Geometrically Principled Connections in Graph Neural Networks

Shunwang Gong[1*]   Mehdi Bahri[1*]   Michael M. Bronstein[1,2,3]   Stefanos Zafeiriou[1,4]

[1]Imperial College London   [2]Twitter   [3]USI Lugano   [4]FaceSoft.io

{shunwang.gong16, m.bahri, m.bronstein, s.zafeiriou}@imperial.ac.uk

Abstract

Graph convolution operators bring the advantages of deep learning to a variety of graph and mesh processing tasks previously deemed out of reach. With their continued success comes the desire to design more powerful architectures, often by adapting existing deep learning techniques to non-Euclidean data. In this paper, we argue geometry should remain the primary driving force behind innovation in the emerging field of geometric deep learning. We relate graph neural networks to widely successful computer graphics and data approximation models: radial basis functions (RBFs). We conjecture that, like RBFs, graph convolution layers would benefit from the addition of simple functions to the powerful convolution kernels. We introduce affine skip connections, a novel building block formed by combining a fully connected layer with any graph convolution operator. We experimentally demonstrate the effectiveness of our technique, and show the improved performance is the consequence of more than the increased number of parameters. Operators equipped with the affine skip connection markedly outperform their base performance on every task we evaluated, i.e., shape reconstruction, dense shape correspondence, and graph classification. We hope our simple and effective approach will serve as a solid baseline and help ease future research in graph neural networks.

1. Introduction

The graph formalism has established itself as the *lingua franca* of non-Euclidean deep learning, as graphs provide a powerful abstraction for very general systems of interactions. In the same way that classical deep learning developed around the Convolutional Neural Networks (CNNs) and their ability to capture patterns on grids by exploiting local correlation and to build hierarchical representations by stacking multiple convolutional layers, most of the work on graph neural networks (GNNs) has focused on the formulation of convolution-like local operators on graphs.
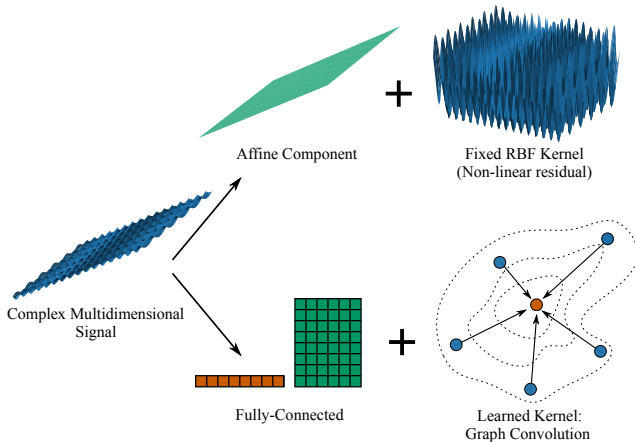
---

*Authors contributed equally.

Figure 1: The comparison made in this paper between learned graph convolution kernels and RBF interpolation suggests augmenting graph convolution operators with additive affine transformations, implemented as parametric connections between layers. Our affine skip connections improve the network's ability to represent certain transformations, and enable better use of the vertex features.

In computer vision and graphics, early attempts at applying deep learning to 3D shapes were based on dense voxel representations [42] or multiple planar views [48]. These methods suffer from three main drawbacks, stemming from their *extrinsic* nature: high computational cost of 3D convolutional filters, lack of invariance to rigid motions or non-rigid deformations, and loss of detail due to rasterisation.

A more efficient way of representing 3D shapes is modeling them as surfaces (two-dimensional manifolds). In computer graphics and geometry processing, a popular type of efficient and accurate discretisation of surfaces are *meshes* or simplicial complexes (see, *e.g.*, [8, 10, 25, 7, 37, 21, 14]), which can be considered as graphs with additional structure (faces). *Geometric deep learning* [9] seeks to formulate *intrinsic* analogies of convolutions on meshes accounting for these structures.

As a range of effective graph and mesh convolution operators are now available, the attention of the community is turning to improving the basic GNN architectures used in

graph and mesh processing to match those used in computer vision. Borrowing from the existing literature, extensions of successful techniques such as residual connections [22] and dilated convolutions [52] have been proposed [38, 39, 51], some with major impact in accuracy [29]. We argue, however, that due to the particularities of meshes and to their non-Euclidean nature, geometry should be the foundation for architectural innovations in geometric deep learning.

Contributions In this work, we provide a new perspective on the problem of deep learning on meshes by relating graph neural networks to Radial Basis Function (RBF) networks. Motivated by fundamental results in approximation, we introduce geometrically principled connections for graph neural networks, coined as *affine skip connections*, and inspired by thin plate splines. The resulting block learns the sum of any existing graph convolution operator and an affine function, allowing the network to learn certain transformations more efficiently. Through extensive experiments, we show our technique is widely applicable and highly effective. We verify affine skip connections improve performance on shape reconstruction, vertex classification, and graph classification tasks. In doing so, we achieve *best in class* performance on all three benchmarks. We also show the improvement in performance is significantly higher than that provided by residual connections, and verify the connections improve representation power beyond a mere increase in trainable parameters. Visualizing what affine skip connections learn further bolsters our theoretical motivation.

Notations Throughout the paper, matrices and vectors are denoted by upper and lowercase bold letters (*e.g.*, $\mathbf{X}$ and ($\mathbf{x}$), respectively. $\mathbf{I}$ denotes the identity matrix of compatible dimensions. The ith column of $\mathbf{X}$ is denoted as $\mathbf{x}_i$. The sets of real numbers is denoted by $\mathbb{R}$. A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of *vertices* $\mathcal{V} = \{1, \ldots, n\}$ and *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The graph structure can be encoded in the *adjacency matrix* $\mathbf{A}$, where $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$ (in which case i and j are said to be *adjacent*) and zero otherwise. The *degree matrix* $\mathbf{D}$ is a diagonal matrix with elements $d_{ii} = \sum_{j=1}^{n} a_{ij}$. The *neighborhood* of vertex i, denoted by $\mathcal{N}(i) = \{j : (i, j) \in \mathcal{E}\}$, is the set of vertices adjacent to i.

2. Related work

Graph and mesh convolutions The first work on deep learning on meshes mapped local surface patches to pre-computed geodesic polar coordinates; convolution was performed by multiplying the geodesic patches by learnable filters [33, 5]. The key advantage of such an architecture is that it is intrinsic by construction, affording it invariance to isometric mesh deformations, a significant advantage when dealing with deformable shapes. MoNet [35] generalized the approach using a local system of pseudo-coordinates $\mathbf{u}_{ij}$ to represent the neighborhood $\mathcal{N}(i)$ and a family of

learnable weighting functions w.r.t. $\mathbf{u}$, e.g., Gaussian kernels $w_m(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_m)^\mathsf{T} \boldsymbol{\Sigma}_k^{-1}(\mathbf{u} - \boldsymbol{\mu}_m)\right)$ with learnable mean $\boldsymbol{\mu}_m$ and covariance $\boldsymbol{\Sigma}_m$. The convolution is

$$\mathbf{x}_i^{(k)} = \sum_{m=1}^{M} \theta_m \sum_{j \in \mathcal{N}(i)} w_m(\mathbf{u}_{ij}) \mathbf{x}_j^{(k-1)} \tag{1}$$

where $\mathbf{x}_i^{(k-1)}$ and $\mathbf{x}_i^{(k)}$ denotes the input and output features at vertex i, respectively, and $\boldsymbol{\theta}$ is the vector of learnable filter weights. MoNet can be seen as a Gaussian Mixture Model (GMM), and as a more general form of the Graph Attention (GAT) model [45]. Local coordinates were re-used in the Spline Convolutional Network [18], which represents the filters in a basis of smooth spline functions. Another popular attention-based operator is FeaStNet [46], that learns a soft mapping from vertices to filter weights, and has been applied to discriminative [46] and generative models [32]:

$$\mathbf{x}_i^{(k)} = \mathbf{b} + \frac{1}{|\mathcal{N}(i)|} \sum_{m=1}^{M} \sum_{j \in \mathcal{N}(i)} q_m(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}) \mathbf{W}_m \mathbf{x}_j^{(k-1)} \tag{2}$$

where $\mathbf{W}_m$ a matrix of learnable filters weights for the m-th filter, q_m is a learned soft-assignment of neighbors to filter weights, and $\mathbf{b}$ the learned bias of the layer.[1]

ChebNet [15] accelerates spectral convolutions by expanding the filters on the powers of the graph Laplacian using Chebychev polynomials. Throughout this paper, we will refer to the n-order expansion as ChebNet-n. in particular the first order expansion ChebNet-1 reads

$$\mathbf{X}^{(k)} = -\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \mathbf{X}^{(k-1)} \boldsymbol{\Theta}_1 + \mathbf{X}^{(k-1)} \boldsymbol{\Theta}_0 \tag{3}$$

with $\mathbf{L} = -\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ the normalised symmetric graph Laplacian, $\mathbf{A}$ is the graph adjacency matrix, and $\mathbf{D}$ is the degree matrix. In computer graphics applications, ChebNet has seen some success in mesh reconstruction and generation [40]. However, due to the fact that spectral filter coefficients are basis dependent, the spectral construction is limited to a single domain. We therefore do not evaluate the performance of ChebNet on correspondence tasks. We refer to [28, 16] for constructing compatible orthogonal bases across different domains. The Graph Convolutional Network (GCN) model [27] further simplifies (3) by considering first-order polynomials with dependent coefficients, resulting in

$$\mathbf{X}^{(k)} = \tilde{\mathbf{L}} \mathbf{X}^{(k-1)} \boldsymbol{\Theta}, \tag{4}$$

where $\tilde{\mathbf{L}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$. By virtue of this construction, GCN introduces self-loops. GCN is perhaps the simplest graph neural network model combining vertex-wise feature transformation (right-side multiplication by $\boldsymbol{\Theta}$) and graph propagation (left-side multiplication by $\tilde{\mathbf{L}}$).

---

[1] It is tacitly assumed here that $i \in \mathcal{N}(i)$.

For this reason, it is often a popular baseline choice in the literature, but it has never applied successfully on meshes.

Recently, models based on the simple consistent enumeration of a vertex's neighbors have emerged. SpiralNet [31] enumerates the neighbors around a vertex in a spiral order and learns filters on the resulting sequence with a neural network (MLP or LSTM). The recent SpiralNet++ [20] improves on the original model by enforcing a fixed order to exploit prior information about the meshes in the common case of datasets of meshes that have the same topology, *e.g.*, [3, 4, 40]. The SpiralNet++ [20] operator is written $\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(||_{j \in S(i,M)} \mathbf{x}_j^{(k-1)} \right)$ with $\gamma^{(k)}$ an MLP, $||$ the concatenation, and $S(i, M)$ the spiral sequence of neighbors of i of length (*i.e.* kernel size) M.

Finally, we include the recently proposed Graph Isomorphism Network (GIN) [50] with the update formula

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left((1 + \epsilon^{(k)}) \cdot \mathbf{x}_i^{(k-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j^{(k-1)} \right). \quad (5)$$

This model is designed for graph classification and was shown [50] to be as powerful as the Weisfeiler-Lehman graph isomorphism test.

Skip connections and GNNs Highway Networks [43, 44] present shortcut connections with data-dependant gating functions, which are amongst the first architectures that provided a means to effectively train deep networks. However, highway networks have not demonstrated improved performance due to the fact that the layers in highway networks act as non-residual functions when a gated shortcut is "closed". Concurrent with this work, pure identity mapping [22] made possible the training of very deep neural networks, and enabled breakthrough performance on many challenging image recognition, localization, and detection tasks. They improve gradient flow and alleviate the vanishing gradient problem. DenseNets [24] can be seen as a generalization of [22] and connect all layers together. Early forms of skip connections in GNNs actually predate the deep learning explosion and can be traced back to the Neural Network for Graphs (NN4G) model [34], where the input of any layer is the output of the previous layer plus a function of the vertex features [49, section V.B]. [2] In [29], the authors propose direct graph equivalents for residual connections and dense connections, provide an extensive study of their methods, and show major improvements in the performance of the DGCNN architecture [47] with very deep models.

3. Motivation: Radial Basis Interpolation

The main motivation of this paper comes from the field of data interpolation. Interpolation problems appear in many

machine learning and computer vision tasks. In the general setting of *scattered data interpolation*, we seek a function $\hat{f}$ whose outputs $\hat{f}(\mathbf{x}_i)$ on a set of scattered data points $\mathbf{x}_i$ equals matching observations $\mathbf{y}_i$, *i.e.*, $\forall i, f(\mathbf{x}_i) = \mathbf{y}_i$. In the presence of noise, one typically solves an approximation problem potentially involving regularization, *i.e.*

$$\min_f \sum_i d(\hat{f}(\mathbf{x}_i), \mathbf{y}_i) + \lambda L(\hat{f}), \quad (6)$$

where d measures the adequation of the model $\hat{f}$ to the observations, λ is a regularization weight, and L encourages some chosen properties of the model. For the sake of the discussion, we take $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$. In computer graphics, surface reconstruction and deformation (*e.g.* for registration [13]) can be phrased as interpolation problems.

In this section, we draw connections between graph convolutional networks and a classical popular choice of interpolants: Radial Basis Functions (RBFs).

Radial basis functions An RBF is a function of the form $\mathbf{x} \mapsto \phi(||\mathbf{x} - \mathbf{c}_i||)$, with $||.||$ a norm, and $\mathbf{c}_i$ some pre-defined centers. By construction, the value of an RBF only depends on the distance from the centers. While an RBF function's input is scalar, the function can be vector-valued.

In interpolation problems, the centers are chosen to be the data points ($\mathbf{c}_i = \mathbf{x}_i$) and the interpolant is defined as a weighted sum of radial basis functions centered at each $\mathbf{x}_i$:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^N w_i \phi(||\mathbf{x} - \mathbf{x}_i||). \quad (7)$$

Interpolation assumes equality, so the problem boils down to solving the linear system $\mathbf{\Phi}\mathbf{w}_i = \mathbf{b}_j$, with $\mathbf{\Phi}_{j,i} = \phi(||\mathbf{x}_i - \mathbf{x}_j||)$ the matrix of the RBF kernel (note that the diagonal is $\phi(\mathbf{0}) \ \forall i$). The kernel matrix encodes the relationships between the points, as measured by the kernel.

Relaxing the equality constraints can be necessary, in which case we solve the system in the least squares sense with additional regularization. We will develop this point further to introduce our proposed affine skip connections.

Relations to GNNs An RBF function can be seen as a simple kind of one layer neural network with RBF activations centered around every points (*i.e.* an RBF network [11, 36]). The connection to graph neural networks is very clear: while the RBF matrix encodes the relationships and defines a point's neighborhood radially around the point, graph neural networks rely on the graph connectivity to hard-code spatial relationships. In the case of meshes, this encoding is all-the-more relevant, as a notion of distance is provided either by the ambient space (the graph is embedded) or directly on the Riemannian manifold. The latter relates to the RBFs with geodesic distance of [41].

---

[2] We refer to [29, section 2.1] for a summary of subsequent approaches.

Most GNNs used on meshes fall into the message passing framework [19]:

$$\mathbf{x}_i^{(k)} =$$
$$\gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \underset{j \in \mathcal{N}(i)}{\square} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{ij}^{(k-1)} \right) \right),$$
(8)

where $\square$ denotes a differentiable permutation-invariant function, (*e.g.* max or $\sum$), ϕ a differentiable kernel function, γ is an MLP, and $\mathbf{x}_i$ and $\mathbf{e}_{ij}$ are features associated with vertex i and edge (i, j), respectively. This equation defines a compactly supported, and possibly *non-linear*, function around the vertex. For the MoNet equation (1) the connection to RBFs is direct. Contrary to RBFs, the filters of modern GNNs do not have to be radial. In fact, anisotropic filters [5, 6] have been shown to perform better than isotropic ones [33, 40]. The other major differences are:

1. The filters are learned functions, not pre-defined; this allows for better inductive learning and task-specificity

2. The filters apply to any vertex and edge features

3. Some operators support self-loops, but $\mathrm{diag}(\mathbf{\Phi}) = \phi(\mathbf{0})$ irrespective of the features $\mathbf{x}_i$

We note that the compact support of (8) is a design decision: early GNNs built on the graph Fourier transform lacked compactly-supported filters [23]. In RBF interpolation, global support is sometimes desired as it is a necessary condition for maximal *fairness* of the interpolated surfaces (*i.e.* maximally smooth), but also induces computational complexity and numerical challenges as the dense kernel matrices grow and become ill-conditioned [1]. This motivated the development of fast methods to fit locally supported RBFs [2]. In [23] the authors argue compactly-supported kernels are desirable in graph neural networks for computational efficiency, and to promote learning local patterns. This especially justified for meshes, for which the graph structure is very sparse. Additionally, stacking convolutional layers is known to increase the receptive field, including in graph neural networks [49]. The composition of locally supported filters can therefore yield globally supported mappings.

RBFs and polynomials A common practice with RBFs is to add low-order polynomial terms to the interpolant:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{x}_i||) + P(\mathbf{x}).$$
(9)

The practical motivation is to ensure polynomial mappings of some order can be represented exactly and to avoid unwanted oscillations when approximating flat functions, *e.g.* affine transformations of an image should be exactly

affine. One can show [1] this is equivalent to ensuring the RBF weights lie in the null space of the polynomial basis, also known as the *vanishing moments condition*.

However, polynomials appear organically when the RBF kernel is derived to be optimal for a chosen roughness measure, typically expressed in terms of the integral of a squared differential operator D (below in one dimension):

$$||Df||^2 = \int |Df(x)|^2 dx,$$
(10)

e.g., $D = \frac{d^2}{dx^2}$. In other words, when the kernel is sought to be optimal for a given *regularization functional*. Differential operators are very naturally expressed on meshes in terms of finite difference approximations. In this case, we identify D with its corresponding stencil matrix. The interpolation problem becomes the minimization of (10) subject to the interpolation constraints.

It can be shown [1] that for such problems the RBF kernel is the Green's function of the squared differential operator, and that for an operator of order m, polynomials of order $m - 1$ span the null space. Therefore, the complete solution space is the direct sum[3] of the space of polynomials of order $m - 1$ (the null space of the operator) and the space spanned by the RBF kernel basis[4].

Thin Plate Splines (TPS) An important special case is the RBF interpolant for a surface $z(\mathbf{x})$, $\mathbf{x} = [x \quad y]^\top$ that minimizes the bending energy $\int \int \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x \partial y} + \frac{\partial^2 f}{\partial y^2} dx dy = ||\Delta^2 f||$. The solution is the well-known biharmonic spline, or *thin plate spline*, $\phi(r) = r^2 \log r, \quad r = ||\mathbf{x} - \mathbf{x}_i||$, with a polynomial of degree 1 (*i.e.* an affine function)

$$\hat{f}(\mathbf{x}) = \sum_i w_i \phi(||\mathbf{x} - \mathbf{x}_i||) + \mathbf{A}\mathbf{x} + \mathbf{b}.$$
(11)

Generalizations to higher dimensions yield *polyharmonic splines*. These splines maximize the surface *fairness*. From (11) it is also clear *the polynomial doesn't depend on the structure of the point set* and is common for all points.

4. Geometrically Principled Connections

In Section 3, we highlighted key similarities and differences between continuous RBFs and discrete graph convolution kernels . We then exposed how adding low-order polynomials to RBFs kernels is both beneficial to enable efficient fitting of flat functions, and deeply connected to regularization of the learned functions, and noted the polynomial component does not depend on spatial relationships. Based on these observations, we conjecture that graph convolution operators could, too, benefit from the addition of

---

[3]Hence the vanishing moment condition.
[4]This result comes from phrasing the problem as regularization in a Reproducing Kernel Hilbert Space. To keep the discussion short in this manuscript, we refer the reader to relevant resources such as [1, Section 7].

Figure 2: Our block learns the sum of one graph convolution and a shortcut equipped with an affine transformation.

a low-order polynomial to ensure they can represent flat functions exactly, and learn functions of a vertex's features independently from its neighbours. We introduce a simple block that achieves both goals.

Inspired by equation (11), we propose to augment a *generic* graph convolution operator with *affine skip connections*, *i.e.*, inter-layer connections with an affine transformation implemented as a fully connected layer. The output of the block is the sum of the two paths, as shown in Figure 2.

Our block is designed to allow the fully connected layer to learn an affine transformation of the current feature map, and let the convolution learn a residue from a vertex's neighbors. For message passing, we obtain:

$$\mathbf{x}_i^{(k)} =$$
$$\gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \underset{j \in \mathcal{N}(i)}{\square} \phi^{(k)}(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j}^{(k-1)}) \right) \quad (12)$$
$$+ \mathbf{A}^{(k)} \mathbf{x}_i^{(k-1)} + \mathbf{b}^{(k)}.$$

The fully connected layer could be replaced by an MLP to obtain *polynomial connections*, however, we argue the stacking of several layers creates sufficiently complex mappings by composition to not require deeper sub-networks in each block: a balance must be found between expressiveness and model complexity. Additionally, the analogy with TPS appears well-motivated for signals defined on surfaces. **As a matter of notation, we refer to our block based on operator Conv with affine skip connections as *Aff*-Conv.**

In equations (9), (11) and (12), the polynomial part does not depend on a vertex's neighbors, but solely on the feature at that vertex. This is similar to PointNet [12] that learns a shared MLP on all points with no structural prior. In our block, the geometric information is readily encoded in the graph, while the linear layer is applied to all vertices independently, thus learning indirectly from the other points regardless of their proximity.

Residual blocks with projections In [22, Eq. (2)], the authors introduced a variation of residual blocks with a projection implemented as a linear layer. Their motivation is to handle different input and output sizes. We acknowledge the contribution of residual connections and will demonstrate our block provides the same benefits and more for GNNs.

5. Experimental evaluation

Our experiments are designed to highlight different properties of affine skip connections when combined. We present the individual experiments, then draw conclusions based on their entirety. All implementation details (model architecture, optimizers, losses, *etc*.), and details about the datasets (number of samples, training/test split) are provided in Appendix A of the supplementary material.

5.1. Experimental design

Mesh reconstruction The task is to reconstruct meshes with an auto-encoder architecture, and relates the most to interpolation. To validate the proposed approach, we firstly show the performance of attention-based models, MoNet and FeaStNet, on shape reconstruction on CoMA[40] for different values of M. For a kernel size of M, we compare the vanilla operators (MoNet, FeaStNet), the blocks with residual skip connections (*Res*-MoNet, *Res*-FeaStNet), the blocks with affine skip connections (*Aff*-MoNet, *Aff*-FeaStNet), and the vanilla operators with kernel size $M+1$ (MoNet+, FeaStNet+)[5]. We evaluated kernel sizes 4, 9, and 14. We report the mean Euclidean vertex error and its standard deviation, and the median Euclidean error. Results with SplineCNN [17] are shown in Appendix B of the supplementary material.

Mesh correspondence The experimental setting is mesh correspondence, *i.e.*, registration formulated as classification. We compare MoNet, FeaStNet and their respective blocks on the FAUST [3] dataset. We purposefully do not include SpiralNet++ and ChebNet on this problem: the connectivity of FAUST is *fixed* and vertices are in correspondence already. These methods assume a fixed topology and therefore have an unfair advantage. We report the percentage of correct correspondences as a function of the geodesic error.

Mesh correspondence with GCN The GCN [27] model is arguably the most popular graph convolution operator, and has been widely applied to problems on generic graphs thanks to its simplicity. However, its performance degrades quickly on meshes, which makes the entry bar higher for prototyping graph-based approaches in 3D vision. We investigate whether affine skip connections can improve the performance of GCN, and by how much. We choose the 3D shape correspondence task, in order to allow for comparison with the other models already included in this study. As detailed in the supplementary material, the network used in this experiment is relatively deep, with three convolution layers. In [27, Appendix B] the authors add residual connections to GCNs deeper than two layers to alleviate vanishing gradients. In order to prove affine skip connections have a geometric meaning, we must eliminate the possibility that better per-

---

[5]Increasing the kernel size by 1, and adding an affine skip connection lead to the same number of weight matrices.

method	M=4			M=9			M=14		
	mean error	median	# param	mean error	median	# param	mean error	median	# param
ChebNet[†]	0.659 ± 0.783	0.391	92.5k	4.329 ± 3.591	3.453	154.9k	4.348 ± 3.587	3.469	217.3k
ChebNet	$\mathbf{0.520 \pm 0.655}$	**0.294**	92.5k	$\mathbf{0.438 \pm 0.562}$	**0.244**	154.9k	$\mathbf{0.407 \pm 0.523}$	**0.227**	217.3k
Res-ChebNet	0.531 ± 0.668	0.299	92.5k	0.444 ± 0.570	0.275	154.9k	0.412 ± 0.530	0.229	217.3k
SpiralNet++[†]	$\mathbf{0.554 \pm 0.674}$	**0.320**	92.5k	0.430 ± 0.542	0.239	154.9k	0.385 ± 0.491	0.214	217.3k
SpiralNet++	0.578 ± 0.705	0.333	92.5k	$\mathbf{0.426 \pm 0.538}$	**0.238**	154.9k	$\mathbf{0.383 \pm 0.489}$	**0.212**	217.3k
Res-SpiralNet++	0.575 ± 0.703	0.331	92.5k	0.432 ± 0.541	0.243	154.9k	0.395 ± 0.496	0.223	217.3k
FeaStNet	0.599 ± 0.730	0.342	93.8k	0.524 ± 0.646	0.297	157.9k	0.488 ± 0.599	0.279	221.9k
FeaStNet+	0.587 ± 0.723	0.333	106.6k	0.517 ± 0.635	0.292	170.7k	0.477 ± 0.594	0.268	234.8k
Res-FeaStNet	0.565 ± 0.701	0.314	93.8k	0.483 ± 0.602	0.266	157.9k	0.441 ± 0.554	0.279	221.9k
Aff-FeaStNet	$\mathbf{0.543 \pm 0.676}$	**0.303**	106.3k	$\mathbf{0.470 \pm 0.585}$	**0.261**	170.4k	$\mathbf{0.431 \pm 0.543}$	**0.237**	234.4k
MoNet	0.671 ± 0.760	0.450	92.7k	0.528 ± 0.604	0.354	155.4k	0.480 ± 0.551	0.321	218.1k
MoNet+	0.627 ± 0.693	0.429	105.2k	0.528 ± 0.587	0.366	167.9k	0.480 ± 0.540	0.329	230.6k
Res-MoNet	0.540 ± 0.612	0.335	92.7k	0.426 ± 0.479	0.271	155.4k	0.374 ± 0.417	0.238	218.1k
Aff-MoNet	$\mathbf{0.499 \pm 0.579}$	**0.298**	105.2k	$\mathbf{0.406 \pm 0.455}$	**0.251**	167.9k	$\mathbf{0.347 \pm 0.386}$	**0.218**	230.5k

Table 1: **3D shape reconstruction** experiments results in the CoMA [40] dataset. Errors are in millimeters. All the experiments were ran with the same network architecture. We show the results of each operator for different kernel sizes (i.e., # of weight matrices). *Aff*- denotes the operators equipped with the proposed affine skip connections, *Res*- denotes the operators with standard residual connections, and † indicates we remove the separate weight for the center vertex.
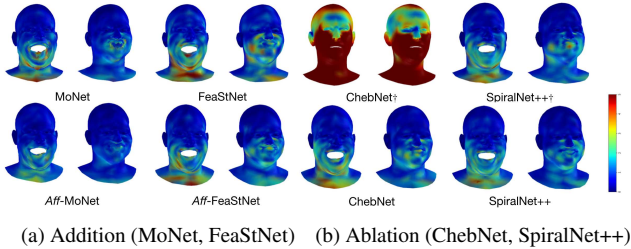

(a) Addition (MoNet, FeaStNet)   (b) Ablation (ChebNet, SpiralNet++)

Figure 3: **Sample reconstructions:** addition of affine skip connections and ablation of the center vertex weights.

Figure 4: Example reconstructed faces obtained by passing samples (**top**) through a trained autoencoder built on the *Aff*-MoNet block. The **middle** row shows reconstructions produced by the full autoencoder. The **bottom** row shows the result of passing through the affine skip connections only in the decoder at inference. The connections learn a smooth component common to the samples - across identities and expressions, as expected from the motivation.

formance comes solely from improved gradient flow. We include in this study a GCN block with vanilla residual connections (*Res*-GCN), in order to isolate the gradient flow improvements from the geometric improvements. Overall, we compare vanilla GCN, *Res*-GCN, and our *Aff*-GCN.

Graph classification   We compare MoNet, FeaStNet, and their respective residual and affine skip connection blocks on graph classification on Superpixel MNIST [35, 18]. The Superpixel MNIST dataset used in [35] and [18] represents the MNIST images as graphs. We use 75 vertices per image. All models use a kernel size of 25. We include GIN (built with a 2-layer MLP) for the similarity of its update rule with our block, in the GIN-0 ($\epsilon = 0$) variant for its superior performance as observed in [50]. We compare GIN with GCN, *Res*-GCN, and *Aff*-GCN. Here, graph connectivity is **not** fixed. We report the classification accuracy.

Ablation: separate weights for the centre vertex   To show the inclusion of the center vertex is necessary, we perform an ablation study of ChebNet, and SpiralNet++ on shape reconstruction on CoMA. From equation (3), we see the zero order term $\mathbf{X}\mathbf{\Theta}_0$ is an affine function of the vertex

features. We remove it from the expansion of ChebNet-$(M + 1)$ to obtain ChebNet-$M^{\dagger}$: $\mathbf{X}^{(k)} = \mathbf{L}^{(M+1)}\mathbf{X}^{(k-1)}\mathbf{\Theta}_{M+1} + \ldots + \mathbf{L}\mathbf{X}^{(k-1)}\mathbf{\Theta}_1$. Both models have identical numbers of weight matrices, but ChebNet-M learns from the vertices alone at order 0. For SpiralNet++, the center vertex is the first in the sequence $\{vertex||neighbors\}$. We rotate the filter (*i.e.* move it one step down the spiral) to remove the weight on the center vertex while keeping the same sequence length. We obtain SpiralNet++†. The number of weight matrices is constant. All models have kernel size 9.

Ablation: self-loops *vs*. affine skip connections   We also compare FeaStNet with and without self-loops (FeaStNet†),
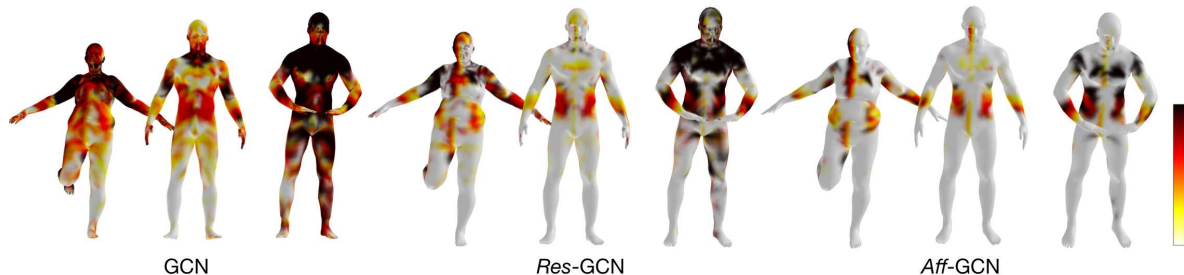
GCN   *Res*-GCN   *Aff*-GCN

Figure 5: **Shape correspondence** experiments on the FAUST humans dataset. Per-vertex heatmap of the geodesic error for three variants of the GCN operator. Geodesic error is measured according to [26].
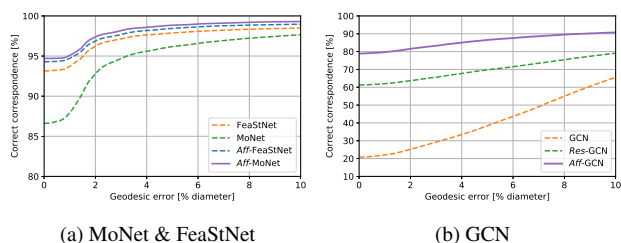

(a) MoNet & FeaStNet

(b) GCN

Figure 6: **Shape correspondence** accuracy: the x axis displays the geodesic error in % of the mesh diameter, and the y axis shows the percentage of correspondences that lie within a given radius around the correct vertex. All experiments were ran with the same architecture. *Aff*-GCN only has **1%** more parameters than GCN.

and the matching blocks, on all experiments.

5.2. Results and discussion

Based on the evidence collected, we draw conclusions about specific properties of our affine skip connections.

Parameter specificity The results of varying the kernel size on shape reconstruction can be found in Table 1 along with the corresponding number of parameters for control. Increasing the kernel size by 1 (MoNet+, FeaStNet+) provides only a minor increase in performance, *e.g.*, for $M = 9$ and $M = 14$, MoNet and MoNet+ have the same mean Euclidean error and the median error of MoNet with $M = 9$ actually *increases* by **3.4%**. In contrast, the affine skip connections always drastically reduce the reconstruction error, for the same number of additional parameters. In particular, the mean Euclidean error of MoNet decreased by **25.6%** for $M = 4$, and by **23.1%** for $M = 9$. We conclude our affine skip connections have a specific different role and augment the representational power of the networks beyond simply increasing the number of parameters. Our block with MoNet achieves the new *state of the art* performance on this task.

What do affine skip connections learn? In Figure 4, we observe the linear layers in the connections learned information common to all shapes. This result strengthens our

Method	Acc. (%)		Kernel Size	# Param
GIN-0	57.75		-	25k
GCN	31.21		-	15.9k
Res-GCN	42.32		-	15.9k
Aff-GCN	**58.96**		-	22.1k
FeaStNet	11.35		25	166k
Res-FeaStNet	58.09		25	166k
Aff-FeaStNet	**59.50**		25	172k
Pseudo-Coord.	*Degree*	*Position*	-	-
MoNet	53.10	96.57	25	164k
Res-MoNet	53.75	96.82	25	164k
Aff-MoNet	**72.00**	**97.14**	25	170k

Table 2: **Classification accuracy** of different operators and blocks on the Superpixel MNIST dataset with 75 superpixels. For MoNet, we report performance using pseudo-coordinates computed from the vertex positions, or from the connectivity only (vertex degrees).

analogy with the polynomial terms in RBF interpolation: the coefficients of the polynomial function are learned from all data points and shared among them. In one dimension, this can be pictured as learning the trend of a curve. Our visualizations are consistent with this interpretation.

Vertex-level representations We report the mesh correspondence accuracy as a function of the geodesic error for FeaStNet, MoNet, and the blocks in Figure 6a. We observe consistent performance improvements for both operators. The performance difference is remarkable for MoNet: for a geodesic error of 0, the accuracy improved from 86.61% to 94.69%. *Aff*-MoNet is the new *state of the art* performance on this problem[6]. We conclude affine skip connections improve vertex-level representations.

Laplacian smoothing and comparison to residuals We show the performance of GCN and its residual and affine blocks in Figure 6b. The accuracy of vanilla GCN is only around 20%. We can hypothesize this is due to the equivalence of GCN with Laplacian smoothing [30] - blurring the features of neighboring vertices and losing specificity - or

---

[6]Excluding methods that learn on a fixed topology.

method	mean error	median	# param	M	acc. (%)	# param.	M	acc. (%)	# param.	M
FeaStNet	0.524 ± 0.646	0.297	157.9k	9	93.14	1.91M	10	11.35	166k	25
Aff-FeaStNet	$\mathbf{0.470 \pm 0.585}$	0.261	170.4k	9	**94.29**	1.92M	10	**59.50**	172k	25
FeaStNet[†]	0.519 ± 0.634	0.297	157.9k	9	93.72	1.91M	10	11.35	166k	25
Aff-FeaStNet[†]	$\mathbf{0.463 \pm 0.577}$	0.256	170.4k	9	**94.36**	1.92M	10	**60.07**	172k	25
(a) **Shape Reconstruction**:					(b) **Correspondence**			(c) **Classification**		

Table 3: **Ablations: affine skip connection *vs*. self-loop**. We show the performances of FeaStNet under the settings of with and without self-loop (denoted with [†]) and with and without affine skip connections regarding the tasks of shape reconstruction on CoMA, shape correspondence on FAUST, and classification on MNIST with 75 superpixels. M denotes the kernel size (*i.e.* # weight matrices). For correspondence, test accuracy is the ratio of the correct correspondence prediction at geodesic error 0.

to the vanishing gradient problem. Our block outperforms vanilla residuals by a large margin: the classification rate of *Aff*-GCN is nearly 79% while *Res*-GCN only reaches 61.27%. Visually (Figure 5), *Res*-GCN provides marked improvements over GCN, and *Aff*-GCN offers another major step-up. A similar trend is seen in Table 1 and Table 2. In [22] the authors observed a minor performance increase between vanilla residuals and residual connections with projection, that they attributed to the higher number of parameters. The differences we observe are not consistent with such marginal improvements. This shows *not only* our approach provides all the benefits of residuals in solving the vanishing gradient problem, it achieves *more* on *geometric data*, and that the improvements are not solely due to more trainable parameters or improved gradient flow. In particular, with affine skip connections, Eq. 4 of [30] becomes $\sigma(\tilde{\mathbf{L}}\mathbf{H}^{(l)}\mathbf{\Theta}^{(l)} + \mathbf{H}^{(l)}\mathbf{W}^{(l)})$, with $\tilde{\mathbf{L}}$ the augmented symmetric Laplacian, and $\mathbf{W}^{(l)}$ the parameters of the affine skip connection. Thus, the *Aff*-GCN block is no longer equivalent to Laplacian smoothing.

Discriminative power   Our results on Superpixel MNIST are presented in Table 2. Our affine skip connections improve the classification rate across the board. GCN with affine skip connections outperform GIN-0 by over 1 percentage point, with 12% fewer trainable parameters. This result shows *Aff*-GCN offers competitive performance with a smaller model, and suggests the augmented operator is significantly more discriminative than GCN. Assuming the terminology of [50], FeaStNet employs a *mean* aggregation function, a choice known [50] to significantly limit the discriminative power of GNNs and which could explain its very low accuracy in spite of its large (166k) number of parameters. In contrast, *Aff*-FeaStNet is competitive with *Aff*-GCN and outperforms GIN-0. As GIN is designed to be as powerful of the WL test, these observations suggest affine skip connections improve the discriminative power of graph convolution operators. As a result, *Aff*-MoNet outperformed the current *state of the art*, for coordinate-based and degree-based pseudo-coordinates.

Role of the center vertex   As seen in the first six rows of Table 1, the performance of the models is higher with weights for the center vertex, especially for ChebNet. Note the comparison is at identical numbers of parameters. Fig-

ure 3 provides sample ablation and addition results. This shows convolution operators need to learn from the center vertices. We found that removing self-loops in FeaStNet actually *increased* the performance for both the vanilla and the block operators. Table 3 shows results on all experiments. The affine skip connection consistently improved the performance of models regardless of the self-loops. We conclude graph convolution operators should be able to learn specifically from the center vertex of a neighborhood, independently from its neighbors. A similar observation was made in [50] where independent parameters for the center vertex are shown to be required for graph convolution operators to be as discriminative as the WL test.

6. Conclusion

By relating graph neural networks to the theory of radial basis functions, we introduce *geometrically principled connections* that are both easily implemented, applicable to a broad range of convolution operators and graph or mesh learning problems, and highly effective. We show our method extends beyond surface reconstruction and registration, and can dramatically improve performance on graph classification with arbitrary connectivity. Our MoNet block achieves state of the art performance and is more robust to topological variations than sequence (SpiralNet++) or spectrum-based (ChebNet) operators. We further demonstrate our blocks improve on vanilla residual connections for graph neural networks. We believe our approach is therefore interesting to the broader community. Future work should study whether affine skip connections have regularization effects on the smoothness of the learned convolution kernels.

References

[1] Ken Anjyo, J P Lewis, and Frédéric Pighin. Scattered Data Interpolation for Computer Graphics. In *ACM SIGGRAPH 2014 Courses*, SIGGRAPH '14, pages 27:1—-27:69, New York, NY, USA, 2014. ACM. 4

[2] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Advances in Computational Mathematics*, 1999. 4

[3] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, jun 2014. IEEE. 3, 5

[4] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 5573–5582, jul 2017. 3

[5] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 3197–3205, 2016. 2, 4

[6] 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 *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 4

[7] Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1–20, 2011. 1

[8] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006. 1

[9] 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. 1

[10] Michael M Bronstein and Iasonas Kokkinos. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1704–1711. IEEE, 2010. 1

[11] Dave Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988. 3

[12] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85. IEEE, jul 2017. 5

[13] Yucong Chen, Junli Zhao, Qingqiong Deng, and Fuqing Duan. 3D craniofacial registration using thin-plate spline transform and cylindrical surface projection. *PLoS ONE*, 2017. 3

[14] Yoni Choukroun, Gautam Pai, and Ron Kimmel. Sparse approximation of 3d meshes using the spectral geometry of the hamiltonian operator. *Journal of Mathematical Imaging and Vision*, 60(6):941–952, 2018. 1

[15] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, (Nips):3844–3852, 2016. 2

[16] Davide Eynard, Artiom Kovnatsky, Michael M Bronstein, Klaus Glashoff, and Alexander M Bronstein. Multimodal manifold analysis by simultaneous diagonalization of laplacians. *IEEE transactions on pattern analysis and machine intelligence*, 37(12):2505–2517, 2015. 2

[17] 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. 5

[18] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Muller. SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018. 2, 6

[19] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, pages 2053–2070, 2017. 4

[20] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, oct 2019. 3

[21] Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. Unsupervised learning of dense shape correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4370–4379, 2019. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE, jun 2016. 2, 3, 5, 8

[23] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. pages 1–10, 2015. 4

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. 3

[25] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30, page 79. ACM, 2011. 1

[26] Vladimir G. Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM Transactions on Graphics*, 30(4):1, jul 2011. 7

[27] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Neural Networks. In *ICLR 2017*, pages 1–14, 2017. 2, 5

[28] Artiom Kovnatsky, Michael M Bronstein, Alexander M Bronstein, Klaus Glashoff, and Ron Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, volume 32, pages 439–448. Wiley Online Library, 2013. 2

[29] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs Go As Deep As CNNs? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 3

[30] Qimai Li, Zhichao Han, and Xiao Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018. 7, 8

[31] Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes. In *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part III*, pages 349–362, 2018. 3

[32] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable Shape Completion with Graph Convolutional Autoencoders. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (128):1886–1895, 2018. 2

[33] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic Convolutional Neural Networks on Riemannian Manifolds. *Proceedings of the IEEE International Conference on Computer Vision*, 2015-Febru:832–840, 2015. 2, 4

[34] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 2009. 3

[35] 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. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:5425–5434, 2017. 2, 6

[36] John Moody and Christian J. Darken. Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, 1989. 3

[37] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012. 1

[38] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Column networks for collective classification. In *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 2017. 2

[39] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. Semi-supervised user geolocation via graph convolutional networks. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2018. 2

[40] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3D Faces using Convolutional Mesh Autoencoders. In *The European Conference on Computer Vision (ECCV)*, sep 2018. 2, 3, 4, 5, 6

[41] Taehyun Rhee, Youngkyoo Hwang, James Dokyoon Kim, and Changyeong Kim. Real-time facial animation from live video tracking. In *Proceedings - SCA 2011: ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2011. 3

[42] Song, S., A Khosla, Xiao, and J. 3D ShapeNets: A deep representation for volumetric shapes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, jun 2015. 1

[43] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 3

[44] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015. 3

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, oct 2018. 2

[46] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jun 2018. 2

[47] 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):146:1—-146:12, oct 2019. 3

[48] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:1544–1553, nov 2016. 1

[49] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. jan 2019. 3, 4

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *Int. Conf. on Learning Representations*, oct 2018. 3, 6, 8

[51] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken Ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *35th International Conference on Machine Learning, ICML 2018*, 2018. 2

[52] Fisher Yu and Vladen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. *ICLR*, jun 2016. 2