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 (CVPR)*, pp. 2589–2597, 2018.

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

Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3173–3182, 2021.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision (ECCV)*, pp. 90–105, 2018.

Xingyi Yang and Xinchao Wang. Kolmogorov–Arnold transformer. In *International Conference on Learning Representations (ICLR)*, 2025.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics*, 35(6):210:1–210:12, 2016. doi: 10.1145/2980179.2980238.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268, 2021.