Adaptive Spiral Layers for Efficient 3D Representation Learning on Meshes

Francesca Babiloni[1,2], Matteo Maggioni[1], Thomas Tanay[1], Jiankang Deng[1,2],
Ales Leonardis[1], Stefanos Zafeiriou[2]
[1]Huawei, Noah's Ark Lab [2]Imperial College London
{f.babiloni22, j.deng16, s.zafeiriou}@imperial.ac.uk
{matteo.maggioni, thomas.tanay, ales.leonardis}@huawei.com

Abstract

The success of deep learning models on structured data has generated significant interest in extending their application to non-Euclidean domains. In this work, we introduce a novel intrinsic operator suitable for representation learning on 3D meshes. Our operator is specifically tailored to adapt its behavior to the irregular structure of the underlying graph and effectively utilize its long-range dependencies, while at the same time ensuring computational efficiency and ease of optimization. In particular, inspired by the framework of Spiral Convolution, which extracts and transforms the vertices in the 3D mesh following a local spiral ordering, we propose a general operator that dynamically adjusts the length of the spiral trajectory and the parameters of the transformation for each processed vertex and mesh. Then, we use polyadic decomposition to factorize its dense weight tensor into a sequence of lighter linear layers that separately process features and vertices information, hence significantly reducing the computational complexity without introducing any stringent inductive biases. Notably, we leverage dynamic gating to achieve spatial adaptivity and induce global reasoning with constant time complexity benefitting from an efficient dynamic pooling mechanism based on Summed-Area-tables. Used as a drop-in replacement on existing architectures for shape correspondence our operator significantly improves the performance-efficiency trade-off, and in 3D shape generation with morphable models achieves state-of-the-art performance with a three-fold reduction in the number of parameters required. Project page: https://github.com/Fb2221/DFC

1. Introduction

Convolutional layers have emerged as the de facto standard methodology to effectively capture local similarity on a grid, hence leading to a notable paradigm shift in fields that analyze regular Euclidean data, such as object detection, image classification, speech recognition, and machine translation. Recently, advances in geometric deep learning have sparked interest towards processing non-Euclidean data, however, applying classic neural networks to such kind of data (i.e., 3D mesh or point cloud) still presents significant challenges.

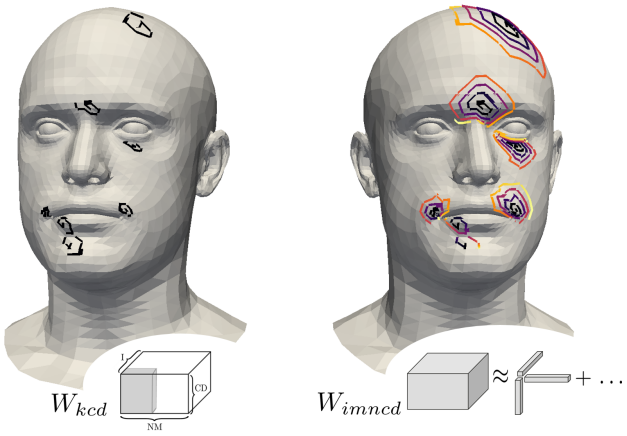


Figure 1: **Visual comparison of SpiralConvolution (left) and our operator (right).** Our proposed operator adapts to the underlying structure of the mesh, learning vertex-adaptive and mesh-adaptive spiral sequences of variable lengths, ranging from 7 to 70. In contrast, the base SpiralConvolution operator works on a fixed receptive field of length 9. The weight tensors of both operators are shown in the image using Einstein notation and a schematic representation. Our proposed operator approximates its dense weight tensor through CP decomposition for efficient implementation. The subscripts indicate respectively: k a fixed sequence length, m a variable sequence length, c, d input and output channels, n and i the number of vertices and meshes considered.

Early deep-learning methods address this issue in an extrinsic manner, applying convolutions directly in the 3D Euclidean space [58, 46], but suffer from high computational complexity and a lack of smoothness in the data represen-

tation. Conversely, more recent methods propose intrinsic methods for non-Euclidean domains [6], and are capable of achieving better results than their extrinsic counterparts. For instance, mesh-based methods, which are tailored the specific topological connectivity of the vertices, typically achieve better performance and lower run times [4, 18, 16, 8]. In contrast, Graph Neural Networks (GNNs) build a more general representation, but tend to reach suboptimal results because they only leverage the graph nature of the meshes [50, 37, 17, 12, 15]. More recently SpiralNet [4] and its extension SpiralNet++ [18] define anisotropic mesh operators that are analogous to 2D convolutions on patches. This is achieved by enforcing a local spiral ordering of the vertices of the mesh, to robustly capture local information. Despite the impressive application of this operator on 3D meshes that share a fixed topology, its expressivity is still constrained by a shared, static, and local set of weights that remains unchanged after training. Consequently, the processing of SpiralNet is independent of the data and task at hand, it is not influenced by the vertex on which it is applied, and it is confined to considering only local regions.

In this work, we propose to build upon the SpiralNet framework by overcoming these limitations, while maintaining the original properties of topology awareness, efficiency, and ease of optimization. We introduce a novel operator for 3D meshes, specifically designed to leverage the irregular structure and long-range dependencies inherently present in the mesh. Guided by the use of Einstein notation, we construct a light-weight mesh-operator by factorizing a dense and general weight tensor via CP decomposition. The factorization is designed to separate the processing of feature and vertex dimensions. Specifically, vertices are processed by promoting two different inductive biases. First, we enable global reasoning by capturing long-range interactions using a dynamic learnable pooling layer implemented as Summed-Area-Tables (SAT) which is capable of efficiently adapting the receptive field depending on the input. Second, we enable robust local reasoning by capturing local information using a gating operator to dynamically adapt the response of our layer to both the input vertex and mesh. The proposed operator focuses on the application of 3D meshes that share a fixed topology, but can be extended to remeshed variants and it is suitable for tasks such as reconstruction, shape correspondence, and data synthesis.

In line with other work designed for 3D meshes on fixed topology [4, 18, 16, 8], we evaluate our methodology on two well-established tasks: namely, 3D shape correspondence and mesh generation through 3D Morphable Models. Our findings demonstrate superior performance on the 3D human shape benchmarks of COMA, FAUST, SCAPE, DFAUST and SYNHAND, quantitatively illustrating best-in-class results. In comparison to SpiralConv, our approach achieves state-of-the-art performance while reducing the parameter count by 64% and 84% respectively demonstrating the efficacy of adaptive processing in practical applications. An overview of our operator and its closest competitor is presented in Figure 1.

2. Related Work

Geometric Deep Learning on Meshes. The field of geometric deep learning [6] has gained important traction in recent years. Here, we briefly review related works analyzing 3D shapes. *Point-based methods* describe 3D shapes in terms of point clouds [41, 42, 48, 32, 19]. These simple and flexible extrinsic approaches however tend to achieve lower performance when compared to mesh-based methods that explicitly leverage the 3D surface connectivity. *Mesh-based methods* can be broadly divided into two categories depending on their definition of convolution: spectral domain-based methods and spatial domain-based methods. Notable spectral domain-based methods view meshes as specific type of graphs that can processed using graph-based neural networks (GNN) such as Chebnet [12] which uses fast localized spectral convolutions and GAT [50] which applies the concept of attention to graph structures. The use of a spectral view-point is also used in state-of-the-art networks of ACSCNN [30] and DiffusionNet [45]. Spatial domain-based methods directly define convolution and pooling operations for triangle meshes. Examples include the early work of [40, 22], MoNet [37] and SplineCNN [15] which use either a Gaussian or B-Spline kernel function for spatial convolutions, FeastNet [51] which establishes connections between filter weights and graph neighborhoods using a dynamic graph-convolution operator. Examples of more recent work are HSN [56] and PFCNN [59] which directly address the rotation ambiguity problem present in 3D meshes, and the state-of-the-art Field Convolution [36]. In the same line of research, and the most relevant prior art for our work, is Spiral Convolution, which applies learnable filters to aggregate a set of neighboring vertices extracted from the mesh following a spiral trajectory (SpiralNet) [4], potentially with a fixed ordering and multiple scales (SpiralNet++) [18]. This serialization encodes information about the topology of the mesh efficiently and effectively. Differently from our method, this operator is still limited by the inductive bias of static and local processing.

Dynamic Methods for Neural Networks. The idea of using a layer whose weights are dependent on the input data can be traced back to early CNNs using max-pooling [24]. Dynamic Networks [21] for Euclidean-data emerged multiple times in the context of 2D Computer Vision [20, 23, 7, 54, 13, 34] as well as in NLP [49, 57, 53, 33]. Dynamicity is also a core component of many notable oper-

ators of geometric deep learning [50, 60, 62, 5], where the importance of vertices in a graph is recalibrated according to their content.In the context of 3D meshes that share a fixed topology, in [61, 55] a differentiable pooling operator is proposed to dynamically group vertices based on their embeddings, in [8] an attention mechanism is adopted to compute vertex feature upsampling and downsampling, and in [16] learnable weighting matrices are used to first resample the neighbors of each vertex and then apply shared anisotropic filters. In contrast, our method is the only one capable to introduce all dynamic components (mesh-adaptivity, vertex-adaptivity, dynamic receptive field) without significantly increasing computational complexity and parameter count.

Tensor Notation for Neural Networks. In machine learning, tensor algebra [27] is commonly used to analyze neural network layers [39] and describe methods that speed up their inference via factorization [10, 29, 38]. Recently, the Einstein notation has gained traction as a practical way to improve code readability [44, 28] and enable efficient tensor calculus [28]. In this work, we use Einstein notation as an intuitive way to highlight tensor characteristics.

3. Background

In this section, we recall the general form of SpiralConvolution, as introduced in [18], and we establish a formalism to describe generic linear operations in neural networks using Einstein notation.

SpiralConvolution. In [4, 18], the authors develop convolution-like operators that extract an ordered set of neighbors around each vertex in the 3D mesh following spiral trajectories [31]. This simple and intuitive serialization can be formally defined as

$$\mathbf{x}_n^{(l)} = \gamma^{(l)} \left(\mathop{||}_{j \in \mathcal{S}(n,K)} \mathbf{x}_j^{(l-1)} \right) \tag{1}$$

where l denotes a layer in the neural network, j and n are two vertices in the mesh, $||$ is a concatenation operator, $\gamma^{(l)}$ is an MLP, $\mathcal{S}(n, K)$ is the operator that creates the spiral sequence of length (*i.e.* kernel size) K corresponding to vertex n, and $\mathbf{x}_j^{(l-1)}$ and $\mathbf{x}_n^{(l)}$ denote the input and output features. In other words, similarly to standard convolutions, this operator generates output features at any given reference position by aggregating a number of local neighbors extracted from a local region centered around the reference position.

Einstein Notation. Following [28], tensors are denoted with uppercase letters, and indices to the dimensions of the tensors are denoted in lowercase subscripts. For instance $X_{ijk} \in \mathbb{R}^{I \times J \times K}$ is a tensor of size $I \times J \times K$ with three dimensions (or modes) indexed by $i \in [1, I]$, $j \in [1, J]$, and $k \in [1, K]$. Using Einstein notation, any multiplication among tensors can be written as: $C_{s_3} = \sum_{(s_1 \cup s_2) \setminus s_3} A_{s_1} B_{s_2}$ where s_1, s_2, and s_3 are the index sets of the left argument, the right argument, and the result tensor, respectively. The summation of inner products will be made explicit by underlining indexes corresponding to the dimensions to reduce in the input tensors. Let us illustrate our notation through a set of common tensor operations. The Hadamard product between $Y, X \in \mathbb{R}^{I \times J}$ can be written as $Z_{ij} = X_{ij} Y_{ij}$ and is equivalent to the algebraic notation $Z = X \odot Y$, being $\odot$ an element-wise multiplication. The matrix multiplication of $X \in \mathbb{R}^{I \times K}$ and $Y \in \mathbb{R}^{K \times J}$ can be written as $Z_{ij} = X_{i\underline{k}} Y_{\underline{k}j}$ and is equivalent to the algebraic notation $Z = XY$; note that in this case we reduce dimension K. Finally, the outer product between $Y \in \mathbb{R}^{I}$ and $X \in \mathbb{R}^{J}$ is denoted as $Z \in \mathbb{R}^{I \times J}$ as $Z_{ij} = X_i Y_j$, and it is equivalent to the algebraic expression $Z = Y^\top X$. When using a chained sequence of operations, we use the $\rfloor$ symbol to write each intermediate result.

4. Method

As a starting point of our investigation, we re-formulate the spiral convolution operator of Equation (1) using Einstein notation. This provides two main advantages: first, it enables an intuitive formulation and description of multi-dimensional tensor expressions without resorting to heavy tensor algebra; and second, it allows to analyze and discuss the different roles played by the weight tensor of a Neural Network layer, identified in its generic form with the uppercase W. With this in mind, let us define the SpiralConvolution operator in Einstein notation as

$$Y_{ind} = X_{in\underline{mc}} \, W_{\underline{mc}d} \tag{2}$$

where X_{inmc} is a batch of $i \in [1, I]$ input meshes composed of $n \in [1, N]$ vertices. Each vertex is the origin of a spiral neighborhood of $m \in [1, K]$ (e.g. $K = 9$ [4]) vertices and $c \in [1, C]$ features (*i.e.* channels). The W_{mcd} learnable weights used to aggregate vertices along the spirals are then applied to the input mesh tensor to generate an output Y_{ind} of processed meshes characterized by $n \in [1, N]$ vertices and $d \in [1, D]$ features. Upon closer inspection, Equation (2) clearly highlights three shortcomings for the weight tensor W_{mcd}: first, the weights are independent of the input; second, the weights are shared among all vertices; and third, the weights are limited to a fixed receptive field determined by the spiral length K. In other words, this basic form of the operator lacks of the characteristics of dynamicity to the input and spatial positions that have been proven successful in modern CNNs [47, 34] and even transformer architectures [49, 13]. Thus, we propose to generalize the original operator with the following modification of its weight ten-

sor, relaxing its aforementioned three inductive biases:

$$Y_{ind} = X_{in\underline{mc}}\, W_{i\underline{mn}\underline{c}d} \tag{3}$$

where $m \in [1, N]$ now contains the complete set of vertices in the mesh and thus follows (much longer) spiral trajectories. Note that in this formulation, the dynamic weights are implicitly assumed to be a function of input, $W_{imncd} = g(X_{inmcd})$, via an operator g, which in relevant literature is typically parameterized by a light-weight neural network [20]. In other words, we generalize the original SpiralConvolution of Equation (2) and propose a non-linear operator which is mesh-adaptive, vertex-adaptive, and has global receptive field. However, in this form, our non-linear generalization is hindered by high memory requirements (*i.e.* size of the weights), and $\mathcal{O}(N^2 \cdot C \cdot D)$ computational complexity. To overcome these drawbacks, we propose to approximate the behavior of the original Equation (3) by factorizing the dense tensor W_{imncd} into a sequence of lower-dimensional matrices which can be still easily trained end-to-end. Firstly, we separate vertex-wise processing from channels-wise processing by using the CP Decomposition[1], as is typically done for depthwise separable convolutions [9], obtaining:

$$\begin{cases} X^1_{imnr} = X_{imn\underline{c}} W^1_{\underline{c}r} \\ X^3_{inr} = X^1_{i\underline{m}nr} W^{23}_{i\underline{m}nr} \\ Y_{ind} = X^3_{in\underline{r}} W^4_{\underline{r}d} \end{cases} \tag{4}$$

where W^1_{cr} and W^4_{rd} implement linear layers and $r \in [1, R]$ is the rank of the decomposition, representing the number of components that are used to approximate the original tensor W_{inmcd}. Equation (4) is an improvement over (3), since it reduces the computational complexity to the sum of a sequence of operations plus the complexity of the function used to create matrix W^{23}_{imnr}. Nevertheless, a naïve implementation is still limited to cases where the data size is small enough to fit in memory: first, directly using an hypernetwork to generate the tensor W^{23}_{imnr} is computationally very demanding since its computation scales quadratically with the number of vertices; second, in cases where the number of vertices is higher than features, the complexity of equation (4) is still $\mathcal{O}(N^2)$. To overcome this problem and satisfy the two seemingly mutually exclusive objectives of efficient implementation and non-stringent inductive biases, we propose a new alternative formulation for the term W^{23}_{imnr} as a succession of two light-weight vertex-reasoning terms: one local dynamic term associated with the weight tensor W^2_{imnr} and one efficient global dynamic term associated with the weight tensor W^3_{imnr}. The first term focuses on replacing the shared and static weights in SpiralConvolution with a tensor capable of adaptively processing the local neighbors along the spiral, hence significantly relaxing the stringent inductive bias of the original operator. To achieve this goal efficiently, we propose to use a vertex-wise gating layer (G), similarly to [33]:

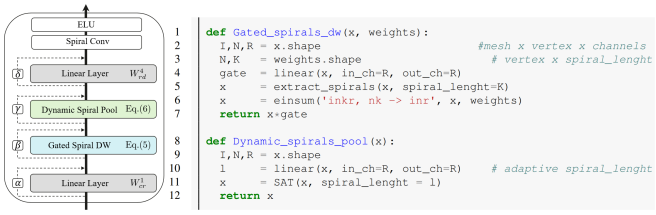
$$X^2_{inr} = X^1_{i\underline{m}nr} W^2_{i\underline{m}nr} = \left[X^1_{i\underline{m}nr} W^L_{\underline{m}n} \right]_{inr} W^G_{inr} \tag{5}$$

with $W^G_{inr} = f^G(X_{inr})$ where f^G is an arbitrary function which we implement as a linear transformation, and W^L_{mn} is a weight tensor of learnable parameters that relates each $n \in [1, N]$ vertex in the mesh to its $m \in [1, K]$ local neighbors along the spiral. The second term aims to relax the bias of local and static receptive field in SpiralConvolution. We propose to use a learnable pooling function (P) to obtain dynamic and potentially global reasoning on the entire mesh while avoiding the direct computation of a weight tensor of N^2 parameters:

$$X^3_{inr} = X^2_{i\underline{m}nr} W^3_{i\underline{m}nr} = \left[X^2_{i\underline{m}nr} W^S_{\underline{m}l} \right]_{i\underline{l}nr} W^P_{i\underline{l}n} \tag{6}$$

where the size of the extraction of spiral patches: X^2_{imnr} of Equation (6) comes from X^2_{inr} of Equation (5) where a spiral neighborhood indexed by $m \in [1, N]$ has been extracted for each vertex. W^S_{ml} is a triangular matrix of ones implementing a cumulative sum over X^2_{imnr} along the m dimension and $W^P_{iln} = f^P(X_{imnr})$ is a dynamic function predicting the size of the receptive field for each mesh and vertex as one-hot vectors along the l dimension, with ($l \in [1, M]$). In practice, in our implementation we use integral image or Summed-Area-Tables (SAT) [11, 52, 63] a technique for fast region accumulation over 2D images that enables the computation of pooling operations on a receptive field of arbitrary size with a constant computational cost. We adapt this technique to spirals and implement it as a 1D cumulative sum over the spiral length, followed by a thresholding operator determining the size of the receptive field .

The advantages of this implementation are two-fold: i) the model is able to actively adapt the receptive field according to the irregular structure of the mesh and ii) vertex-reasoning can be performed at constant computational cost




```
def Gated_spirals_dw(x, weights):
  I,N,R = x.shape                          #mesh x vertex x channels
  N,K   = weights.shape                     # vertex x spiral_lenght
  gate  = linear(x, in_ch=R, out_ch=R)
  x     = extract_spirals(x, spiral_lenght=K)
  x     = einsum('inkr, nk -> inr', x, weights)
  return x*gate

def Dynamic_spirals_pool(x):
  I,N,R = x.shape
  l     = linear(x, in_ch=R, out_ch=R)     # adaptive spiral_lenght
  x     = SAT(x, spiral_lenght = l)
  return x
```

Figure 2: **Overview** of our implementation. $\alpha, \beta, \gamma, \delta$ are learnable scalars.

[1]Also known as CANDECOMP/PARAFAC or polyadic decomposition, which is a generalization of singular value decomposition (SVD) for multi-dimensional tensors [27]

Method	Acc.(%)	Param(M)
FeaStNet*	79.24	1.91
MoNet*	86.05	1.91
ChebyNet*	98.77	1.91
SpiralNet*	72.84	1.91
SplineCNN [15]	99.20	4.15
SpiralNet++ [18]	99.88	1.91
Ours	**99.88**	**0.31**

Table 1: **3D shape correspondence on FAUST** . Comparison among different geometric operators in term of the test accuracy (i.e ratio of correct correspondence with null geodesic error). Results of methods marked with "*" are reported as in [18].

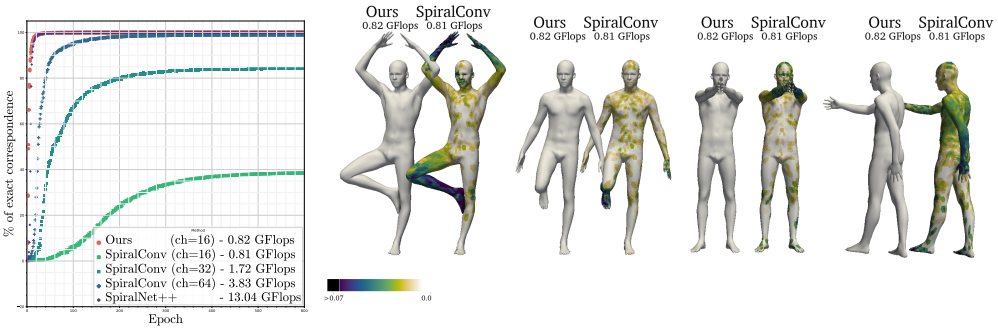


Figure 3: **3D shape correspondence performance on FAUST. Left) Validation curves** of Ours, SpiralConv, and SpiralNet++. SpiralConv is evaluated in three variants with an increasing number of channels (16, 32, 64), characterized by 124K, 256K and 563K parameters respectively. SpiralNet++ uses SpiralConv but follow the architecture described in [18]. Our method trend resembles SpiralNet++ (purple), while requiring x15 less GFlops. **Right) Visual comparison** between our method and SpiralConv under the same network complexity (0.82 GFlops) and training setup (ch=16). Geodesic error is shown saturated at 10% of the geodesic diameter. Thanks to its adaptive behavior, our method outperforms a static SpiralConv network by a large margin.

even when the predicted ideal receptive field is global ($m \in [1, N]$). Moreover, thanks to the use of interpolation, the whole layer is fully-differentiable and learned end to end. In our experiments, we modify the architecture to add one of our blocks before every static convolution, and add a residual connection after each component to ease convergence. Figure 2 shows an overview of our implementation. More details can be found in additional material.

5. Experiments

In this section, we compare different operators used to process 3D deformable shapes that are represented as 3D meshes that share a common topological structure, such as human faces and bodies. We benchmark our approach using two established tasks, *i.e.* mesh reconstruction and shape correspondence, comparing our method against existing mesh structure-based operators for 3D representation learning and generation.

5.1. 3D Correspondence

We evaluate the performance of the proposed method in the task of 3D dense shape correspondence, a task that aims at registering each node of a given shape with the corresponding node of a reference shape [35]. Following related work [3, 35, 37, 51], we use the FAUST dataset [2] which includes 10 human shapes represented by 6,890 vertices captured in 10 different poses, covering a variety of actions. As in [35, 3], we use the first 80 human shapes in the FAUST dataset for training and the remaining 20 for testing. The registration task is formulated as classification, with the ground truth correspondence of FAUST meshes given implicitly via nodes arranged in the same order for every example. The quality of the performance is measured as the percentage of the inferred correspondences that are located inside the region delimited by a geodesic radius r around the correct node, using the calculation outlined in [25]. To showcase the ability of our operator to process 3D meshes in an efficient yet adaptive way, we compare its performance against other well-known geometric deep learning operators [51, 37, 12, 4, 15, 18]. Specifically, we follow [18] and use a six-layer neural network to process directly raw 3D coordinates. We set the number of features as 16 in all layers, and use our operator on the three central layers. A detailed description of the architecture can be found in additional material.To ensure a fair comparison with the rest of the evaluated methods, we train our model using the standard cross-entropy loss and an Adam [26] optimizer with a learning rate of 3e-3. Moreover, we set the kernel size of the local operator to $K = 9$. In Table 1, we report the accuracy of our model and other approaches in obtaining exact correspondences (i.e. 0% geodesic error). Remarkably, our method reaches an accuracy of 99 % with a parameter count of approximately 300k, 83% less than the second-best performing method (SpiralNet++) and more than 92% less than the parameter count of SplineCNN (4.15M). Note that our operator is not limited to consider only 9 vertices, since is still able to learn the optimal size of the receptive field thanks to the use of the efficient learnable pooling layer. Instead it can leverage mesh topologies, local and global structures in the data, and dynamically adapt its response to the input mesh, achieving a significantly better accuracy-complexity trade-off than all other baselines. Lastly, we compare against other more recent deep-geometric methods for mesh processing. Figure 4 shows the % of correspondence against the normalized geodesic error according to the Princeton benchmark protocol, a measure of how well a shape descriptor matches a point on a surface to its corresponding point on another surface. As visible, our method outperforms the majority of the prior art and even performs

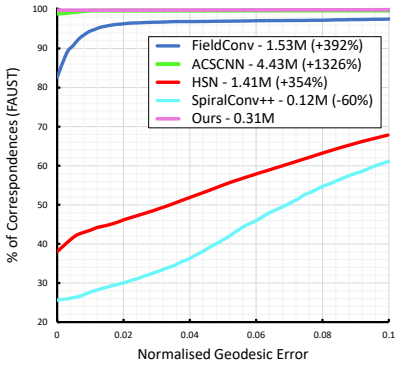


Figure 4: **FAUST**. Percentage of correspondences for a given geodesic error. Methods are reported together with their parameter count.

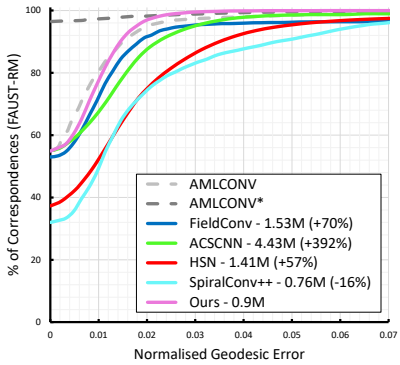


Figure 5: **FAUST Remeshed**. Adaptive spirals provide a good trade off between performance and parameter count.

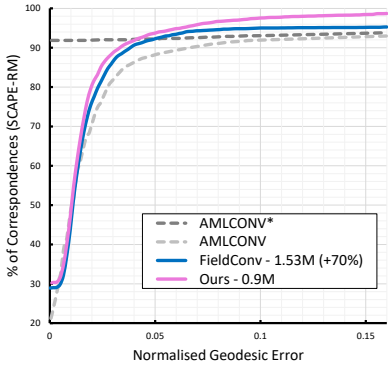


Figure 6: **SCAPE Remeshed**. Adaptive spirals outperform both the alternative operator of FieldConv and AMLCONV, which leverages functional maps.

comparably to ACSNN, which uses $+1326\%$ more parameters.

5.1.1 Comparison with SpiralConvolution

In this section, we compare our method against its closest prior art, the SpiralConvolution operator used in SpiralNet++ [18] (SpiralConv). For the sake of fairness, we use the same backbone architecture, and replace the three central layers of the network with a SpiralConv of different numbers of channels (16, 32, 64). We use as benchmark the task of 3D shape correspondence on FAUST: performance is measured as the percentage of exact shape correspondences, whereas complexity is measured both as the number of floating-point operations (FLOPS) and parameter count. The results in Figure 3 (left) illustrate the performance of all compared operators throughout training, together with the validation curve of SpiralNet++, reported for reference. As clearly visible, our method exhibits the best convergence properties: *i)* it significantly outperforms a SpiralConv variant of similar complexity (ch=16), *ii)* it converges to almost perfect accuracy faster than the variant of SpiralConv with x4 more channels (ch=64), and *iii)* it outperforms even Spiralnet++ which accounts for x15 more GFlops. These results demonstrate the effectiveness of our proposed operator and the importance of appropriately including adaptive mechanisms in the processing of 3D meshes. Finally, in Figure 3 (right) we visualize the geodesic error of our method against the SpiralConv variant having comparable GFlops (*i.e.* the one with 16 channels). This comparison clearly highlights the superior performance of our method, thus confirming that including adaptive processing operations leads to learning better mesh representations.

5.1.2 Adaptive Weights Visualization

A core characteristic of our operator consists of its ability to adaptively generate different kernels depending on the mesh and vertex at hand. Here, we seek to provide additional insights by visualizing the generated convolutional weights for the task of dense shape correspondence on the FAUST dataset, using a t-SNE plot. We select a sample of 10 meshes and 10 vertices (equidistant between 0 and 6889) considering 100 examples in total. Then we extract all kernels from the 3 adaptive spiral layers and project them into 2D points using t-SNE. As visible in Figure 7, plotting these points clearly shows the adaptive nature of our kernels: firstly, kernels processing the same vertex are grouped in distinct clusters. Furthermore, within each cluster, we observe how there is a local variation of the kernels as they adapt their response to the 10 different meshes.

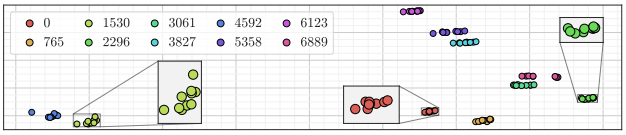


Figure 7: **t-SNE plot of the adaptive kernels** used by our method on the task of Dense shape correspondence on FAUST.

5.2. Correspondence on different mesh topologies

Next, we explore the ability of our method to generalize on meshes with varying positions and triangulation, by experimenting with the task of finding pointwise correspondence between shapes that do not share the same mesh connectivity. We compare against additional geometric deep-learning works [30, 36, 56] on the more challenging 5k remeshed variant of the FAUST dataset. Moreover, we provide evidence of the generalization ability of our method by testing on the 5K remeshed variant of the SCAPE [1] dataset (a dataset containing 71 registered meshes of a single subject in different poses). In practice, we follow the same network design as in the previous section and trained our method as well as the static baseline of SpiralConv++. To ensure a fair comparison with recent works, we follow the experimental setup described in FieldConv [36], replicating comparing using the same input descriptors and performance metric. As visible in Figure 5 and 6 SpiralConv++ is vastly outperformed by our counterpart on FAUST and fails to converge on SCAPE, possibly due to its limited receptive field. Despite the fact that we compare against methods

Method	Mean Error	Median Error	Params
FeaStNet [51]	0.523 ± 0.643	0.297	158K
MoNet [37]	0.526 ± 0.605	0.353	155K
COMA [43]	0.470 ± 0.598	0.263	**117K**
ChebyConv [12]	0.436 ± 0.562	0.242	155K
Neural3DMM [4]	0.443 ± 0.560	0.245	157K
SpiralNet++[18]	0.423 ± 0.534	0.236	155K
Aff-MoNet [17]	0.406 ± 0.455	0.251	168K
Ours -II	**0.389 ± 0.521**	**0.214**	172K
Ours -III	**0.373 ± 0.507**	**0.207**	198K

Table 2: **Performance comparison on 3D shape reconstruction** on the COMA dataset, using a 4-stages encoder-decoder and kernel size K=9. Our method outperforms competitors by a large margin with a small increase in parameter count.

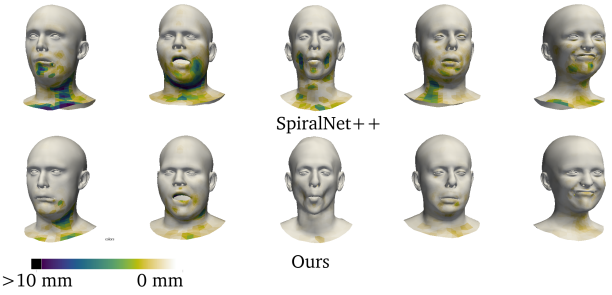


Figure 8: **Qualitative results of 3D shape reconstruction on the COMA dataset**. Visualization of the per-vertex euclidean distance from ground truth (mm) for our method and SpiralNet++. Errors are saturated at 10mm. 3DMM largely benefits from the dynamic behavior of our method .

specifically designed to process meshes of changing topologies, our solution achieves the best performance. Moreover, our performance is comparable to more complex state-of-the-art functional maps pipelines like AMLCONV [14]. These results confirm that our method is capable of generalizing robustly beyond meshes of fixed topologies, by adapting its instance and vertex aware response beyond the graph structure of the template mesh.

5.3. 3D Reconstruction with 3DMM

In this section, we showcase the ability of our method to capture complex 3D shape variations and generate high-quality reconstructions. Specifically, we investigate the task of 3D shape reconstruction with Neural 3D Morphable Models (3DMM) and compare the results obtained using our approach against existing state-of-the-art methods. As our benchmark, we use the COMA dataset [43] which includes 20,466 3D Meshes capturing 12 classes of extreme expression sequences for 12 different subjects. Each mesh is pre-processed to have 5,023 vertices. We follow the interpolation setting in [43]. To evaluate the generalization ability of the model, *i.e.* the ability to reconstruct data similar to those seen during training, we split the dataset in training and test samples with a ratio of 9:1. We then measure performance as the error between each vertex in the input and its reconstruction. We use as the evaluation metric the euclidean distance (in millimeters) in the 3D space reporting for each evaluated method, the mean and median euclidean errors, and the standard deviation over all samples and vertices.

5.3.1 Effect of Different Operators

In this section, we =evaluate the performance of our method against different 3D mesh processing operators: COMA [43], FeaStNet [51], MoNet [37], ChebyNet [12], SpiralNet [4], SpiralNet++ [18], and Aff-Monet [17]. To ensure a fair comparison, we employ the same model architecture, local kernel size, and training setup as related work [17]. We use as base architecture a 4-stage encoder-decoder with a latent space dimension d=16, ELU activation function, and a number of channels for each stage set to be [32,32,32,64], trained for 300 epochs using an Adam optimizer with a learning rate of 0.001 and a learning rate decay of 0.99 per epoch. The batch size is set to 32, and the kernel size for local layers is set to $K = 9$. To investigate the effectiveness of our proposed operator to enrich the 3D representation learning of an established baseline, we use SpiralConvolution 3DMM architecture modified to include our operator in either two (Ours-II) or three (Ours-III) stages. Further implementation details can be found in the supplementary material. Table 2 reports performance together with the number of parameters as a measure of complexity. The experimental results show that our method significantly improves the representation power of the network, outperforming all compared methods by a large margin with only a slight increase in the number of parameters. Our proposed method achieves a mean error of 0.389 ± 0.521 and 0.373 ± 0.507 for Ours-II and Ours-III, respectively, which represents a 7.6% and 20.5% decrease in mean error compared to the state-of-the-art SpiralNet++ and a 17.0% and 20.9% decrease compared to the lightest model COMA.

Figure 8 shows the qualitative results of 3D shape reconstruction on the COMA dataset, highlighting the contribution of our block when applied to the SpiralNet++ backbone. For each vertex, the error is computed as the Euclidean distance from the ground truth. Our model makes use of the adaptive convolutional filters, which ensure unique processing of each shape, and the dynamic receptive field allowing effective learning at multiple scales. It is able to reconstruct high-fidelity 3D shapes and reproduce a high degree of facial expression details, vastly improving the baseline. Figure 1 shows the learned spiral sequences of our method compared against the baseline for 6 different vertices. While the spiral length of SpiralNet++ is limited to 9, our operator is able to learn the optimal length dynamically, spanning from 7 to 70 length $[41, 13, 70, 62, 68, 66, 7]$. More visualization can be found in additional material.

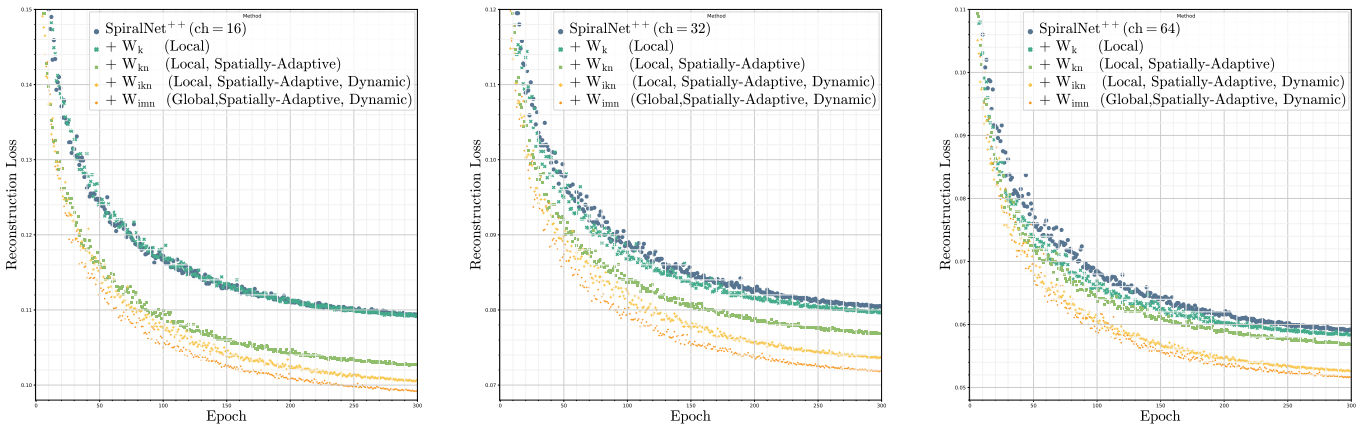


Figure 9: **Study on the effect of different inductive biases.** The SpiralNet++ baseline is augmented with different layers for three different network sizes ch=16,32,64 (from left to right). Training curves show how the effect of layers incorporating different inductive biases in the network impacts the overall performance. Interestingly, different characteristics play different roles in different network sizes. In all cases, our method (orange) is capable to integrate global reasoning, vertex-wise adaptivity (i.e. spatially-adaptive), and instance-wise adaptivity (*i.e.* dynamic) achieving the best performance.

5.3.2 Effect of Different Characteristics

We recall that our layer acts as a non-linear generalization for the SpiralConvolution layer. It has three main characteristics: *i)* is spatially adaptive *ii)* is dynamic (*i.e.* its weight tensor responds to each mesh in a unique way) and *iii)* is capable to enlarge its receptive field up via learnable pooling, potentially including all the vertices in the mesh. In Figure 9 we study the contribution of each characteristic using a SpiralNet++ backbone with three different sizes ($ch = 16, 32, 64$). The baseline operator in SpiralNet++ can be defined from a static weight tensor W_{kcd} of fixed receptive field $K = 9$. In this experiment, we incorporate operators characterized by different inductive biases (which are made explicit by the subscripts in the corresponding weight tensors) into the central 2 stages of the architectures to evaluate their individual contribution over the baseline in terms of final reconstruction loss. Specifically, we evaluate four additions to SpiralNet++: *i)* a layer parametrized with a tensor W_k implemented as a depth-wise convolution with $K = 9$ (cyan line), *ii)* a layer parametrized with a tensor W_{kn} representing a depth-wise spatially adaptive convolution with $K = 9$ (green line), *iii)* a layer parametrized with a tensor W_{ikn} implemented as a spatially adaptive and dynamic depth-wise convolution with $K = 9$, analogous to that one described in Equation (5) (yellow line), and lastly *iv)* a layer parametrized with a tensor W_{imn} that is global, spatially adaptive and dynamic, implemented containing both components from Equation (5) and Equation (6) (orange line). As clearly visible from the figure, for all network sizes, the use of W_k does not provide any particular advantage over the baseline, since the information contained in W_k is already included in the baseline tensor W_{kcd}. As expected, the other operations increasingly improve performance, although their relative contributions change depending on the size of the network. For instance, characteristics of spatial adaptivity are more effective for small network sizes, because intuitively, the ability of the network to process larger receptive fields is limited by its capacity. However, in all cases, our method achieves the best performance, and vastly outperforms the baseline configuration, thus demonstrating that characteristics of adaptive processing with dynamic receptive field cannot be easily captured by naïvely increasing the number of parameters.

5.3.3 Effect of Kernel Sizes

To validate the impact of the proposed approach with different receptive fields, we follow the experimental setup of [17] and extend the results on shape reconstruction for two extra values of local kernel size ($K = 4$ and $K = 14$) while keeping the remaining architecture and experimental setup unchanged. Moreover, we extend the previous section results evaluating our method in 3 variants (-II, -III, -IV), depending on the number of encoder-decoder stages augmented in the original architecture. Results are provided in Table 3. As visible from the table, our method shows consistent improvement across network dimensions and kernel sizes, thus providing state-of-the-art performance on all the evaluated setups with $+11.98\%$, $+16.13\%$ and $+16.26\%$ improvement for $K = 4$, $K = 9$ and $K = 14$ respectively. Notably, the use of tensor factorization and Summed-Area-Tables limits the overall complexity of the network, allowing our method to leverage vertex awareness, mesh aware-

Method	K=4 Mean Error	K=4 Median Error	K=4 Params	K=9 Mean Error	K=9 Median Error	K=9 Params	K=14 Mean Error	K=14 Median Error	K=14 Params
ChebNet*	0.659 ± 0.783	0.391	92K	4.329 ± 3.591	3.453	155K	4.348 ± 3.587	3.469	217K
SpiralNet++*	0.554 ± 0.674	0.320	92K	0.430 ± 0.542	0.239	155K	0.385 ± 0.491	0.214	217K
FeaStNet*	0.599 ± 0.730	0.342	94K	0.524 ± 0.646	0.297	158K	0.488 ± 0.599	0.279	222K
MoNet*	0.671 ± 0.760	0.450	93K	0.528 ± 0.604	0.354	155K	0.480 ± 0.551	0.321	218K
Aff-MoNet [17]	0.499 ± 0.579	0.298	105K	0.406 ± 0.455	0.251	168K	0.347 ± 0.386	0.218	230K
Ours -II	0.512 ± 0.665	**0.296**	106K	**0.389 ± 0.521**	**0.214**	172K	**0.343 ± 0.481**	**0.197**	239K
Ours -III	**0.472 ± 0.611**	**0.271**	119K	**0.373 ± 0.507**	**0.207**	198K	**0.340 ± 0.468**	**0.188**	276K
Ours -IV	**0.439 ± 0.599**	**0.244**	160K	**0.345 ± 0.481**	**0.186**	289K	**0.314 ± 0.454**	**0.166**	419K

Table 3: **3D shape reconstruction results for different kernel sizes K.** Methods are trained on the COMA [43] dataset, with a 4-stage encoder-decoder structure. All errors are given in millimeters. Our method, in the -II and -III variants, augment the central 2 and 3 stages of SpiralNet++ with our proposed layer, respectively. Variant -IV augment all 4 stages. Results for the methods presenting a "*" are reported from [17]. Our method's results are consistent across kernel and network sizes, outperforming competitors in all their variants.

ness, and dynamic learning of the receptive field without adding unnecessary complexity overhead. In additional material, we extend these results on two additional datasets.

5.3.4 Adaptive Weights Visualization

We provide extra visualizations of the behavior of the adaptive weights produced by our method. We extend the analysis done on 3D shape correspondence to the task of 3D mesh generation using 3DMM. We start by randomly selecting a subset of 10 meshes from the COMA test set. Then, we extract the learnable kernels from the two inner stages of Ours-II model, specifically grouping together encoder and decoder kernels according to the size of the resolution of the mesh. We follow the setup described for the FAUST dataset, and we select an equidistanced subset of 10 vertices, projecting their specialized kernels into 2D points using t-SNE. Figures 10 and 11 show the mesh-adaptive and instance-adaptive design of our operators. In the plots, kernels grouped together are associated with different vertices, and furthermore, inside each group, a different response can be identified for every different mesh. This trend is replicated across stages and tasks, providing insights into the adaptive nature of our method.

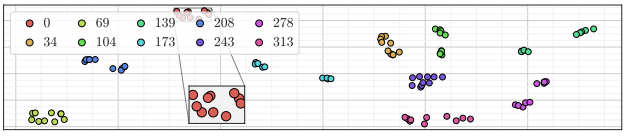


Figure 10: **t-SNE plot of the adaptive kernels** used by our method on the task of 3D shape generation on COMA. Stage 2

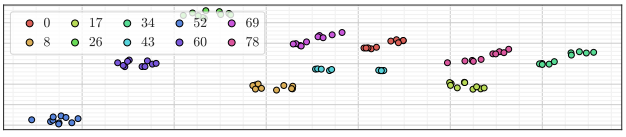


Figure 11: **t-SNE plot of the adaptive kernels** used by our method on the task of 3D shape generation on COMA. Stage 1

6. Conclusion

Drawing inspiration from modern deep learning architectures in the Euclidean domain, we propose a new adaptive operator for 3D meshes that extends SpiralConvolution to its non-local, mesh-aware, and vertex-aware variant. The adaptive nature of our operator does not come at a large computational cost, because it is specifically designed to factorize individual operations into lighter processing blocks, without sacrificing performance. In the task of shape correspondence, compared to SpiralNet++, our operator reduces the Giga Floating Point Operations per second by almost 94% without any loss in performance. Further, using our operator to enrich the representation of 3DMM, we report state-of-the-art performance. We provide evidence of the adaptive behavior of our method using t-SNE plots. In additional material we extend the experimental section with results on additional datasets, extra visualization, and ablations, providing more insights into the inner workings of our method. These results demonstrate that adapting operators to the underlying graph structure can be accomplished without significantly increasing complexity, showcasing our method as a promising new operator for 3D representation learning. However, extracting spirals from meshes with different topologies necessarily limits our method to separately process the spirals for each individual mesh, which could be cumbersome in real applications. In the future, we plan to address this problem by explicitly adapting our method to work with heterogeneous meshes and sampling trajectories. Additional interesting directions would be considering different, non-necessarily spiral, trajectories in order to adapt to different representations such as point clouds or even 2D images on a grid.

7. Acknowledgement

S. Zafeiriou and part of the research were funded by the EPSRC Fellowship DEFORM(EP/S010203/1) and by the EPSRC Project GNOMON (EP/X011364/1).

References

[1] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005. 6

[2] 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*, pages 3794–3801, 2014. 5

[3] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 5

[4] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7213–7222, 2019. 2, 3, 5, 7

[5] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 3

[6] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[7] 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*, pages 11030–11039, 2020. 2

[8] Zhixiang Chen and Tae-Kyun Kim. Learning feature aggregation for deep 3d morphable models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13164–13173, 2021. 2, 3

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 4

[10] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. Deep polynomial neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4021–4034, 2021. 3

[11] Franklin C Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 207–212, 1984. 4

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016. 2, 5, 7

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3

[14] Mohammad Farazi, Wenhui Zhu, Zhangsihao Yang, and Yalin Wang. Anisotropic multi-scale graph convolutional network for dense shape correspondence. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3146–3155, 2023. 7

[15] 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*, pages 869–877, 2018. 2, 5

[16] Zhongpai Gao, Junchi Yan, Guangtao Zhai, Juyong Zhang, and Xiaokang Yang. Robust mesh representation learning via efficient local structure-aware anisotropic convolution. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2, 3

[17] Shunwang Gong, Mehdi Bahri, Michael M Bronstein, and Stefanos Zafeiriou. Geometrically principled connections in graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11415–11424, 2020. 2, 7, 8, 9

[18] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019. 2, 3, 5, 6, 7

[19] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020. 2

[20] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2, 4

[21] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021. 2

[22] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Trans. Graph.*, 38(4), jul 2019. 2

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[24] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009. 2

[25] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM transactions on graphics (TOG)*, 30(4):1–12, 2011. 5

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[27] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. 3, 4

[28] Sören Laue, Matthias Mitterreiter, and Joachim Giesen. A simple and efficient tensor calculus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4527–4534, 2020. 3

[29] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 3

[30] Qinsong Li, Shengjun Liu, Ling Hu, and Xinru Liu. Shape correspondence using anisotropic chebyshev spectral cnns. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 14658–14667, 2020. 2, 6

[31] Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3

[32] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1800–1809, 2020. 2

[33] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021. 2, 4

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 3

[35] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 5

[36] Thomas W Mitchel, Vladimir G Kim, and Michael Kazhdan. Field convolutions for surface cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10001–10011, 2021. 2, 6

[37] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, 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*, pages 5115–5124, 2017. 2, 5, 7

[38] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015. 3

[39] Yannis Panagakis, Jean Kossaifi, Grigorios G Chrysos, James Oldfield, Mihalis A Nicolaou, Anima Anandkumar, and Stefanos Zafeiriou. Tensor methods in computer vision and deep learning. *Proceedings of the IEEE*, 109(5):863–890, 2021. 3

[40] Adrien Poulenard and Maks Ovsjanikov. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 2

[41] 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*, pages 652–660, 2017. 2

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[43] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, pages 704–720, 2018. 7, 9

[44] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2021. 3

[45] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 2

[46] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1

[47] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096–10106. PMLR, 2021. 3

[48] 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*, pages 6411–6420, 2019. 2

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 2, 3

[51] 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*, pages 2598–2606, 2018. 2, 5, 7

[52] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 4

[53] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2

[54] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 2

[55] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic

graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), oct 2019. 3

[56] Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)*, 39(4):92–1, 2020. 2, 6

[57] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019. 2

[58] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1

[59] Yuqi Yang, Shilin Liu, Hao Pan, Yang Liu, and Xin Tong. Pfcnn: Convolutional neural networks on 3d surfaces using parallel frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13578–13587, 2020. 2

[60] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888, 2021. 3

[61] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 3

[62] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[63] Linguang Zhang, Maciej Halber, and Szymon Rusinkiewicz. Accelerating large-kernel convolution using summed-area tables. *arXiv preprint arXiv:1906.11367*, 2019. 4