Array 26 (2025) 100392

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

KA-GCN: Kernel-Attentive Graph Convolutional Network for 3D face analysis

Francesco Agnelli *, Giuseppe Facchi, Giuliano Grossi, Raffaella Lanzarotti

Università degli Studi di Milano, Milano, 20122, Italy

ARTICLE INFO

Keywords:
Graph Structure Learning (GSL)
3D face analysis
FaceScape dataset
Headspace dataset
Florence dataset
Sexual dimorphism recognition
Expression recognition
Action unit recognition
Attention network
Small dataset

ABSTRACT

Graph Structure Learning (GSL) methods address the limitations of real-world graphs by refining their structure and representation. This allows Graph Neural Networks (GNNs) to be applied to broader unstructured domains such as 3D face analysis. GSL can be considered as the dynamic learning of connection weights within a layer of message passing in a GNN, and particularly in a Graph Convolutional Network (GCN). A significant challenge for GSL methods arises in scenarios with limited availability of large datasets, a common issue in 3D face analysis, particularly in medical applications. This constraint limits the applicability of data-intensive GNN models, such as Graph Transformers, which, despite their effectiveness, require large amounts of training data. To address this limitation, we propose the Kernel-Attentive Graph Convolutional Network (KA-GCN). Our key finding is that integrating kernel-based and attention-based mechanisms to dynamically refine distances and learn the adjacency matrix within a Graph Structure Learning (GSL) framework enhances the model's adaptability, making it particularly effective for 3D face analysis tasks and delivering strong performance in data-scarce scenarios. Comprehensive experiments on the Facescape, Headspace, and Florence datasets, evaluating age, sexual dimorphism, and emotion, demonstrate that our approach outperforms state-of-the-art models in both effectiveness and robustness, achieving an average accuracy improvement of 2%. The project page is available on GitHub [1].

1. Introduction

In recent decades, Graph Neural Networks (GNNs) [1] have emerged as a valuable tool for learning from graph-structured data. GNNs have been successfully applied in several domains, including computer vision [2,3], natural language processing [4], chemistry [5], physics [6], biology [7], traffic networks [8], and recommendation systems [9].

The effectiveness of GNNs can be attributed to their ability to take advantage of the abundant information present in both graph structure and attributes, exploiting a message-passing mechanism to calculate global information, called states, for each local node [10]. Nevertheless, it is important to acknowledge that real-world graphs are often incomplete and noisy, which presents a major hurdle when applying GNNs to practical problems. GNNs exhibit high sensitivity to the quality of the provided data, which typically requires a denoised graph structure to effectively learn informative representations [11,12]. In some fields, the structure is provided directly, dealing with objects with an inherently defined graph structure, such as molecules or physical systems. On the other hand, in non-structural scenarios where the graph structure is not explicitly defined in the data, it is essential to carefully consider how to establish or acquire it [13]. Examples of application domains necessitating graph structure definition include natural language processing [4] and computer vision [14,15].

Recently, a considerable amount of literature explored the central idea of Graph Structure Learning (GSL) [11,12,16]. GSL aims to learn an optimized graph structure along with corresponding representations. According to [13], GSL approaches can be categorized into three areas: metric learning, probabilistic modeling, and direct optimization. Here, we focus in particular on metric learning methods, where edge weights are calculated by learning a metric function between the two end nodes. Interestingly, GSL can be conceptually viewed as part of a message-passing layer within a GNN, where the adjacency matrix is learned dynamically together with the other updating weights. Notable examples are GAT [17] and Transformer [18] graph convolutional layers, which exploit the correlations between nodes to derive the edge weights.

Although the guiding principle behind the learning of the adjacency matrix can be applied in any field, our focus in this study is on the unique characteristics of 3D faces [19,20],to determine the optimal graph structure for the various tasks in which 3D faces are used. 3D faces, unlike molecules, miss a predefined graph structure. The nodes

* Corresponding author.
E-mail address: francesco.agnelli@unimi.it (F. Agnelli).
[1] https://github.com/phuselab/KA-CONV

https://doi.org/10.1016/j.array.2025.100392
Received 3 December 2024; Received in revised form 17 March 2025; Accepted 22 March 2025
Available online 7 April 2025

in the graph can be accurately determined by a set of facial landmarks [1,21], but the challenge of determining potentially weighted links between these nodes remains unresolved. Another challenge to consider when working on 3D face analysis is the potential scarcity of data. This issue is particularly prominent in real-world scenarios like medical contexts, where there are limited samples to represent various categories (for example, rare diseases), making the task of learning particularly harsh. In fact, most of the existing 3D face datasets are significantly limited in the amount of data they offer. The biggest datasets available are still very limited: the Bosphorus dataset [22] contains 4666 samples, BU-3DFE [23] includes 2500 samples, and FaceScape [24] has 18760 samples. This scarcity reduces the performance of models specifically designed for large-scale problems, such as Transformers [25], requiring the introduction of architectural biases to enhance the model's accuracy.

The method we propose is framed within attentive models, leveraging their ability to focus on critical nodes while discarding irrelevant computations. In particular, it is based on the intuition that incorporating the concepts of proximity and distance between nodes into the attention mechanism serves as inductive positional bias [26–29], making it especially effective for small sample size problems. To accomplish this, we propose a kernel mechanism that adapts to the task at hand by learning the optimal neighborhood configuration.

In summary, this paper introduces a novel graph convolutional layer, the Kernel-Attentive Convolutional Layer (KA-Conv), which prioritizes the positions of landmarks in relation to other node features. The approach allows dynamically emphasized connections among nodes, whether close or distant. Nevertheless, it is possible to apply this approach to other kinds of graphs in which the node features can be divided into two sets, with one set being more relevant and prioritized over the other. Our model is expressly designed to perform effectively in scenarios with limited data, owing to the combination of a strong inductive positional bias with an attentive mechanism.

The rest of the paper is organized as follows: Section 2 formally introduces the problem formulation. Section 3 explains the proposed convolutional layer and how it is used in the proposed model. Section 4 details the features associated with the graph nodes. Section 5 shows the results and comparisons obtained. Section 6 draws some conclusions.

2. Notation and problem formulation

In the subsequent discussion, we denote a graph as a tuple $G = (V, E)$, where $v_i \in V$ indicates a vertex, $e_{ij} = (v_i, v_j) \in E \subseteq V \times V$ indicates an edge from node v_i to node v_j, and the cardinality of graph G is $N = |V|$. The neighborhood of a node v_i is denoted by the set $\mathcal{N}(v_i) = \{v_j \in V | (v_i, v_j) \in E\}$. $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix, where $A(i,j) > 0$ if and only if $e_{ij} \in E$, and each value of the adjacency matrix $a_{ij} \in \mathbb{R}^+$ represents the weight of the correspondent edge. In the context of 3D face graphs constructed from facial landmarks, the set V represents the 3D landmarks, which are points $p \in \mathbb{R}^3$ extracted from the point cloud P and correspond to distinct facial features, such as the corners of the eyes, the tip of the nose, and other key anatomical positions.

The graph structure can be enhanced with attributes. We denote by $H \in \mathbb{R}^{N\times d}$ the matrix of node attributes, where $h_i \in \mathbb{R}^d$ represents the features vector associated with node v_i. Similarly, edge attributes can be incorporated, with the matrix $K \in \mathbb{R}^{M\times c}$ representing the edge features, where $M = |E|$ and $k_{ij} \in \mathbb{R}^c$ denotes the feature vector for edge e_{ij}. The core of GSL is to simultaneously learn a new adjacency matrix $\tilde{A}$ and its corresponding graph representation $\tilde{H} \in \mathbb{R}^{N\times\tilde{d}}$ optimized for specific downstream tasks. The edge representation $\tilde{K} \in \mathbb{R}^{M\times\tilde{c}}$ can also be learned [13].

As stated in Section 1, here we focus on metric learning methods, where adjacency matrices are dynamically learned together with the other learnable weights through the message-passing layer within the GNN. Formally, metric learning approaches involve learning a metric function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined on node embeddings h_i, h_j as

$$\tilde{A}(i,j) = \phi(h_i, h_j). \tag{1}$$

Depending on the choice of function ϕ, it is possible to further classify the metric approaches into two subgroups:

- *Kernel-based approaches*, where the function ϕ is modeled with a traditional kernel function. The pioneering work was AGCN [30], which exploited the generalized Mahalanobis distance by rewriting the metric function as

$$\phi(h_i, h_j) = \sqrt{(h_i - h_j)^T M (h_i - h_j)}, \tag{2}$$

$$\tilde{A}(i,j) = \exp\left(-\frac{\phi(h_i, h_j)}{2\sigma^2}\right), \tag{3}$$

  where the symmetric positive semi-definite matrix $M = WW^T$, with $W \in \mathbb{R}^{N\times N}$ is a trainable matrix.
- *Attentive approaches*, where an attention network is used to model a new graph structure based on node interactions. It is worth noticing that an attentive network such as GAT [17] can be equivalently interpreted as an attentive GSL approach.

  Formally, given a node v_i, the iterative aggregation and update steps in the message-passing mechanism at layer l are

$$h_i^l = \gamma\left(h_i^{l-1}, \bigoplus_{j:v_j\in\mathcal{N}(v_i)} \phi(h_i^{l-1}, h_j^{l-1})\right), \tag{4}$$

  where $\bigoplus$ denotes a differentiable, permutation-invariant function, and ϕ and γ denote learnable functions. For graph-attentive convolution, Eq. (4) can be made explicit as

$$h_i^l = W_1 h_i^{l-1} + \sum_{v_j\in\mathcal{N}(v_i)} \alpha_{ij}^{l-1} W_2 h_j^{l-1}, \tag{5}$$

  where, with a little abuse of notation on the softmax function, the attentive coefficients at step $l-1$ are defined as

$$\alpha_{ij}^{l-1} = \text{softmax}\left(\frac{(W_3 h_i)^T (W_4 h_j + W_5 k_{ij})}{\sqrt{d}}\right), \tag{6}$$

  being $\{W_i\}_{i=1}^5$ trainable parameters. This leads to equating the adjacency matrix with the learned attention coefficients:

$$A^l(i,j) = \alpha_{ij}^{l-1}, \quad \forall (i,j) \in \mathbb{R}^{N\times N}. \tag{7}$$

Furthermore, sparsity regularization may be applied to reduce computational costs. Standard approaches result in k-NN graphs, where each node has at most k neighbors, or ϵ-NN graphs, where edges above a threshold ϵ are discarded.

3. The proposed method

As discussed in Section 1, we propose an attentive graph convolutional layer that utilizes the informational value of node distances through a kernel-based approach.

Specifically, assume the node feature vector is given by the concatenation of two components:

$$h_i = [\, pos_i \,\|\, \tilde{h}_i \,],$$

where $\|$ denotes vector concatenation, $pos_i = (x_i, y_i, z_i)$ represents the coordinates of the node (or, more generally, a set of relevant features), and $\tilde{h}_i = h_{i,4:d} \in \mathbb{R}^{d-3}$ are the remaining node features. We introduce the Kernel-Attentive Graph Convolutional layer (KA-Conv), where the metric function ϕ in Eq. (1) is defined as

$$\phi(h_i, h_j) = \delta(\mathcal{A}(h_i, h_j)), \tag{8}$$

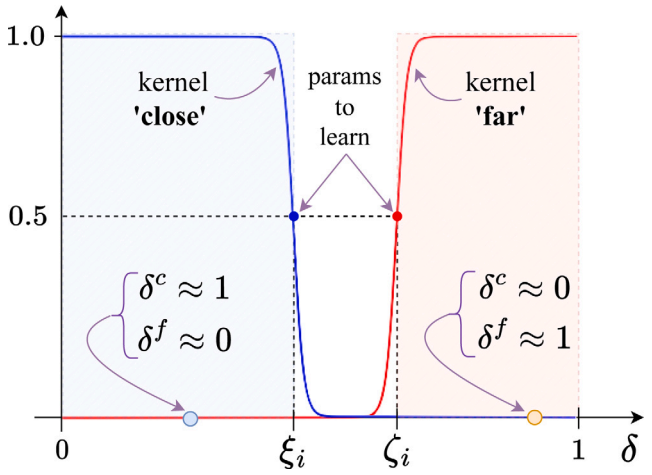

Fig. 1. The kernels 'close' (blue) and 'far' (red) as distance measures. The parameters ξ_i and ζ_i are initially set to 0.5 and then learned during the training.

with δ and $\mathcal{A}$ representing, respectively, a kernel and an attentive function, further elaborated in subsequent sections. Intuitively, this formula allows the model to incorporate both spatial and feature-based relationships, ensuring that nodes are weighted according to both their geometric proximity and learned feature similarities.

Notably, GNNs designed for point clouds follow a similar strategy. For instance, PointNetConv [31] uses a symmetric function to aggregate features directly from raw point clouds, ensuring order invariance and capturing global structure. PointGNNConv [32] iteratively updates point features using a GNN, preserving spatial relationships between points. PointTransformerConv [33] introduces self-attention to point clouds, enabling the model to learn long-range dependencies as well as local context. XConv [34] employs a permutation-invariant convolution operation to hierarchically extract features, capturing both local and global structures of point clouds. However, these models still lack a clear distinction in handling spatially close nodes versus those distant during convolution.

3.1. The kernel-attentive convolutional layer

To simultaneously optimize the graph structure and its associated representations, we integrate both kernel and attention mechanisms in the structure of our proposed layer KA-Conv. The kernel mechanism is designed to model the graph topology by learning a distance metric $\delta(pos_i, pos_j)$ that accurately captures the absolute positions of nearby and distant nodes. In parallel, attention coefficients $\mathcal{A}(\tilde{h}_i, \tilde{h}_j)$ are derived from the other geometric features using an attentive approach. The final graph representation is achieved by effectively integrating these two approaches.

Specifically, the distance measure is designed to increase the model's flexibility in prioritizing either nearby or distant nodes, enabling the model to learn varying weights based on the distance between nodes. To achieve this, two different three-dimensional distance measures, both inspired by the sigmoid function, are employed, as illustrated in Fig. 1.

The first measure δ^f, that is, the kernel 'far' defined as

$$\delta^f(pos_i, pos_j) = \left(1 + e^{-\lambda\left(\|pos_i - pos_j\|_2 - \xi_i\right)}\right)^{-1}, \tag{9}$$

can be intuitively understood as a high-pass filter that emphasizes larger distances while disregarding closely connected nodes, effectively highlighting significant macro-level information within the graph, such as facial dimensions or proportions. The parameter ξ_i, initially set to 0.5, is learnable and controls the filter threshold for each node v_i, allowing the model to choose the best parameter for its neighborhood. For all pairs $i \neq j$, the L_2 norms $\|pos_i - pos_j\|_2$ are normalized within the interval $[0, 1]$ to ensure both training stability and broad applicability. The value of λ is high enough (for example > 100) to allow the function δ^f to be a continuous approximation of the Heaviside function.

The second distance measure δ^c, that is, the kernel 'close', is symmetrically defined as

$$\delta^c(pos_i, pos_j) = 1 - \left(1 + e^{-\lambda\left(\|pos_i - pos_j\|_2 - \zeta_i\right)}\right)^{-1}, \tag{10}$$

where ζ_i is another learnable parameter analogous to ξ_i. Intuitively, this measure acts as a low-pass filter, giving more weight to spatially close nodes.

The computation of each attention coefficient $\mathcal{A}(\tilde{h}_i, \tilde{h}_j) = z^r_{ij}$ involves the features $\tilde{h}_i$ and $\tilde{h}_j$, with a specific emphasis on the chosen distance measure $r \in \{c, f\}$, as

$$z^r_{ij} = \mathrm{softmax}\left(\mathrm{LReLU}\left(\frac{\left(W^r_1 \tilde{h}_i\right)^T W^r_2 \tilde{h}_j}{\sqrt{d^{l-1}}}\right)\right), \tag{11}$$

where $W^r_1, W^r_2 \in \mathbb{R}^{(d^{l-1}-3)\times(d^{l-1}-3)}$ are two learnable matrices that do not depend on the nodes v_i, v_j, and LReLU is an abbreviation for the LeakyReLU function.

The edge weights α^r_{ij} are determined by the product of the attention coefficient z^r_{ij} and the distance coefficient $\delta^r(pos_i, pos_j)$ as

$$\alpha^r_{ij} = z^r_{ij}\delta^r(pos_i, pos_j). \tag{12}$$

Intuitively, this allows the model to determine the weight of the connection between nodes relying on both geometrical and positional features.

Optionally, to reduce computational costs, a sparsity constraint can be introduced. This can be done by defining

$$M^r = \max(\alpha^r_{ij}), \quad \forall (i, j) \in N \times N, \tag{13}$$

and using it as a threshold along with an arbitrary constant $\epsilon \in [0, 1]$, as follows:

$$\alpha^r_{ij} = \begin{cases} 0 & \text{if } \alpha^r_{ij} < \epsilon\, M^r \\ \alpha^r_{ij} & \text{if } \alpha^r_{ij} \geq \epsilon\, M^r. \end{cases} \tag{14}$$

Finally, the edge weights $\alpha^f_{ij}, \alpha^c_{ij}$ are linearly combined to obtain the final attention coefficients

$$\alpha_{ij} = l^c_{ij}\alpha^c_{ij} + l^f_{ij}\alpha^f_{ij}, \tag{15}$$

where $l^c_{ij}, l^f_{ij} \in [0, 1]$ are learnable parameters. The use of two separate parameters, instead of a convex combination, gives the model greater independence in emphasizing the most relevant component without, however, neglecting the other one altogether.

Once the kernel-attentive coefficients are settled, the updated state is derived using an attentive graph convolution as in Eq. (5). Following [35], the adjacency matrix obtained is normalized to $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where $D_{i,i} = \sum_{j=1}^{n} A_{ij}$.

If the graph is equipped with edge features k_{ij}, we supply an integration of KA-Conv to take full advantage of them. In this vein, we followed the design of the Transformer convolution [18], exploiting the edge features in both the calculation of the attention and the aggregate step of message-passing.

Firstly, Eq. (11) is completed as

$$z^r_{ij} = \mathrm{softmax}\left(\mathrm{LReLU}\left(\frac{\left(W^r_1 \tilde{h}_i\right)^T W^r_2 \tilde{h}_j + W^r_3 k_{ij}}{\sqrt{d^{l-1}}}\right)\right),$$

where $W^r_3 \in \mathbb{R}^{1\times c}$ is a linear, learnable transformation.

For the message-passing, we integrate edge features keeping the relation between edge e_{ij} and neighbor v_j of node v_i. Unlike the Transformer approach, where a linear transformation of the edge feature

is summed with the node feature, we opt to concatenate node and edge features and then apply a linear transformation. This approach should be more robust when the node and edge feature dimensions differ significantly, avoiding overexposing the low-dimensional vector. Hence, Eq. (5) is rewritten as

$$h_i^l = W_4 h_i^{l-1} + \sum_{v_j \in \mathcal{N}(v_i)} \alpha_{ij}^{l-1} W_5 \left[h_j^{l-1} \,\|\, k_{ij}^{l-1} \right], \tag{16}$$

where $\|$ denotes the vector concatenation on the last dimension, $W_4 \in \mathbb{R}^{d^{l-1} \times d^l}$ and $W_5 \in \mathbb{R}^{(d^{l-1}+c)\times d^l}$.

Algorithm 1 describes all the steps necessary to update the state of a node v_i using the KA-Conv. In our case, the set of edges E is equal to $N \times N$. However, the whole model can be easily adapted to different choices of the edge matrix E. Since KA-Conv can be seen as a more complex version of a Transformer graph layer with the integration of biases, the computational cost is slightly higher but still quadratic in the dimension of the input (in both cases $\mathcal{O}(d^2|E|)$). However, as pointed out in Section 1, the rationale of this model is the limited availability of 3D face data. Consequently, the increased resource usage is negligible.

Algorithm 1 KA-Conv layer

Input:

- features $h_i = (pos_i, \tilde{h}_i)$, $h_j = (pos_j, \tilde{h}_j)$, k_{ij}
- hyperparameters λ, ϵ
- learnable params W_1^r, W_2^r, W_3^r, W_4, W_5 l_{ij}^r, ξ_i, ζ_i

Output: updated features h_i

for $j : e_{ij} \in E$ **do**

$\alpha_{ij}^f = \left(1 + e^{-\lambda(||pos_i - pos_j||_2 - \xi_i)}\right)^{-1}$

$\alpha_{ij}^c = 1 - \left(1 + e^{-\lambda(||pos_i - pos_j||_2 - \zeta_i)}\right)^{-1}$

for $r \in \{c, f\}$ **do**

$z_{ij}^r = \text{softmax}\left(\text{LReLU}\left(\frac{\left(W_1^r \tilde{h}_i\right)^T W_2^r \tilde{h}_j + W_3^r k_{ij}}{\sqrt{d}}\right)\right)$

$\alpha_{ij}^r = z_{ij}^r \alpha_{ij}^r$

$\alpha_{ij}^r = \text{Normalize}_j(\alpha_{ij}^r)$ ▷ Normaliz. in $[0, 1]$

end for

$\alpha_{ij} = l_{ij}^c \alpha_{ij}^c + l_{ij}^f \alpha_{ij}^f$ ▷ Eq. (15)

$\alpha_{ij} = \alpha_{ij} / \left(\sum_{k=1}^n \alpha_{ik} \sum_{k=1}^n \alpha_{kj}\right)^{1/2}$ ▷ Laplacian

$h_i = W_4 h_i + \alpha_{ij} W_5 \left[h_j || k_{ij}\right]$

end for

3.2. The KA-GCN model

The KA convolutional layer described in Section 3.1 is integrated into the Kernel-Attention Graph Convolutional Network (KA-GCN) as shown in Fig. 2. In the design of the architecture, the model comprises one KA-Conv layer. After that, a combination of BatchNorm and ReLU generates the node embedding for the final classification. To address overfitting, a dropout layer has been added to the training part. The final classification layer is composed of an add pooling followed by a fully connected classifier.

4. Nodes and edges features

We characterize the graph by extracting local geometric features and basic statistics, similar to the bags-of-words approach in computer vision. One-hot encoding is used to uniquely identify facial landmarks. Fast Point Feature Histograms (FPFH) are employed to ensure nodes accurately reflect surface properties [36]. FPFH is a feature representation technique for 3D point cloud data that describes the local geometry around a query point. This representation is robust to pose variations and sampling density, making it effective for handling noisy sensor data.

Formally, given a point p_i on P, we examine its k nearest neighbors to compute pairwise comparisons. For each pair of points $p_i, p_j \in P$ we build a reference frame consisting of three unit vectors defined as

$$s = n_i, \tag{17}$$

$$t = s \times \frac{p_i - p_j}{\|p_i - p_j\|_2}, \tag{18}$$

$$z = s \times t, \tag{19}$$

where $\times$ denotes the cross product of two vectors, and n_i is the surface normal vector at p_i.

The difference between the two normals n_i and n_j can be expressed as a set of angular features:

$$\alpha = t^T n_j, \tag{20}$$

$$\phi = s^T \frac{p_i - p_j}{\|p_i - p_j\|_2}, \tag{21}$$

$$\theta = \arctan(z^T n_j, s^T n_j), \tag{22}$$

The attributes α and θ represent n_j as an azimuthal angle and the cosine of a polar angle, respectively, while ϕ represents the direction of the translation from p_i to p_j.

For each point $p_i \in P$, the resulting tuples $(\alpha, \varphi, \theta)_j$ obtained confronting p_i with its k neighbors $p_j \in P$ (here $k = 100$), for $j \in \{1, \dots, k\}$, are stored in the 11-bin histograms $H_\alpha(p_i)$, $H_\varphi(p_i)$ and $H_\theta(p_i)$. The final descriptor, known as SPFH (Simple Point Feature Histogram), is obtained by equalizing each histogram and horizontally concatenating them as

$$\text{SPFH}(p_i) = [H_\alpha(p_i), H_\phi(p_i), H_\theta(p_i)].$$

The ultimate FPFH feature is calculated by gathering the local information of each landmark point $v_i \in V$ on the face with respect to its neighbors on P, as

$$\text{FPFH}(v_i) = \text{SPFH}(v_i) + \frac{1}{|\mathcal{N}(v_i)|} \sum_{p_j \in \mathcal{N}(v_i)} \frac{1}{\delta_{ij}} \text{SPFH}(p_j)$$

where δ_{ij} denotes the distance between landmark v_i and its neighbor p_j.

For each node v_i, the final embedding for the node feature is stored as

$$h_i = (pos_i, \tilde{h}_i) = (pos_i, \text{FPFH}(v_i), ohe_i)$$

where $ohe_i \in \{0, 1\}^N$ is the one-hot encoding of the landmarks in the graph.

The edge characterization mostly relies on geodesic paths. As described in [37], the fundamental concept for computing a geodesic path between two nodes involves iterative execution of edge flips, similar to the traditional Delaunay flip algorithm. However, this approach does not provide constant path dimensions since they depend on the number of mesh points crossed. To address this challenge, we opt to interpolate the points along the various geodesic paths using a 3-spline interpolation. After that, q equidistant points are uniformly sampled for each path and concatenated, producing the embedding

$$k_{ij} = \text{Flat}\left(\text{Spline}(v_i, v_j)\right), \tag{23}$$

where $\text{Spline}(v_i, v_j) \in \mathbb{R}^{3\times q}$ and $\text{Flat} : \mathbb{R}^{3\times q} \to \mathbb{R}^{3q}$ is an operator that concatenates the points sampled from splines. In all our experiments, the number of points sampled on the splines in Eq. (23) is set to $q = 50$.

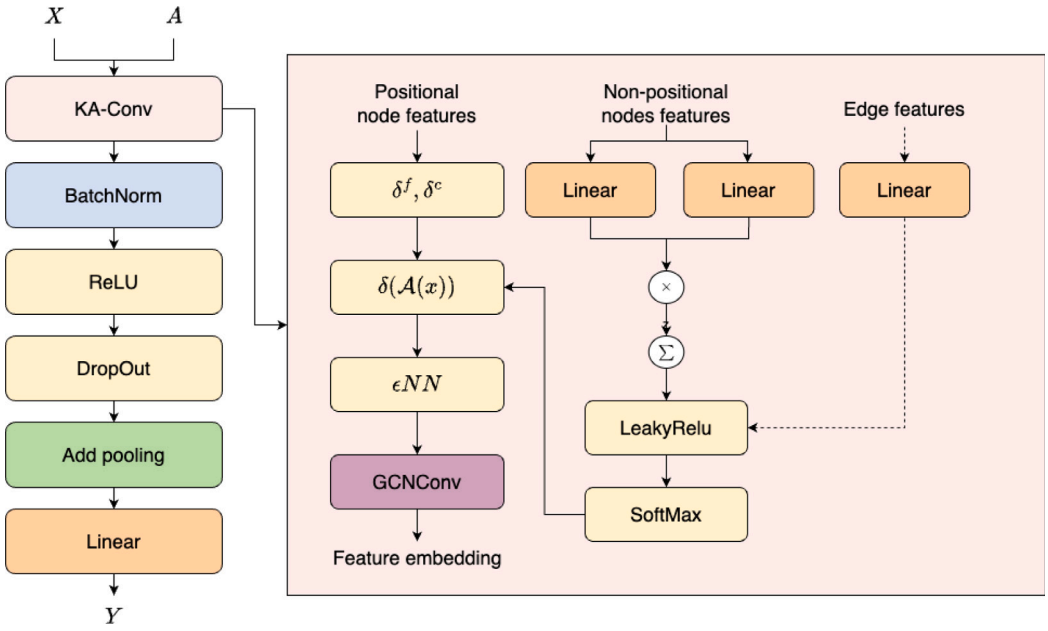

Fig. 2. Architecture of the KA-GCN model. Features X and the adjacency matrix A are dynamically updated in the KA-Conv layer. Positional features are exploited by kernels 'close' and 'far' δ^f, δ^c and integrated into $\delta(\mathcal{A}(x))$ (Eq. (8)) with the attention coefficients calculated on the non-positional features. An epsilon-nearest neighbors operation ϵNN followed by a graph convolutional layer (GCNConv) complete the layer. The overall architecture incorporates batch normalization, a ReLU activation function, and a dropout layer. A pooling operator is then applied to produce the final embedding, which is subsequently used by a linear layer for classification.

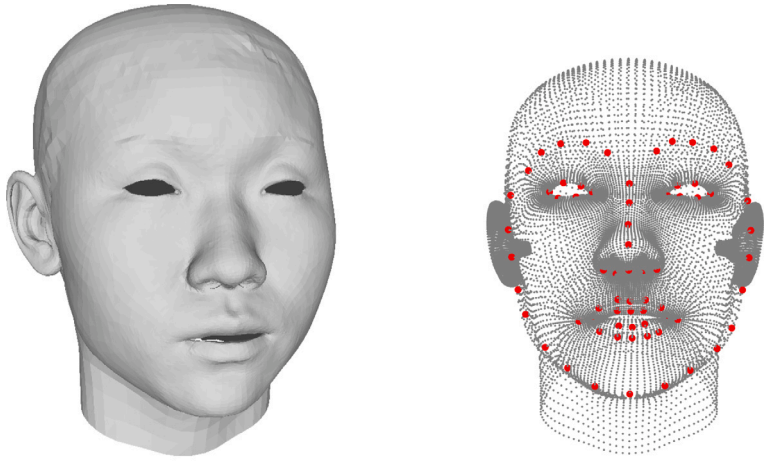

Fig. 3. (a) 3D mesh sample from FaceScape; (b) Facial landmarks of a sample from FaceScape.

5. Experimental results

In the experimental session, we test KA-GCN on three datasets: FaceScape [24], Headspace [38] and Florence [39]. We verify the model on different tasks (sexual dimorphism, emotion and action unit recognition, age estimation) and compare it with models that employ cutting-edge attentive convolutional layers (such as Transformer and GAT) as well as state-of-the-art point-cloud GNN layer. Moreover, parameter tuning and ablation studies are provided to investigate the impact of different configurations on the model's performance.

5.1. Datasets and experiment setup

The **FaceScape** [24] dataset provides textured 3D faces obtained from 938 subjects, and among them, 823 subjects have a complete acquisition including samples of four primary facial expressions (neutral, sadness, smile, and anger), and seven action units (dimpler, chin raiser, grin, brow raiser, brow lower, lip puckerer, lip funneler), together with sexual dimorphism information. This enables us to conduct three types of analysis: the binary identification of sexual dimorphism, detecting emotions across four categories, and recognizing action units across seven groups. The data set is provided with 68 facial landmarks automatically extracted from each face. Fig. 3 shows a representative subject example with the corresponding mesh and the landmarks extracted on the face.

The **Headspace** dataset [38] consists of 3D head images from 1519 individuals, each wearing a tight-fitting latex cap to minimize hairstyle-related imaging artifacts. For this study, we selected samples that included available age and sexual dimorphism data. This selection

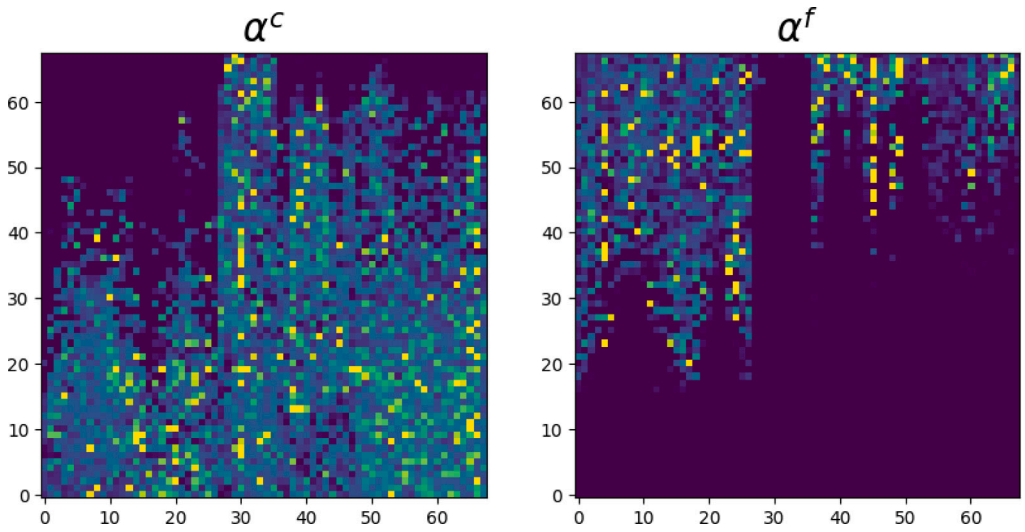

Fig. 4. The coefficients α_{ij}^c (left) and α_{ij}^f (right) from Eq. (12), representing the adjacency matrix learned by KAConv. In both images, the coefficients in each column refer to each node weight, with the progressive index of the node on the x-axis. In particular, the vertical ordering is determined by the distance each node has with all the others, and ranges from the closest (bottom) to the farthest (top).

process resulted in a subset of 1163 samples used for sexual dimorphism and age classification. Additionally, the dataset includes 68 facial landmarks automatically extracted from each face.

The **Florence** dataset [39] is composed of 3D face meshes from 51 individuals, each represented by two frontal views. Our study focuses on sexual dimorphism classification. From each sample, 84 facial landmarks were extracted using the Deep-MVLM method [40].

5.2. Implementation details and ablation studies

Given the distinct characteristics inherent to each task, we performed thorough parameter tuning to identify the most suitable model for each scenario. We have chosen not to apply the ϵ-threshold to make use of all the possible information from the data.

Cross-entropy was selected as the loss function for all tasks, except for the age task for Headspace, for which the Mean Square Error loss has been chosen. The models were optimized using the Adam algorithm with a learning rate of $lr = 0.001$. Both the training and testing datasets were processed in batches of 32. Each model was trained using 5-fold cross-validation over 100 epochs.

To evaluate the significance of both nearby and distant nodes, we conduct ablation studies comparing the proposed KA-GCN model with two different version: the c-KA-GCN model, where the attention coefficients are represented as $\alpha_{ij} = \alpha_{ij}^c$ in Eq. (15), and the f-KA-GCN model, where the attention coefficients are represented as $\alpha_{ij} = \alpha_{ij}^f$.

Table 1 presents the results of the ablation studies conducted on the FaceScape, Headspace and Florence datasets for generic 3D face tasks. On the FaceScape and Florence datasets, KA-GCN consistently outperforms the other variants. Nevertheless, all models exhibit efficient performance, providing flexibility in model selection based on computational constraints. A similar trend is observed with the Headspace dataset, even if here the presence of both far and close nodes becomes relevant to obtain high performances. This reaffirms our initial assumption that both nearby and distant nodes contribute valuable information, especially in datasets with few samples. Additionally, we analyzed the impact of the parameters ζ_i and ξ_i in Eqs. (9) and (10). Fixing the parameters across all nodes (i.e., setting $\zeta_i = \zeta$ and $\xi_i = \xi$ for all $i \in N$) leads to a new model, ζ-KA-GCN, where the balance between the close and far kernels is still learned but uniform across all nodes. This modification results in a significant performance decline, reinforcing the idea that each node requires a tailored value of ζ and ξ to account for its specific neighbors. Fig. 5 shows the learned coefficients $\{\xi_i, \zeta_i\}_{i=1}^N$ for the action unit, sexual dimorphism, and emotion tasks within the FaceScape dataset: for each node in these tasks, the value transition is smooth, starting from 0.5, indicating a generally stable but essential shift.

To better understand the contributions of the kernels 'close' and 'far', Fig. 4 presents an example of the attention coefficients, α_{ij}^c and α_{ij}^f, obtained from the kernel 'close' (left) and the kernel 'far' (right) in the sexual dimorphism classification task on the FaceScape dataset. To distinguish the individual contribution of far and close nodes, the values on each column are referenced to each node, say node i, and ordered in an ascending manner from bottom to top according to the distance that node i has with all other nodes. Therefore, the intensity at position (i, j) in both images is defined on the basis of the distance measures applied on node j (on the jth row) and node i (on the ith column). The final adjacency matrix is obtained by applying the learned weighted sum of these two components.

5.3. Comparison with SOTA models

Comparisons on multiple 3D face analysis tasks on different datasets have been established to evaluate KA-Conv against state-of-the-art convolutional layers involved in graph structure learning with attention mechanisms. In particular, we have chosen a GAT convolution [17] and a Transformer convolution [18] adapted for graph structures, to confront the model with baseline attentive layers. Moreover, we validate our proposal against state-of-the-art layers that consider both attention and position of the nodes during the message passing: EdgeConv [3], DynamicEdgeConv [3], PointNetConv [31], PointGNNConv [32], PointTransformerConv [33], and XConv [34]. Furthermore, a baseline graph convolution [35] has been implemented too. To explicitly highlight the impact of each convolutional layer, all models have been constructed by sharing the architecture, as described in Section 3.2, with the convolutional layer customized accordingly. The hyperparameters for each model have been tuned to identify the optimal configuration for each of them in the same fashion of Section 5.2. Models trained on the FaceScape dataset were evaluated using a 5-fold Cross-Validation method across 100 epochs. The same approach has been adopted for the Headspace and the Florence datasets.

The comparison presented in Table 2 demonstrates that KA-GCN achieves the best performance across all datasets and tasks, with the exception of the sexual dimorphism classification task on the Florence dataset, which still performs well, ranking second to DynamicEdgeConv.

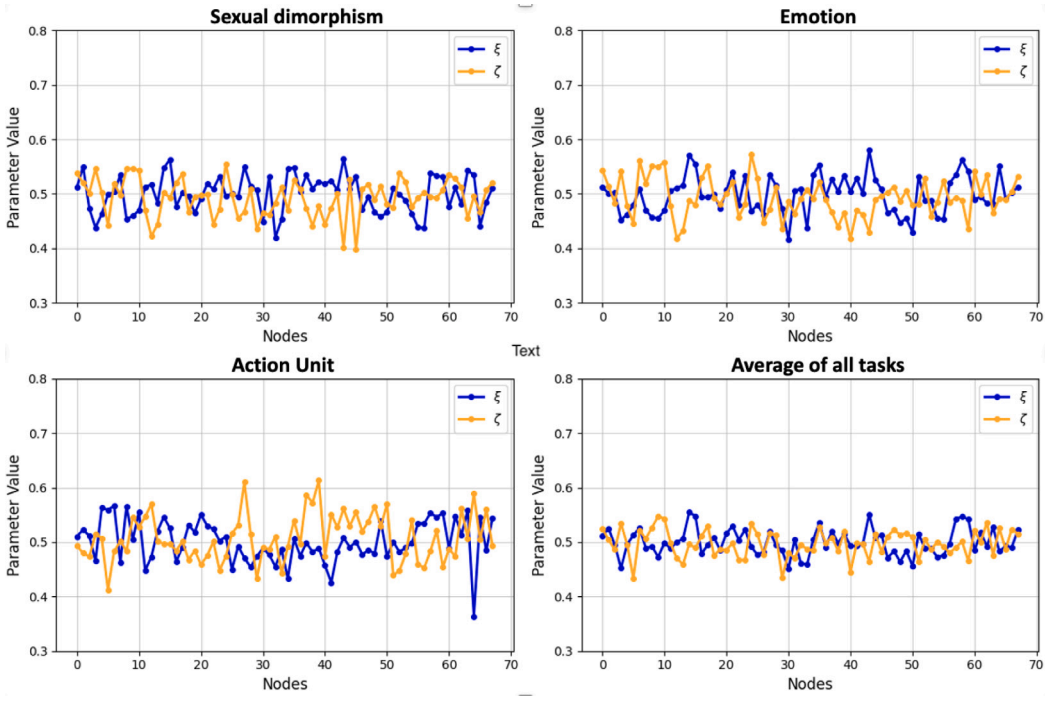

Fig. 5. The learned parameters from Eqs. (9) and (10), representing the shift for the threshold ξ_i and ζ_i. The figures show the three tasks analyzed on the FaceScape dataset together with the mean values on all tasks.

Table 1
Ablation studies on KA-Conv using the FaceScape, Headspace, and Florence datasets for various tasks. The results are presented in terms of accuracy, except for the age estimation task in the Headspace dataset, where the Mean Square Error (MSE) is reported. For the c-KA-GCN and f-KA-GCN variants, the attention coefficients are modified as $\alpha_{ij} = \alpha_{ij}^c$ and $\alpha_{ij} = \alpha_{ij}^f$, respectively, in Eq. (15). In the case of ζ-KA-GCN, we set $\zeta_i = \zeta$ and $\xi_i = \xi$ for all $i \in N$ in Eqs. (9) and (10).

Dataset	Model	Sex dimorphism ↑	Emotion ↑	Action Unit ↑	Age ↓
FaceScape	f-KA-GCN	91.57 ± 1.69	97.98 ± 1.56	98.81 ± 1.04	–
	c-KA-GCN	91.66 ± 1.43	98.48 ± 2.73	98.77 ± 0.88	–
	ζ-KA-GCN	87.38 ± 1.99	86.86 ± 3.29	92.02 ± 2.59	–
	KA-GCN	**92.67 ± 1.58**	**98.87 ± 1.12**	**98.85 ± 0.97**	–
Headspace	f-KA-GCN	88.74 ± 1.90	–	–	6.97 ± 0.13
	c-KA-GCN	83.81 ± 1.97	–	–	3.31 ± 0.71
	ζ-KA-GCN	82.73 ± 1.16	–	–	2.23 ± 0.20
	KA-GCN	**90.15 ± 3.64**	–	–	**2.10 ± 0.04**
Florence	f-KA-GCN	77.74 ± 3.90	–	–	–
	c-KA-GCN	80.41 ± 1.97	–	–	–
	ζ-KA-GCN	70.73 ± 4.16	–	–	–
	KA-GCN	**81.46 ± 1.51**	–	–	–

6. Discussion and conclusions

This study presents KA-Conv, a new convolutional layer that incorporates node distance into metric spaces using a kernel-attentive approach. In line with the widely studied GSL paradigm, the model can discover graph substructures in 3D face data to enhance GNN predictions on small datasets.

We conducted experiments on limited datasets to examine whether the inductive positional bias supports robust performance despite the constraints of data availability. Ablation studies underscore the crucial role of incorporating both nearby and distant nodes in the analysis, especially with few samples per class. Additionally, they emphasize the importance of allowing the model sufficient flexibility to determine the optimal balance between close and distant nodes for each specific node.

Performance variations across datasets suggest that the accuracy of landmark localization and the overall quality of the data are key factors influencing the results. Specifically, KA-GCN demonstrates superior accuracy compared to all competitors on the FaceScape and Headspace datasets, where landmarks are provided. On the Florence dataset, where landmarks are automatically extracted, the method still achieves state-of-the-art performance, though it does not reach optimal levels.

Although this study focuses on 3D face analysis, the core principles of KA-GCN are widely applicable to other domains involving graph structures that require dynamic learning or optimization. This is particularly relevant in data-scarce environments where key points can be effectively leveraged. In general, tasks in different domains may benefit from the development of more flexible models that can selectively filter neighbors based on distance generalizing KA-GCN.

Finally, we point out that potential extensions could involve relaxing the current constraints to enable more adaptable clustering of neighbors. For example, employing "bandpass" filters could allow the selection of non-contiguous distance bands rather than focusing solely on the extremes. Alternatively, distance bands deemed insignificant by the trained model could be excluded entirely, offering a complementary strategy to enhance the filtering process.

Table 2
Comparison between our proposal and the competitors on FaceScape, Headspace and Florence datasets for different tasks. All the results report the accuracy obtained, except for the age task in Headspace, for which the Mean Square Error is reported. The value reported in magenta is the best value, while the second one is in blue.

Dataset	Model	Sex dimorphism ↑	Emotion ↑	Action Unit ↑	Age ↓
FaceScape	GCNConv [35]	82.28 ± 4.99	76.16 ± 5.71	71.02 ± 6.24	–
	GATConv [17]	78.09 ± 6.26	66.16 ± 5.07	63.41 ± 9.55	–
	TransformerConv [18]	88.59 ± 5.91	87.03 ± 6.16	79.56 ± 9.12	–
	EdgeConv [3]	91.33 ± 2.92	89.02 ± 2.52	87.99 ± 2.3	–
	DEdgeConv [3]	**92.29 ± 3.10**	**91.91 ± 2.11**	93.85 ± 1.34	–
	PointConv [31]	90.41 ± 1.56	90.04 ± 0.71	**93.92 ± 1.37**	–
	PointGNNConv [32]	84.96 ± 1.74	85.23 ± 1.04	86.89 ± 2.42	–
	PointTConv [33]	91.13 ± 2.99	89.76 ± 2.62	93.25 ± 2.34	–
	XConv [34]	91.53 ± 2.13	90.27 ± 3.19	93.63 ± 1.99	–
	KAConv	**92.67 ± 1.58**	**98.87 ± 1.12**	**98.85 ± 0.97**	–
Headspace	GCNConv [35]	66.83 ± 1.55	–	–	3.17 ± 0.56
	GATConv [17]	72.12 ± 1.70	–	–	**2.34 ± 0.23**
	TransformerConv [18]	87.56 ± 5.32	–	–	2.39 ± 0.26
	EdgeConv [3]	**89.33 ± 0.80**	–	–	8.79 ± 0.15
	DEdgeConv [3]	84.36 ± 0.86	–	–	9.52 ± 0.14
	PointConv [31]	82.71 ± 0.55	–	–	2.35 ± 0.22
	PointGNNConv [32]	64.10 ± 0.70	–	–	7.64 ± 0.15
	PointTConv [33]	82.71 ± 0.54	–	–	9.14 ± 0.14
	XConv [34]	74.02 ± 0.86	–	–	7.03 ± 0.13
	KAConv	**90.15 ± 0.36**	–	–	**2.10 ± 0.04**
Florence	GCNConv [35]	65.88 ± 4.55	–	–	–
	GATConv [17]	73.17 ± 0.93	–	–	–
	TransformerConv [18]	71.35 ± 4.99	–	–	–
	EdgeConv [3]	67.44 ± 2.99	–	–	–
	DEdgeConv [3]	**83.36 ± 1.47**	–	–	–
	PointConv [31]	77.08 ± 3.79	–	–	–
	PointGNNConv [32]	66.66 ± 6.42	–	–	–
	PointTConv [33]	76.30 ± 5.54	–	–	–
	XConv [34]	70.05 ± 11.13	–	–	–
	KAConv	**81.46 ± 1.51**	–	–	–

CRediT authorship contribution statement

Francesco Agnelli: Writing – original draft, Validation, Software, Methodology, Conceptualization. **Giuseppe Facchi:** Writing – review & editing, Software, Resources, Methodology, Data curation. **Giuliano Grossi:** Writing – review & editing, Supervision, Investigation, Formal analysis, Conceptualization. **Raffaella Lanzarotti:** Writing – review & editing, Supervision, Formal analysis, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to review and refine the grammar. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw. Learn Syst 2020;32(1):4–24.

[2] Xu D, Zhu Y, Choy CB, Fei-Fei L. Scene graph generation by iterative message passing. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 5410–9.

[3] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph cnn for learning on point clouds. ACM Trans Graph (Tog) 2019;38(5):1–12.

[4] Wu L, Chen Y, Shen K, Guo X, Gao H, Li S, et al. Graph neural networks for natural language processing: A survey. Found Trends® Mach Learn 2023;16(2):119–328.

[5] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. AI Open 2020;1:57–81.

[6] Shlomi J, Battaglia P, Vlimant J-R. Graph neural networks in particle physics. Mach Learning: Sci Technol 2020;2(2):021001.

[7] Li R, Yuan X, Radfar M, Marendy P, Ni W, O'Brien TJ, et al. Graph signal processing, graph neural network and graph learning on biological data: a systematic review. IEEE Rev Biomed Eng 2021.

[8] Jiang W, Luo J. Graph neural network for traffic forecasting: A survey. Expert Syst Appl 2022;207:117921.

[9] Gao C, Zheng Y, Li N, Li Y, Qin Y, Piao J, et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. ACM Trans Recomm Syst 2023;1(1):1–51.

[10] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2008;20(1):61–80.

[11] Luo D, Cheng W, Yu W, Zong B, Ni J, Chen H, et al. Learning to drop: Robust graph neural network via topological denoising. In: Proceedings of the 14th ACM international conference on web search and data mining. 2021, p. 779–87.

[12] Jin W, Ma Y, Liu X, Tang X, Wang S, Tang J. Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020, p. 66–74.

[13] Zhu Y, Xu W, Zhang J, Liu Q, Wu S, Wang L. Deep graph structure learning for robust representations: A survey. 14, 2021, arXiv preprint arXiv:2103.03036.

[14] Patania S, Boccignone G, Buršić S, D'Amelio A, Lanzarotti R. Deep graph neural network for video-based facial pain expression assessment. In: Proceedings of the 37th ACM/SIGAPP symposium on applied computing. 2022, p. 585–91.

[15] Chen C, Wu Y, Dai Q, Zhou H-Y, Xu M, Yang S, et al. A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. IEEE Trans Pattern Anal Mach Intell 2024.

[16] Fatemi B, El Asri L, Kazemi SM. SLAPS: Self-supervision improves structure learning for graph neural networks. Adv Neural Inf Process Syst 2021;34:22667–81.

[17] Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y, et al. Graph attention networks. Stat 2017;1050(20):10–48550.

[18] Shi Y, Huang Z, Feng S, Zhong H, Wang W, Sun Y. Masked label prediction: Unified message passing model for semi-supervised classification. 2020, arXiv preprint arXiv:2009.03509.

[19] Zhou S, Xiao S. 3D face recognition: a survey. Human- Centric Comput Inf Sci 2018;8(1):1–27.

[20] Sandbach G, Zafeiriou S, Pantic M, Yin L. Static and dynamic 3D facial expression recognition: A comprehensive survey. Image Vis Comput 2012;30(10):683–97.
[21] Burger J, Blandano G, Facchi GM, Lanzarotti R. 2S-SGCN: A two-stage stratified graph convolutional network model for facial landmark detection on 3D data. Comput Vis Image Underst 2024;104227.
[22] Savran A, Alyüz N, Dibeklioğlu H, Çeliktutan O, Gökberk B, Sankur B, et al. Bosphorus database for 3D face analysis. In: Biometrics and identity management: first European workshop, BIOID 2008, roskilde, Denmark, May 7-9, 2008. revised selected papers 1. Springer; 2008, p. 47–56.
[23] Zhang Z, Girard JM, Wu Y, Zhang X, Liu P, Ciftci U, et al. Multimodal spontaneous emotion corpus for human behavior analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 3438–46.
[24] Yang H, Zhu H, Wang Y, Huang M, Shen Q, Yang R, et al. FaceScape: a large-scale high quality 3D face dataset and detailed riggable 3D face prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2020.
[25] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Adv Neural Inf Process Syst 2017;30.
[26] Sforza C, Grandi G, De Menezes M, Tartaglia GM, Ferrario VF. Age-and sex-related changes in the normal human external nose. Forensic Sci Int 2011;204(1–3):205–e1.
[27] Gupta S, Markey MK, Bovik AC. Anthropometric 3D face recognition. Int J Comput Vis 2010;90:331–49.
[28] Loconsole C, Miranda CR, Augusto G, Frisoli A, Orvalho V. Real-time emotion recognition novel method for geometrical facial features extraction. In: 2014 international conference on computer vision theory and applications, vol. 1, IEEE; 2014, p. 378–85.
[29] Kukharev G, Kaziyeva N. Digital facial anthropometry: application and implementation. Pattern Recognit Image Anal 2020;30:496–511.
[30] Li R, Wang S, Zhu F, Huang J. Adaptive graph convolutional neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol. 32, (1). 2018.
[31] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.
[32] Shi W, Rajkumar R. Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 1711–9.
[33] Zhao H, Jiang L, Jia J, Torr PH, Koltun V. Point transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 16259–68.
[34] Li Y, Bu R, Sun M, Wu W, Di X, Chen B. Pointcnn: Convolution on x-transformed points. Adv Neural Inf Process Syst 2018;31.
[35] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. 2016, arXiv preprint arXiv:1609.02907.
[36] Rusu RB, Blodow N, Beetz M. Fast point feature histograms (FPFH) for 3D registration. In: 2009 IEEE international conference on robotics and automation. IEEE; 2009, p. 3212–7.
[37] Sharp N, Crane K. You can find geodesic paths in triangle meshes by just flipping edges. ACM Trans Graph 2020;39(6).
[38] Dai H, Pears N, Smith W, Duncan C. Statistical modeling of craniofacial shape and texture. Int J Comput Vis 2020;128(2):547–71.
[39] Bagdanov AD, Del Bimbo A, Masi I. The florence 2d/3d hybrid face dataset. In: Proceedings of the 2011 joint ACM workshop on human gesture and behavior understanding. 2011, p. 79–80.
[40] Paulsen RR, Juhl KA, Haspang TM, Hansen T, Ganz M, Einarsson G. Multi-view consensus CNN for 3D facial landmark placement. In: Asian conference on computer vision. Springer; 2018, p. 706–19.