Graph Edge Classification for Keypoint Grouping in Multi-person Pose Estimation

Marwan Maher*, Radwa Fathalla, and Mohamed Shaheen

Department of Computer Science, College of Computing and Information Technology, Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt
Email: mrwn6@proton.me (M.M.); radwa_fathalla@aast.edu (R.F.), mohamed.shaheen@aast.edu (M.S.)
*Corresponding author

***Abstract*—Human pose estimation is an essential component of computer vision systems involving human activity, as it is concerned with predicting the configuration of the human body in 2D or 3D coordinates. The pose is expressed as related keypoints representing body parts. We consider the case of 2D bottom-up pose estimation, where the location of identity-free keypoints is predicted and then grouped into individual persons. In contemporary work, while the keypoint prediction process is learnable and automated, keypoint-grouping still largely relies on non-learnable optimization algorithms operating in embedding space, grouping similar keypoints regardless of the resulting pose structure. To overcome this limitation, this paper presents the Graph Edge Classifier (GEC), a novel, learnable keypoint-grouping method. In GEC, predicted keypoints are represented as a graph, where each keypoint is connected to all potentially related keypoints via edges. The main objective is to classify edges as either connected or not connected. The method consists of three components: A Graph Neural Network (GNN) encoder for node and edge feature learning, an edge classifier network (decoder), and a post-processing step. Additionally, we introduce two novel update functions for node and edge features within the message-passing neural network framework. Our method achieves an Average Precision (AP) score of 46.1% on the CrowdPose test set, which is comparable to similar bottom-up methods. Moreover, our model is lightweight, with only 0.274 million parameters, making it more efficient in terms of computational resources compared to other learnable keypoint-grouping methods. The learnable, efficient, and structure-aware nature of our approach offers potential for further improvement, especially through integrated end-to-end training of both the backbone and grouping networks.**

***Keywords*—2D human pose estimation, keypoint grouping, message-passing neural network, graph neural networks, edge classification**

I. Introduction

Human Pose Estimation (HPE) is a critical task in machine-to-human interface systems, facilitating video-based human body recognition by machines and enabling a wide range of applications. Numerous studies have utilized HPE methods for various purposes, including human-robot interaction [1], health care [2], sports analytics [3], computer animations [4], and surveillance [5].

HPE aims to localize a set of predefined semantic keypoints (e.g., head, left elbow, or right knee) that represent a kinematic model of the human body. This provides approximate information describing the pose, orientation, and relative configuration of a person's body parts. Classical HPE methods relied on custom handmade feature extractors to detect and locate keypoints in images [6]. However, due to the ability of Deep Neural Networks (DNNs) to automatically learn higher-order features, deep-learning-based feature extractors have become the preferred method for keypoint detection, localization, and grouping in the bottom-up case.

This article focuses on monocular 2D multi-person human pose estimation. There are two main strategies. The first strategy is the top-down approach that begins by detecting all human body instances using an object detection network to find the bounding boxes around humans present in a frame. Then, it processes each instance individually to localize predefined keypoints using either direct Cartesian coordinates regression [7, 8], or 2D Gaussian heatmap-based methods [9]. The second strategy is the bottom-up approach; this method first detects all keypoints without assigning to them an identity and then groups them into individual persons using various techniques. These include part affinity-field [10, 11], where a greedy algorithm maximizes a path integral over a 2D vector field, root joint regression [12] which represents person instances with keypoint positions related to their respective roots via an offset relative to the center joint, Associative Embedding (AE) maps [13, 14], which involve pixel-wise vector embeddings, ensuring pixels belonging to the same person have high cosine similarity while those belonging to different persons have low similarity.

Graphs are widely used to represent diverse forms of data, such as transportation networks and protein structures. Problems that can be represented as graphs could benefit from computational tasks on graphs, such as node/edge attribute inference and graph clustering. Traditionally, features were engineered based on statistics

Manuscript received July 9, 2024; revised August 12, 2024; accepted September 26, 2024; published March 14, 2025.

doi: 10.18178/joig.13.2.130-139

or properties like node degree or homophily. However, automatic feature learning has become more common, with Graph Neural Networks (GNNs) forming the foundation of modern graph processing. GNNs generalize both spectral graph filters, which use the graph Laplacian's spectral decomposition to perform convolution in the frequency domain, and spatial graph filters, which perform local message-passing among nodes. The use of GNNs has gained traction in HPE literature [15–20], particularly in bottom-up multi-person HPE, as graphs are well-suited for modeling the articulated human body through modeling keypoints and their relationships.

Of those bottom-up approaches that utilized GNNs, we observed that edge features are underutilized, which is a missed opportunity. Feature maps such as Associative Embedding (AE) maps in [13] are continuous on their domain (input image), so they contain useful information not only around the keypoints (nodes) but also in the spaces connecting pairs of keypoints (edges).

This article is organized into five sections. The first section introduced HPE, its use cases in computer vision, and an overview of the common approaches used in HPE. The second section summarizes previous work closely related to this study. The third section presents a novel, fully supervised, learnable keypoint grouping method for 2D human pose estimation. Section four presents the results of our method. Lastly, the fifth section concludes the paper. Our main contributions are summarized as follows:

- Reformulating the keypoint-grouping step in bottom-up HPE as a binary classification of graph edges.
- Introducing a lightweight supervised learnable keypoint-grouping method for bottom-up HPE.
- Introducing novel node and edge features update functions into the message-passing neural network layer, incorporating edge features to enhance edge classification accuracy.

II. Related Work

Pictorial Structure (PS) models were first introduced in the 70s [21], which aided object recognition algorithms by reducing the search space for an object by searching for its constituent parts independently. These parts were then combined to form an object based on prior knowledge or constraints. PS models gained popularity with the development of powerful inference algorithms, mainly in object recognition [22] and human pose estimation [23]. Later, advances in convolutional deep neural networks were applied to pose estimation for joint regression [24], yielding absolute coordinate vectors for a single person.

In the current landscape of multi-person HPE, there are two primary approaches: top-down pipelines [25–30] and bottom-up pipelines [11, 13, 14, 27, 31, 32]. Additionally, expressive representation learning of graphs and their components [33, 34] has been integrated into multi-person HPE frameworks due to the natural fit of modeling the articulated human body as a graph. This integration allows for the automatic learning of new graph representations and pattern discovery in graph data, with the help of Graph Neural Networks (GNNs), particularly Message-Passing Neural Networks (MPNNs). Now we briefly discuss related top-down and bottom-up works and graph representation learning applications in 2D HPE.

A. Top-Down

A top-down 2D multi-person HPE pipeline involves two main tasks: a full-body detection module [35], which determines bounding boxes around human bodies in an image, followed by a keypoint localization stage that detects keypoints for a single person within each bounding box. The most common limitations of the top-down approach include: (1) processing N persons in an image requires running the keypoint localization network N times, leading to higher computational costs, due to the repeated processing of overlapping areas; (2) overlapping bounding boxes may contain keypoints from multiple people that the full-body detector failed to identify, making it difficult to exclude them with masking. Despite these challenges, top-down methods often achieve higher average precision scores because the human instance detector provides a rough estimate of where all joints of a person inside a bounding box should be. Sun *et al.* [27] HRNET is a notable top-down approach that uses a multi-resolution Convolutional Neural Network (CNN) and deconvolution layers to generate heatmaps for estimating joint locations. The inclusion of multi-resolution branches improves detections for distant and small individuals.

B. Bottom-up

Bottom-up 2D HPE methods start with detecting all keypoints in an image. The subsequent challenge is grouping these keypoints into individual people, effectively partitioning the set of anonymous joints into classes, each representing a single person. The main appeal of the bottom-up approach is its scalability in crowded scenes, as keypoint detection is performed only once, reducing computational overhead. This makes it more suitable for real-time applications and mobile environments. However, bottom-up methods generally achieve lower accuracy compared to top-down approaches due to the complexity of the keypoint grouping task.

Cao *et al.* [31] proposed a bottom-up method that uses an iterative approach with supervision at each stage. Confidence maps predict joint locations, while part-affinity fields (2D vector fields) relate joints using a greedy algorithm to maximize path integrals over the field. However, the grouping process in this method is not learnable; the greedy algorithm functions as a non-learnable decoder of the part-affinity fields.

More recently, Qiu *et al.* [36] proposed a dynamic graph convolutional network, where a soft adjacent matrix is computed from the dataset, based on the assumption that related keypoints tend to have smaller spatial distances. A dynamic graph is generated based on Bernoulli trials on the entries of the soft adjacency matrix. The learnable parameter W_d and the dynamic graph together are used to update the feature embeddings of the image before they are decoded into keypoint heatmaps and joint relation heatmaps. This approach follows by employing a greedy keypoint-grouping strategy [11].

Jin *et al.* [18] made significant contribution to bottom-up pose estimation by proposing a differentiable online hierarchical graph grouping (clustering) method. This approach builds upon the stacked hourglass architecture and Associative Embedding (AE) introduced in [13, 14]. Their end-to-end trainable network employs a GNN to obtain node embeddings, which are used to iteratively cluster/group keypoints, updating graph connections after each iteration. Despite its innovation, this agglomerative approach has several drawbacks: (1) high computational cost due to the iterative use of a MPNN followed by supervised classifications for edges and macro-nodes while dynamically updating graph connections; (2) Low accuracy in outlier cases (e.g., an image with only upper body keypoints), (3) potential over-smoothing of node features with multiple iterations of MPNN, making the feature vectors indistinguishable. In contrast, our method addresses the keypoint grouping problem through a more straightforward and generalizable binary graph-edge-classification approach. Furthermore, we leverage edge features, which have not been utilized in any of the previous keypoint grouping methods.

C. *Graph Neural Networks*

Deep learning models have demonstrated remarkable versatility in addressing a wide range of tasks across different data formats, including time series and grid-based structures. Similarly, Graph Neural Networks (GNNs) [37–39] have extended the application of deep learning to graph-structured data, via processing graph data with permutation equivariant functions. The foundational concept behind GNNs is the generalization of convolutional operations to non-Euclidean data structures, which allows for the efficient processing and representation learning of graph data.

The process of transforming node and edge features based on the graph structure is referred to as "graph filtering" in the literature. Two main types of graph filters exist: spatial and spectral. Spatial filters are the most relevant in our case. The Message Passing Neural Network (MPNN) [40] framework is a general GNN framework, and many spatial graph filters are a specific case of an MPNN. An MPNN consists of two main components: a message function, which generates messages based on a permutation-equivariant aggregation of a node's neighborhood and the node itself, and an update function, which takes the generated message and the current hidden embedding of a node to produce an updated node feature vector. This iterative message-passing scheme enables MPNNs to effectively capture both local and global node relationships.

The Edge Convolution operator (EdgeConv) [34] was proposed to capture local geometric structures by constructing edge features that describe the relationships between a node and its neighbors. Additionally, [33] formulated a method for edge feature propagation using a learned attention vector to focus on important edge features. In this study we combine aspects from [33, 34] to develop a novel node and edge-feature aggregation and update functions.

III. Proposed Method

A. *Overview*

The proposed bottom-up 2D human pose estimation method consists of three stages: (1) keypoint detection, where we use the HigherHRNET model presented in [14] to localize joint coordinates in the form of a 2D Gaussian heatmap and obtain joint tags. These tags are used as node and edge features for a graph during the keypoint grouping stage. (2) keypoint grouping, a supervised and learnable keypoint grouping stage where a graph is constructed with all possible connections between the appropriate joints according to a chosen connection model. The graph is then processed with a graph-edge-classification network to classify each edge as either connected or not. (3) graph post-processing, which operates on disjoint subgraphs from the output graph iteratively to guarantee no duplicates keypoints in a human instance. The key highlight of our method is the learnability of the keypoint-grouping stage, and the use of edge features. An overview of the proposed method is shown in Fig. 1.

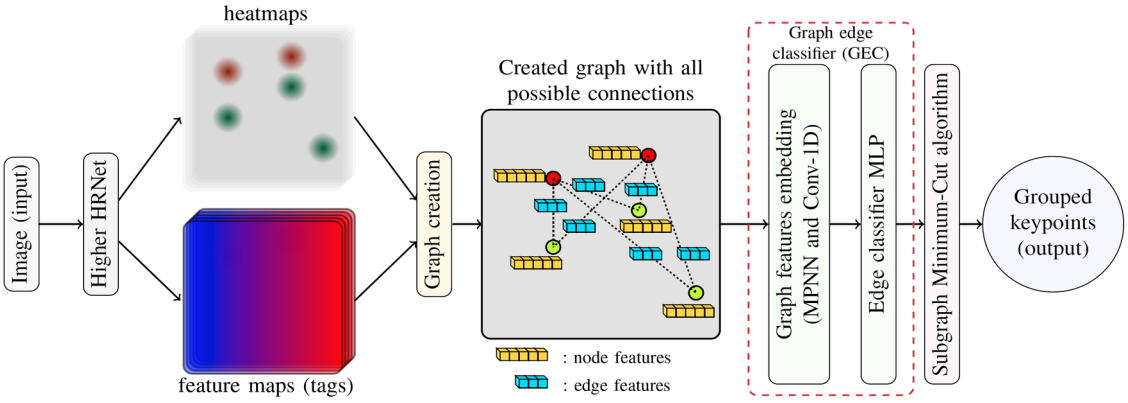

Fig. 1. Overview of our bottom-up pose estimation method. It includes: A backbone network (HRnet) where, features are extracted. Graph creation with node and edge features extracted from the feature maps. The proposed Graph Edge Classifier (GEC) module. Lastly, a post-processing step, where grouped keypoints are output.

B. Keypoint Detection

For keypoint detection we use HigherHRNet [14, 27, 32], a multi-resolution CNN architecture that takes an input image I and outputs predicted keypoint heatmaps $\mathcal{H}$ of size 14×320×320, one for each keypoint type. Additionally, it produces an associative embedding map $\mathcal{T}$ (tagmap), a 2D map where keypoints belonging to the same person have very similar values $\in \mathbb{R}$ (relationship encoding). Mean Squared Error (MSE) loss is used during training. Since keypoint detection is not the main focus of this paper, please refer to [14] for more information. Non-Maximum Suppression (NMS) is applied to the keypoint heatmaps, with a cutoff threshold of 0.1, obtaining 2D coordinates for each keypoint.

C. Keypoint-Grouping

1) Graph construction

A graph $G = \{V, E\}$ is constructed based on a connection model $K = \{(i,j)$ where i,j are joint types$\}$. We consider three connection models: basic, extended, and complete-graph (see Fig. 2). We use these three models during training and compare their performance in the results. Skip connections can improve grouping accuracy due to bypassing occluded intermediary joints.

The created graph has a node set V, that contains all the predicted keypoints at coordinates P. Each node $v_i \in V$ has a node-feature vector x_i which is a concatenation of the normalized Euclidean 2D coordinates p_i, a one-hot-encoding of the keypoint type, and, a 7×7 pixels patch extracted from $\mathcal{T}$ centered at p_i. The edge set $E = \bigcup_{(i,j)\in K} V_i \times V_j$ contains all the edges connecting nodes of type i to all nodes of type j iff $(i,j) \in K$. An edge $e_{ij} \in E$ has edge-feature vector x_{ij} extracted from $\mathcal{T}$ along a line drawn from p_i to p_j. Fig. 1 depicts an overview of our method, including the graph creation process.

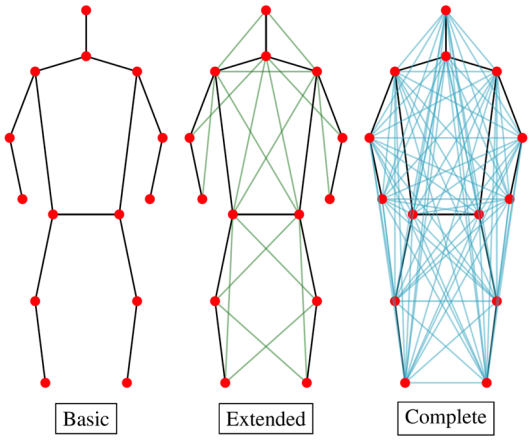

Fig. 2. The three connection types—basic, extended, and complete— that are used to construct the graphs. Nodes represent the detected keypoints, and edges represent the pairwise relationship between two nodes. The goal is to classify the edges into one of two classes.

2) Node and edge features embedding

To carry out edge classification on a graph, we aim to obtain node-feature embeddings such that vectors of keypoints belonging to the same person are closer to each other in embedding space than keypoints of other people. Hence, we introduce a GNN that encodes feature vectors of nodes and edges (see Fig. 3). Each layer of MPNN (referred to as MPNN-# in Fig. 3, where # is the layer number) updates node and edge feature vectors simultaneously. The node-features are updated according to the function:

$$\mathbf{x_i'} = \max_{j \in \mathcal{N}(i)} (\mathrm{ReLU}\,(\mathbf{h_\Theta}\,(\mathbf{x_j} - \mathbf{x_i} \,\|\mathbf{x'_{ij}})) + \mathbf{x_i} \qquad (1)$$

where, $\mathcal{N}(i)$ is the set of neighbors of the node v_i including a self-connection, and, $\boldsymbol{h_\theta}$ is a learnable linear transformation.

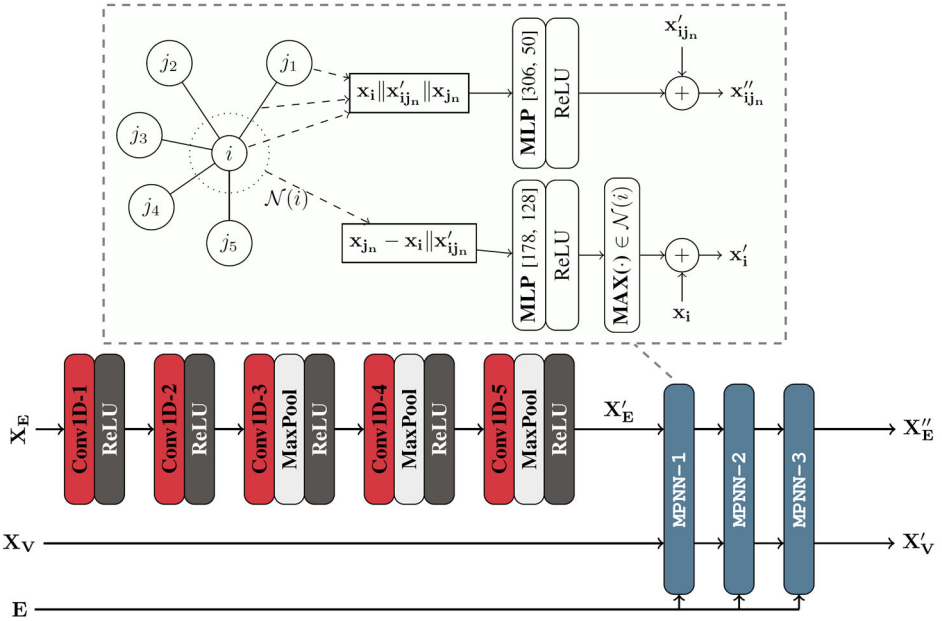

Fig. 3. Graph neural network module, where node and edge features embeddings are obtained. The 1D convolution branch operates on edge features $\mathbf{X_E}$ resulting in intermediate edge features $\mathbf{X'_E}$. Node features $\mathbf{X_V}$ and edge features $\mathbf{X'_E}$ are propagated through message-passing neural network (MPNN) layers, yielding node and edge feature embeddings $\mathbf{X'_V}$ and $\mathbf{X''_E}$.

Initially, edge features $\boldsymbol{x_{ij}}$ are passed through a 1D CNN for feature extraction and dimensionality reduction, obtaining intermediate features $\boldsymbol{x'_{ij}}$. The output from the 1D CNN branch is then used in conjunction with node-features to be propagated through the MPNN layers. Edge-features are updated in each MPNN layer according to the function:

$$\boldsymbol{x''_{ij}} = \mathrm{ReLU}\left(\boldsymbol{W}\left(\boldsymbol{x}_i \| \boldsymbol{x'_{ij}} \| \boldsymbol{x}_j\right)\right) + \boldsymbol{x'_{ij}} \tag{2}$$

where $\boldsymbol{W}$ is a learnable linear transformation. We use three MPNN layers (as shown in Fig. 3) to avoid over-smoothing the signal on the graph, given that the 14-keypoint model of CrowdPose dataset corresponds to a graph diameter of 6.

The rationale for applying 1D convolutions on edge-features extracted from AE maps [13] is as follows: AE maps are continuous 2D functions over their discrete image domain I, as they are the result of pulling pixel values closer at keypoint locations belonging to the same person and pushing pixel values away at the locations of different people. This results in a feature map with pixels forming pseudo-level sets, where keypoints belonging to the same person have similar tag values. Gradual changes between these tag values across keypoint locations represent valuable information, which is why edge-features are considered in the node and edge-feature update functions.

3) Edge classifier

To classify each edge in the input graph, we use an edge-classifier network (see Fig. 4), which takes the concatenation of an edge-feature embedding $\boldsymbol{x''_{ij}}$, the edge's two node-feature embeddings x'_i and x'_j, and extra edge features x^+_{ij} (including the normalized edge length and normalized edge angle $\theta \in (-\pi, \pi]$ from the positive x direction to the right). The output of the edge-classifier network is an array of $|\mathrm{E}| \times 2$, with each row representing the probability of an edge belonging to one of the two classes. This stage can be seen as a decoder for the embedded features.

4) Loss function

The cross-entropy loss function is commonly used for supervised classification tasks. It is a measure of how well a predicted probability (q) approximates a true probability (p), with lower values -approaching zero- being better. It is defined as $H(p,q) = H(p) + D_{KL}(p \parallel q)$, where $H(p) = \sum_{x \in X} p(x) log(\frac{1}{p(x)})$ is the entropy of probability distribution p, which is a measure of the expected surprise of a discrete event X with probability distribution p, and, the Kullback-Leibler divergence $D_{KL}(p \parallel q) = \sum_{x \in X} p(x) log(\frac{q(x)}{p(x)})$ that measures how much one distribution diverges from a reference distribution. We use a weighted cross-entropy loss function $(\mathcal{L})$ (Eq. (3)), to mitigate the effects of class imbalance that occurs due to considering all possible connections between the keypoints. For example, when considering the edges between a head keypoint and all neck keypoints, only one edge is labeled as connected and all others are labeled as disconnected. This disparity in class ratios can cause the network to favor negative classification. Hence, the weighted cross-entropy loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = \frac{1}{|\mathrm{N_e}|} \sum_{\mathrm{i}=1}^{\mathrm{N_e}} \sum_{\mathrm{k}=1}^{2} -\mathrm{w_k} \times \mathrm{y_k^i} \times \log\left(\mathrm{p_k^i}\right) \tag{3}$$

where p^i and y^i are the prediction and target probability vectors for an edge, k is the class, and N_e is the set of edges in a batch. Weight vector $w = \left[1, \frac{N_-}{N_+}\right]$ is calculated from the frequencies of the positive and negative classes in the dataset.

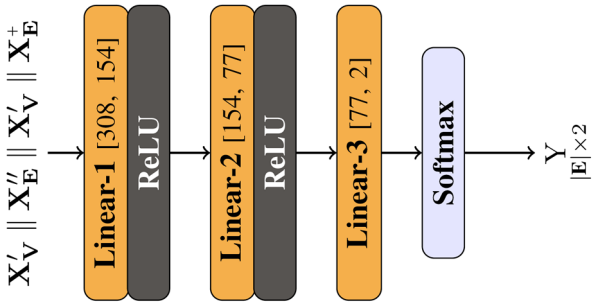

Fig. 4. Edge classifier network. The inputs for an edge (i, j), inputs are: edge feature $\mathbf{x''_{ij}}$ concatenated with its two node features $\mathbf{x_i}$ and $\mathbf{x_j}$, with extra edge features $\mathbf{x^+_{ij}}$. The output Y is the probability vectors for the classes of each edge.

D. Graph Post-Processing

For an input graph G, the ideal output is a set of disjoint subgraphs S, where each subgraph $s \in S$ corresponds to the keypoints of one labeled instance, meaning that each subgraph contains at most one of each joint type. However, due to the imperfection of edge classification, a situation might arise in which subgraph $H \subset G$ contains more than one instance of a joint type (e.g., two heads), or a fusion of two or more people because a joint is connected to two or more instances of the same joint type. To address this, we introduce an edge pruning algorithm (see Algorithm 1) based on the minimum-cut algorithm [41]. The function INIT_WEIGHTS initializes all the edge weights of the graph (if they exist) based on cosine similarity $w_{ij} = d\left(\boldsymbol{x'_i}, \boldsymbol{x'_j}\right) = \frac{x'_i \cdot x'_j}{\|x'_i\| \, \|x'_j\|}$ between features of its two nodes. SUBGRAPHS finds all connected components of the graph (subgraphs) creating new graph for each component. Subgraphs are processed from a queue, one at a time. HAS_DUPLICATES checks if a subgraph has duplicate keypoint types. Nodes of the subgraph are sorted by degree, and GET_DUPLICATES returns the first duplicate pair from the sorted node list. The MINCUT function partitions the subgraph and returns two new subgraphs with the removed edges. REMOVE_EDGES prunes edges from the main graph G. If any of the two resulting subgraphs SG1 and SG2 have duplicates, then they also get added to the processing queue, otherwise process the next subgraph (Algorithm 1).

Algorithm 1: Min-cut for subgraph post-processing

```
Require: |G| > 1
    INIT_WEIGHTS(G)
    SGS ← SUBGRAPHS(G)
    for all SG ∈ SGS do
        Q.ENQUEUE(SG)            ▷ Q: subgraphs queue
        while Q.length ⩾ 1 do
            SG_t ← Q.DEQUEUE()
            if HAS_DUPLICATES(SG_t) then
                sorted_SG ← DEGREE_SORT(SG_t)
                (s,t) ← GET_DUPLICATES(sorted_SG)
                (SG1, SG2, edges)←MINCUT(s, t, SG_t)
                REMOVE_EDGES(EDGES, G)
                if HAS_DUPLICATES(SG1) then
                    Q.ENQUEUE(SG1)
                end if
                if HAS_DUPLICATES(SG2) then
                    Q.ENQUEUE(SG2)
                end if
            end if
        end while
    end for
```

IV. Results

In this section, we detail the dataset used, discuss the training hyperparameters, present comparisons between notable bottom-up human pose estimation methods and our approach, and list the software and hardware used to ensure reproducibility.

A. Datasets

Training and testing were conducted on the CrowdPose dataset [42], which consists of 20,000 images and 80,000 human instances. The author of the dataset split it into 3 sets; 10,000 images for training, 2,000 for validation, and 8,000 for testing. We trained our models using both the training and validation sets and evaluated them on the test set. This split strategy is consistent with previous studies, allowing for a direct comparison of results. To prepare the CrowdPose labels for the Graph Edge Classifier (GEC) module, we applied a bipartite matching algorithm [43] between the predicted keypoints and the labeled joints in the dataset. Each predicted joint was matched to the nearest labeled joint of the same type and vice versa. Based on these assignments, an array of size $|E| \times 2$ was created, where each row represents a binary classification of the graph's connected edges (see Fig. 5).

B. Training

The training of our Graph Edge Classifier network is fully supervised, where each edge in the constructed graph is assigned a label indicating whether it is connected or not. During the forward pass, HigherHRNet-W48 [14] is used as the backbone network for keypoint detection, generating both keypoint heatmaps and tag maps A graph is then constructed from the detected keypoints, with node and edge features extracted at their corresponding locations in the tag maps. Labels are created based on the connections in the graph and the proximity of the detected keypoints to the ground truth. Finally, the GEC network groups the keypoints into human instances.

HigherHRNet was trained for 320 epochs using the Adam optimizer with a learning rate of 10^{-3}, as recommended in [14]. Our keypoint grouping network was trained separately for 100 epochs, also using the Adam optimizer but with a base learning rate of 10^{-5} and a multi-step learning rate scheduler with a decay factor $\gamma = 0.5$ with three milestones. Although the full architecture is end-to-end trainable, we trained the keypoint detection and keypoint grouping networks separately due to computational constraints.

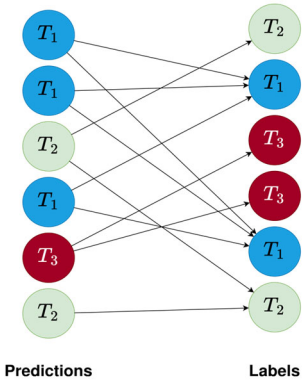

Fig. 5. Minimum weight bipartite matching for creating edge labels, where edge weights are Euclidean distances between two nodes. Only keypoints of the same type are connected, the closest pair are matched.

C. Evaluation metrics and Results

We used the MS COCO evaluator to assess the performance of the GEC model. The primary metric for evaluating Human Pose Estimation (HPE) models trained on COCO or its derivatives is Average Precision (AP). AP is calculated from the Object Keypoint Similarity (OKS) of joint detections at various thresholds, which is analogous to the Intersection over Union (IoU) metric used in object detection for bounding boxes. OKS ranges between [0, 1]. AP represents the average precision across ten OKS values ranging from 0.5 to 0.95. AP^{50} and AP^{75} represent precision at OKS thresholds of 0.5 and 0.75, respectively. Finally, the accuracy of the GEC network was computed for each experiment, with the results summarized in Table I. Evaluation results are reported on the CrowdPose test set, which was not seen by the model during training.

The findings in Table I reveal two key patterns in the AP column. First, increasing graph connectivity improves precision scores. This can be elucidated by considering a node $v \in V$ and two communities of nodes, V_A and V_B, belonging to person A and person B, connected by edge sets E_A and E_B, respectively. As the connections increase, the classifier has more chances of relating v to its correct more densely connected community and vice versa. With more connections, the classifier is more likely to correctly associate node v with its appropriate, more densely connected community. As a result, the classifier learns to favor nodes with a higher clustering coefficient (cc) [44], a measure of the node's cliquishness with its neighbors

Essentially, the classifier prefers connecting a node to a set of nodes that are well-connected among themselves and avoids connections that would lower the cc value. Additionally, increased graph connectivity helps to bypass the issue of joint occlusion by directly linking distant keypoints without depending on intermediate junction keypoint detections. Finally, the inclusion of edge features enhances AP results across all connection variants of the proposed GEC model, as it enriches the classifier's input by considering the feature space between connected nodes.

TABLE I. COMPARISON OF VARIOUS BOTTOM-UP HUMAN POSE ESTIMATION METHODS AND THE PROPOSED GRAPH EDGE CLASSIFIER (GEC) NETWORK. EACH METHOD IS EVALUATED ON THE CROWDPOSE TEST SET AND ITS AVERAGE PRECISION (AP) SCORES ARE REPORTED. THE "EDGE FEATURES" COLUMN INDICATES WHETHER EDGE FEATURES WERE UTILIZED (✓) OR NOT (✗) IN EACH EXPERIMENT

Method	Edge features	Connection model	Backbone	Input size	AP	AP^{50}	AP^{75}
LitePose-S* [45]	N/A	N/A	MobileNetV2	448×448	58.3	81.1	61.8
LitePose-XS* [45]	N/A	N/A	MobileNetV2	256×256	49.5	74.5	51.4
CenterGroup* [46]	N/A	N/A	HRNet-W48	640×640	67.6	87.7	72.7
CoupledEmbedding* [47]	N/A	N/A	HRNet-W32	512×512	68.9	89.0	74.2
CoupledEmbedding* [47]	N/A	N/A	HRNet-W48	640×640	**70.1**	**89.8**	**75.6**
HigherHRNet [14]	N/A	N/A	HRNet-W48	640×640	59.4	82.8	62.4
GEC-basic (proposed)	✗	Basic	HRNet-W48	640×640	37.4	61.9	36.3
GEC-basic (proposed)	✓	Basic	HRNet-W48	640×640	40.5	66.7	39.2
GEC-extended (proposed)	✗	Extended	HRNet-W48	640×640	39.1	62.0	38.4
GEC-extended (proposed)	✓	Extended	HRNet-W48	640×640	45.0	69.5	45.8
GEC-complete (proposed)	✓	Complete graph	HRNet-W48	640×640	46.1	69.7	47.8

Note: N.B., methods marked by * are reported as in GRAPE [48].

Additionally, we compare our GEC model to other bottom-up methods with learnable keypoint-grouping components, with respect to the number of parameters and Multiply-Accumulate Operations (MACs). This comparison exclusively evaluates the keypoint-grouping modules of each method. The results are presented in Table II. While the number of MACs for CenterGroup [46] is not publicly available, it is reasonable to assume that its MACs count is significantly higher than that of the other methods due to its larger parameter count. Fig. 6 shows sample predictions made by the GEC network on randomly selected images from the CrowdPose test set.

TABLE II. COMPARISON OF LEARNABLE KEYPOINT-GROUPING NETWORKS OF BOTTOM-UP POSE ESTIMATION METHODS

Method	# Parameters	# MACs
HGG [18]	0.300 M	4.6 G
CenterGroup [47]	1.700 M	-
GEC (ours)	**0.274** M	**3.3** G

Note: CenterGroup did not publish MACs data.

D. Discussion

Traditional keypoint-grouping methods, such as the Munkres algorithm used in [14, 45, 47] or greedy algorithms used to maximize path integrals such as in [11], function as non-learnable decoders for keypoint features in embedding space (e.g., associative embeddings [13]). These optimization algorithms group keypoints by minimizing an energy function (e.g., total distance between keypoint values in embedding space) without considering other factors, such as pose structure. As a result, they may struggle when keypoints from different individuals have similar embedded values due to proximity or occlusion. In contrast, the proposed Graph Edge Classifier—a supervised learning approach to keypoint-grouping— rectifies this limitation by acting as a learnable decoder for keypoint and edge embeddings. Furthermore, since the grouping step in now learnable, the errors in keypoint association are propagated backward to the backbone network, providing feedback on the output embedding maps and improving feature extraction from images. By learning to classify graph edges, the network implicitly learns the structure of human poses.

As shown in Table I, the inclusion of edge features in the MPNN significantly improves the Average Precision (AP) of the edge classifier network compared to not using them. AP scores also increased when using the extended or complete connection models, which was expected, as skip-connections bypass occluded keypoints. However, feature extraction when using the extended or complete connection models may slightly impact inference speed. Our network achieved an AP score of 46.1%, which is comparable to other state-of-the-art 2D bottom-up pose estimation methods. We attribute the slightly lower AP scores to the separate training of the backbone and the graph edge classifier, which was necessary due to computational limitations, particularly memory constraints. We plan to explore using MobileNet [46] as an alternative backbone network to reduce computational requirements and conduct end-to end training, and use DMoN pooling [49] to resolve issues with duplicate keypoints in subgraphs. Additionally, as shown in Table II, our model is lightweight, with only 0.274 million parameters, outperforming other learnable keypoint-grouping methods in terms of the number of parameters.

E. Materials and Devices

The method was implemented in Python, using pytorch and pytorch-geometric packages. Training was conducted on a PC with 10GB of VRAM.

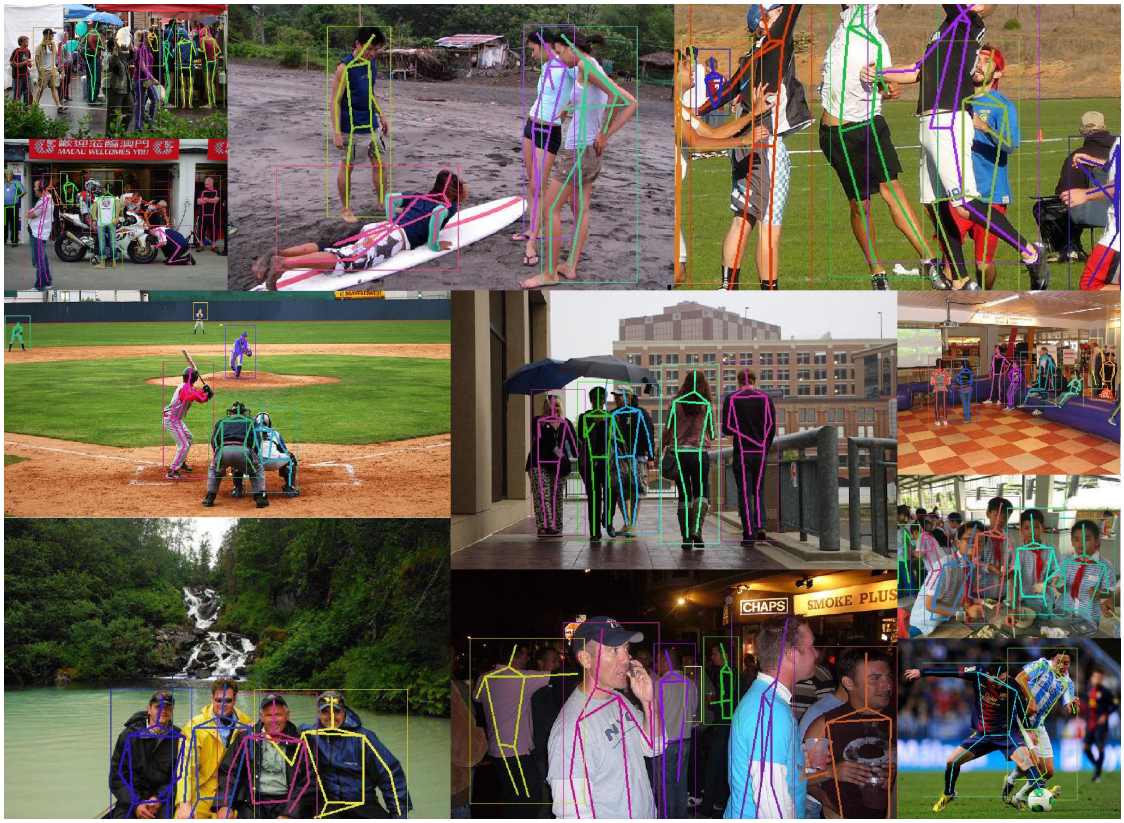

Fig. 6. Qualitative results of the GEC model on some images from the CrowdPose test set.

V. Conclusion

In this paper, we introduced a lightweight supervised learning method for grouping 2D keypoints in bottom-up human pose estimation, framing keypoint-grouping as a problem of graph edge classification. The proposed method leverages the message-passing neural networks framework, incorporating a novel node and edge-feature update functions to learn improved node and edge representations in a graph. The proposed graph edge classifier serves as a learnable decoder of keypoint and edge features in embedding space, in contrast to previous bottom-up pose estimation methods that relied on non-learnable optimization algorithms—such as dynamic programming—to group keypoints by minimizing some energy function. This learnable approach to keypoint-grouping also enhances the backbone network by providing feedback on the keypoint embedding map through the backpropagation of the grouping error to the backbone network.

Future work could explore conducting end-to-end training of the backbone network and the graph neural network using MobileNet alongside the proposed graph edge classifier network. Additionally, transformer architecture might be considered as a potential method for clustering keypoints.

Conflicts of Intrest

The authors declare no conflict of interests.

Autor Contributions

Marwan Maher conceptualized the research idea, wrote the manuscript, implemented the graph edge classification method, made necessary modifications to HigherHRNet model for its integration into the pipeline, and is responsible of all programming aspects of the study. Radwa Fathalla and Mohamed Shaheen have contributed to the theoretical foundations of the research. They also provided valuable support and advice during the experimental phase. All authors approved the final version.

References

[1] C. Xu, X. Yu, Z. Wang, and L. Ou, "Multi-view human pose estimation in human-robot interaction," in *Proc. IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020, pp. 4769–4775.

[2] A. Avogaro, F. Cunico, B. Rosenhahn, and F. Setti, "Markerless human pose estimation for biomedical applications: A survey," *Frontiers in Computer Science*, vol. 5, 2023.

[3] S. Park, J. Y. Chang, H. Jeong, J.-H. Lee, and J.-Y. Park, "Accurate and efficient 3d human pose estimation algorithm using single depth images for pose analysis in golf," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 105–113.

[4] R. Tous, "Pictonaut: Movie cartoonization using 3d human pose estimation and gans," *Multimedia Tools Appl.*, vol. 82, no. 14, pp. 21101–21115, Feb. 2023.

[5] P. G. S. do Couto Soares, A. Silva, and L. F. A. Pereira, "An assault detection system based on human pose tracking for video surveillance," in *Proc. Anais Estendidos da Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2019.

[6] M. Leung and Y.-H. Yang, "First sight: A human body outline labeling system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 359–377, 1995.

[7] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4733–4742.

[8] S. LI, Z.-Q. Liu, and A. B. Chan, "Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network," in *Proc. 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 488–495.

[9] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," in *Proc. 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1913–1921.

[10] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4929–4937.

[11] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," in *Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 11969–11978.

[12] X. Nie, J. Feng, J. Zhang, and S. Yan, "Single-stage multi-person pose machines," in *Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6950–6959.

[13] A. Newell, Z. Huang, and J. Deng, "Associative embedding: end to-end learning for joint detection and grouping," in *Proc. the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 2017, pp. 2274–2284.

[14] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation," in *Proc. CVPR*, 2020.

[15] Y. Yang, Z. Ren, H. Li, C. Zhou, X. Wang, and G. Hua, "Learning dynamics via graph neural networks for human pose estimation and tracking," in *Proc. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 8070–8080.

[16] H. Li, B. Shi, W. Dai, Y. Chen, B. Wang, Y. Sun, M. Guo, C. Li, J. Zou, and H. Xiong, "Hierarchical graph networks for 3d human pose estimation," arXiv preprint, arXiv:2111.11927, 2023.

[17] T. Swain, S. Sinha, A. Singh, K. Verma, and S. Das, "Human pose estimation using GNN," in *Proc. 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC),* 2022, pp. 1–5.

[18] S. Jin, W. Liu, E. Xie, W. Wang, C. Qian, W. Ouyang, and P. Luo, "Differentiable hierarchical graph grouping for multi-person pose estimation," in *Proc*. *European Conference on Computer Vision—ECCV 2020*, 2020, pp. 718–734.

[19] A. Zeng, X. Sun, L. Yang, N. Zhao, M. Liu, and Q. Xu, "Learning skeletal graph neural networks for hard 3d pose estimation," in *Proc. 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 11416–11425.

[20] N. Azizi, H. Possegger, E. Rodolà, and H. Bischof, "3d human pose estimation using möbius graph convolutional networks," in *Proc. 17th European Conference on Computer Vision—ECCV 2022,* Part I, 2022, pp. 160–178.

[21] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67–92, 1973.

[22] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, Jan. 2005.

[23] M. Dantone, J. Gall, C. Leistner, and L. Van Gool, "Human pose estimation using body parts dependent joint regressors," in *Proc. 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3041–3048.

[24] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proc. 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660.

[25] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3711–3719.

[26] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun, "Rethinking on multi-stage networks for human pose estimation," arXiv preprint, arXiv:1901.00148, 2019.

[27] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5686–5696.

[28] Y. Cai, Z. Wang, Z. Luo, B. Yin, A. Du, H. Wang, X. Zhou, E. Zhou, X. Zhang, and J. Sun, "Learning delicate local representations for multiperson pose estimation," in *Proc. European Conference on Computer Vision*, 2020.

[29] J. Wang, X. Long, Y. Gao, E. Ding, and S. Wen, "Graph-PCNN: Two stage human pose estimation with graph pose refinement," in *Proc. 16th European Conference on Computer Vision, ECCV 2020,* 2020, pp. 492–508.

[30] Y. Li, D. Yang, Y. Chen, C. Peng, Z. Sun, and L. Jiao, "A lightweight top-down multi-person pose estimation method based on symmetric transformation and global matching," *IEEE Access*, vol. 10, pp. 22,112–22,122, 2022.

[31] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi person 2d pose estimation using part affinity fields," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1302–1310.

[32] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," in *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, vol. 43, no. 10, pp. 3349–3364.

[33] H. Zhang, J. Xia, G. Zhang, and M. Xu, "Learning graph representations through learning and propagating edge features," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2022.

[34] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph*., vol. 38, no. 5, Oct. 2019.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[36] Z. Qiu, K. Qiu, J. Fu, and D. Fu, "DGCN: Dynamic graph convolutional network for efficient multi-person pose estimation," in *Proc. the AAAI Conference on Artificial Intelligence*, Apr. 2020, vol. 34, no. 07, pp. 11,924–11,931.

[37] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. 2005 IEEE International Joint Conference on Neural Networks*, 2005, vol. 2, pp. 729–734.

[38] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. International Conference on Learning Representations*, 2017.

[40] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. ICML'17*, 2017, pp. 1263–1272.

[41] L. R. Ford and D. R. Fulkerson, *Static Maximal Flow*, Princeton University Press, 1962, pp. 1–35.

[42] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, "Crowdpose: Efficient crowded scenes pose estimation and a new benchmark," in *Proc. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10,855–10,864.

[43] R. Karp, "An algorithm to solve the MXN assignment problem in expected time o (mn log n)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M78/67, Sep. 1978.

[44] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.

[45] Y. Wang, M. Li, H. Cai, W. Chen, and S. Han, "Lite pose: Efficient architecture design for 2d human pose estimation," in *Proc. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13,116–13,126.

[46] G. Braso, N. Kister, and L. Leal-Taixe, "The center of attention: Center-keypoint grouping via attention for multi-person pose estimation," in *Proc. 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 11,833–11,843.

[47] H. Wang, L. Zhou, Y. Chen, M. Tang, and J. Wang, "Regularizing vector embedding in bottom-up human pose estimation," in *Proc. European Conference on Computer Vision, ECCV 2022*, 2022, pp. 107–122.

[48] F. Tian and S. Kim, "Globally-robust instance identification and locally accurate keypoint alignment for multi-person pose estimation," in *Proc. the 31st ACM International Conference on Multimedia*, MM 23, 2023, pp. 4816–4827.
[49] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *J. Mach. Learn. Res*., vol. 24, no. 1, Mar. 2024.