Hybrid Graph Neural Networks for Crowd Counting

Ao Luo,[1] Fan Yang,[2] Xin Li,[2] Dong Nie,[3] Zhicheng Jiao,[4] Shangchen Zhou,[5] Hong Cheng[1]*
[1]Center for Robotics, University of Electronic Science and Technology of China
[2]Inception Institute of Artificial Intelligence
[3]Department of Computer Science, University of North Carolina at Chapel Hill
[4]University of Pennsylvania [5]Nanyang Technological University
aoluo_uestc@hotmail.com, hcheng@uestc.edu.cn

Abstract

Crowd counting is an important yet challenging task due to the large scale and density variation. Recent investigations have shown that distilling rich relations among multi-scale features and exploiting useful information from the auxiliary task, i.e., localization, are vital for this task. Nevertheless, how to comprehensively leverage these relations within a unified network architecture is still a challenging problem. In this paper, we present a novel network structure called Hybrid Graph Neural Network (HyGnn) which targets to relieve the problem by interweaving the multi-scale features for crowd density as well as its auxiliary task (localization) together and performing joint reasoning over a graph. Specifically, HyGnn integrates a hybrid graph to jointly represent the task-specific feature maps of different scales as nodes, and two types of relations as edges: **(i)** multi-scale relations capturing the feature dependencies across scales and **(ii)** mutual beneficial relations building bridges for the cooperation between counting and localization. Thus, through message passing, HyGnn can capture and distill richer relations between nodes to obtain more powerful representations, providing robust and accurate results. Our HyGnn performs significantly well on four challenging datasets: ShanghaiTech Part A, ShanghaiTech Part B, UCF_CC_50 and UCF_QNRF, outperforming the state-of-the-art algorithms by a large margin.

Introduction

Crowd counting, with the purpose of analyzing large crowds quickly, is a crucial yet challenging computer vision and AI task. It has drawn a lot of attention due to its potential applications in public security and planning, traffic control, crowd management, public space design, *etc.*

Same as many other computer vision tasks, the performance of crowd counting has been substantially improved by Convolutional Neural Networks (CNNs). Recently, the state-of-the-art crowd counting methods (Liu, Weng, and Mu 2019; Liu, Salzmann, and Fua 2019; Wan et al. 2019; Liu, Salzmann, and Fua 2019; Jiang et al. 2019) mostly follow the *density-based* paradigm. Given an image or video frame, CNN-based regressors are trained to estimate the crowd density map, whose values are summed to give the entire crowd count.

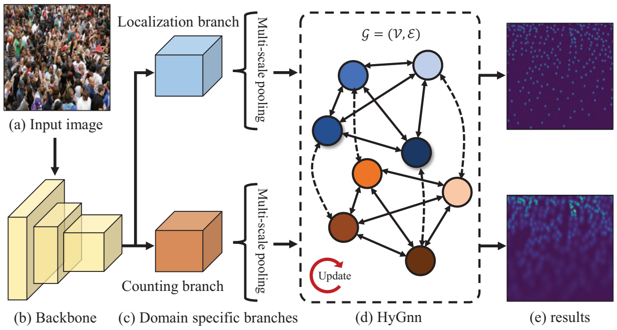

Figure 1: Illustration of the proposed HyGnn model. (a) Input image, in which crowds have heavy overlaps and occlusions. (b) Backbone, which is a truncated VGG-16 model. (c) Domain-specific branches: one for crowd counting and the other for localization. (d)HyGnn, which represents the features from different scales and domains as nodes, while the relations between them as edges. After several message passing iterations, multiple types of useful relations are built. (e) Crowd density map (for counting) and localization map (as the auxiliary task).

Recent studies (Shen et al. 2018; Cao et al. 2018; Li et al. 2017; 2018) have shown that multi-scale information, or *relations* across multiple scales helps to capture contextual knowledge which benefits crowd counting. Moreover, the crowd counting and its auxiliary task (localization), in spite of analyzing the crowd scene from different perspectives, could provide beneficial clues for each other (Liu, Weng, and Mu 2019; Lian et al. 2019). Crowd density map can offer guidance information and self-adaptive perception for precise crowd localization, and on the other hand, crowd localization can help to alleviate local inconsistency issue in density map. The mutual cooperation, or called *mutual beneficial relation*, is the key factor in estimating the high-quality density map. However, most methods only consider the crowd counting problem from one aspect, while ignore the other one. Consequently, they fail to fully utilize mul-

*denotes the Corresponding Author

tiple types of useful relations or structural dependencies in the learning and inferring processes, resulting in sub-optimal results.

One primary reason is the lack of a unified and effective framework capable of modeling the different types of relations (*i.e.,* multi-scale relations and mutual beneficial relations) over a single model. To address this issue, we introduce a novel Hybrid Graph Neural Network (HyGnn), which formulates the crowd counting and localization as a graph-based, joint reasoning procedure. As shown in Fig. 1, we build a hybrid graph which consists of two types of nodes, *i.e.*, counting nodes storing density-related features and localization nodes storing location-related features. Besides, there are two different pairwise relationships (edge types) between them. By interweaving the multi-scale and multi-task features together and progressively propagating information over the hybrid graph, HyGnn can fully leverage the different types of useful information, and is capable of distilling the valuable, high-order relations among them for much more comprehensive crowd analysis.

HyGnn is easy to implement and end-to-end learnable. Importantly, it has two major benefits in comparison to existing models for crowd counting (Liu, Weng, and Mu 2019; Liu, Salzmann, and Fua 2019; Wan et al. 2019; Liu, Salzmann, and Fua 2019; Jiang et al. 2019). **(i)** HyGnn interweaves crowd counting and localization with a joint, multi-scale and graph-based processing rather than a simple combination as done in most existing solutions. Thus, HyGnn significantly strengthens the information flow between tasks and across scales, thereby enabling the augmented representation to incorporate more useful priors learned from the auxiliary task and different scales. **(ii)** HyGnn explicitly models and reasons all relations (multi-scale relations and mutual beneficial relations) simultaneously over a hybrid graph, while most existing methods are not capable of dealing with such complicated relations. Therefore, our HyGnn can effectively capture their dependencies to overcome inherent ambiguities in the crowd scenes. Consequently, our predicted crowd density map is potentially more accurate, and consistent with the true crowd localization.

In our experiments, we show that HyGnn performs remarkably well on four well-used benchmarks and surpasses prior methods by a large margin. Our **contributions** are summarized in three aspects:

- We present a novel end-to-end learnable model, namely Hybrid Graph Neural Network (HyGnn), for joint crowd counting and localization. To the best of our knowledge, HyGnn is the first deep model capable of explicitly modeling and mining high-level relations between counting and its auxiliary task (localization) across different scales through a hybrid graph model.
- HyGnn is equipped with a unique multi-tasking property, where different types of nodes, connections (or edges), and message passing functions are parameterized by different neural designs. With such property, HyGnn can more precisely leverage cooperative information between crowd counting and localization to boost the counting performance.
- We conduct extensive experiments on four well-known benchmarks including ShanghaiTech Part A, ShanghaiTech Part B, UCF_CC_50 and UCF_QNRF, on which we set new records.

Related Works

Crowd Counting and Localization. Early works (Viola, Jones, and Snow 2005) in crowd counting use *detection-based* methods, and employ handcrafted features like Haar (Viola, Jones, and others 2001) and HOG (Dalal and Triggs 2005) to train the detector. The overall performances of these algorithms are rather limited due to various occlusion. The *regression-based* methods, which can avoid solving the hard detection problem, has become mainstream and achieved great performance breakthroughs. Traditionally, the regression models (Chen et al. 2013; Lempitsky and Zisserman 2010; Pham et al. 2015) learn the mapping between low-level images features and object count or density, using gaussian process or random forests regressors. Recently, various CNN-based counting methods have been proposed (Zhang et al. 2015; 2016; Liu, Weng, and Mu 2019; Liu, Salzmann, and Fua 2019; Wan et al. 2019; Liu, Salzmann, and Fua 2019; Jiang et al. 2019) to better deal with different challenges, by predicting a density map whose values are summed to give the count. Particularly, the scale variation issue has attracted the most attention of recent CNN-based methods (Dai et al. 2019; Varior et al. 2019). On the other hand, as observed by some recent researches (Idrees et al. 2018a; Liu, Weng, and Mu 2019; Lian et al. 2019), although the current state-of-the-art methods can report accurate crowd count, they may produce the density map which is inconsistent with the true density. One major reason is the lack of crowd localization information. Some recent studies (Zhao et al. 2019; Liu, Weng, and Mu 2019) have tried to exploit the useful information from localization in a unified framework. They, however, only simply share the underlying representations or interweave two modules for different task together for more robust representations. Differently, our HyGnn considers a better way to utilize the mutual guidance information: explicitly modeling and iteratively distilling the mutual beneficial relations across scales within a hybrid graph. For a more comprehensive survey, we refer interested readers to (Kang, Ma, and Chan 2018).

Graph Neural Networks. The essential idea of Graph Neural Network (GNN) is to enhance the node representations by propagating information between nodes. Scarselli *et al.* (Scarselli et al. 2008) first introduced the concept of GNN, which extended recursive neural networks for processing graph structure data. Li *et al.* (Li et al. 2016) proposed to improve the representation capacity of GNN by using Gated Recurrent Units (GRUs). Gilmer *et al.* (Gilmer et al. 2017) used message passing neural network to generalize the GNN. Recently, GNN has been successfully applied in attributes recognition (Meng et al. 2018), human-object interactions (Qi et al. 2018a), action recognition (Si et al. 2018), *etc.* Our HyGnn shares similar ideas with

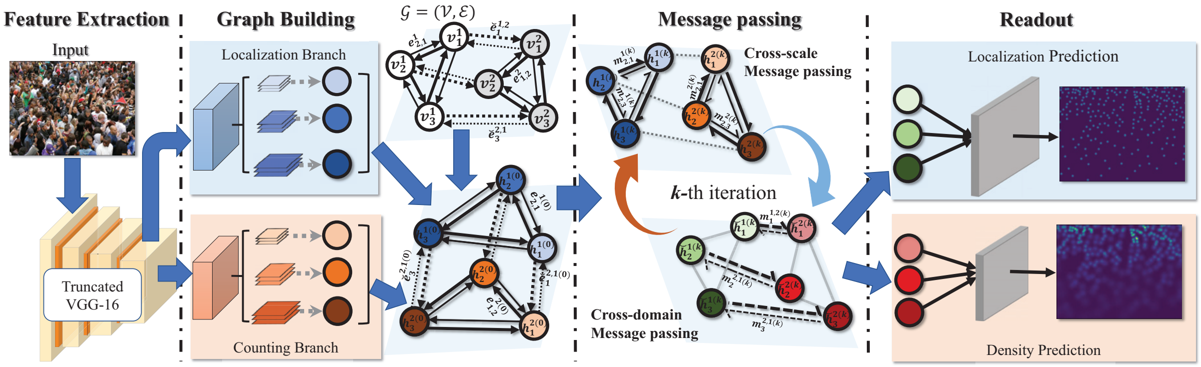

Figure 2: Overall of our HyGnn model. Our model is built on the truncated VGG-16, and includes a Domain-specific Feature Learning Module to extract features from different domain. A novel HyGnn is used to distill multi-scale and cross-domain information, so as to learn better representations. Finally, the multi-scale features are fused to produce the density map for counting as well as generate the auxiliary task prediction (localization map).

above methods that fully exploits the underlying relationships between multiple latent representations through GNN. However, most existing GNN-based models are designed to deal with *only one* relation type, which may limit the power of GNN. To overcome above limitation, our HyGnn is equipped with a *multitasking* property, *i.e.,* parameterizing different types of connections (or edges) and the message passing functions with different neural designs, which significantly discriminates HyGnn from all existing GNNs.

Methodology

Preliminaries

Problem Formulation. Let the crowd counting model be represented by the function $\mathcal{M}$ which takes an Image $\mathcal{I}$ as input and generates the corresponding crowd density map $\mathcal{D}$ (for counting) as well as the auxiliary task prediction, *i.e.*, localization map $\mathcal{L}$. Let $\mathcal{D}^g$ and $\mathcal{L}^g$ be the groundtruth density map and localization map, respectively. Our goal is to learn powerful *domain-specific* representations, denoted as $\mathbf{f}_d$ and $\mathbf{f}_l$, to minimize errors between the estimated $\mathcal{D}$ and groundtruth $\mathcal{D}^g$, as well as between $\mathcal{L}$ and $\mathcal{L}^g$. It should be noted that crowd counting and localization tasks share a common *meta-objective*, and $\mathcal{D}^g$ and $\mathcal{L}^g$ are obtained from the same point-annotations without additional target labels.

Notations. To achieve above goal, we need to capture and distill the underlying dependencies between multi-task and multi-scale features. Given the multi-scale deep density feature maps $\mathcal{F}_d = \{\mathbf{f}_d^{s_i}\}_{i=1}^N$ and multi-scale deep localization feature maps $\mathcal{F}_l = \{\mathbf{f}_l^{s_i}\}_{i=1}^N$, we represent $\mathcal{F}_d$ and $\mathcal{F}_l$ with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite set of nodes and $\mathcal{E}$ is a finite set of edges. The nodes in our HyGnn are further grouped into two types: $\mathcal{V} = \mathcal{V}^1 \bigcup \mathcal{V}^2$, where $\mathcal{V}^1 = \{v_i^1\}_{i=1}^{|\mathcal{V}^1|}$ is the set of counting (density) nodes and $\mathcal{V}^2 = \{v_i^2\}_{i=1}^{|\mathcal{V}^2|}$ denotes the set of localization nodes. In our model, we have the same number of nodes in two latent domains, and therefore $|\mathcal{V}^1| = |\mathcal{V}^2| = N$. Accordingly, there are two types of edges $\mathcal{E} = \mathcal{E}_{i,j} \bigcup \breve{\mathcal{E}}_{m,n}$ between them: **(i)** cross-scale edge $e_{i,j}^m = (v_i^m, v_j^m) \in \mathcal{E}_{i,j}$ stands for the multi-scale relation between nodes from the i_{th} scale to the j_{th} scale within the same domain $m \in \{1, 2\}$, where $i, j \in \{1, \cdots, N\}$; **(ii)** cross-domain edge $\breve{e}_i^{m,n} = (v_i^m, v_i^n) \in \breve{\mathcal{E}}_{m,n}$ reflects mutual beneficial relations between nodes from the domain m to domain n with the same scale $i \in \{1, \cdots, N\}$, where $m, n \in \{1, 2\}$ & $m \neq n$. For each node v_i^m ($i \in \{1, \cdots, N\}$ & $m \in \{1, 2\}$), we learn its updated representation, namely $\mathbf{h}_i^m$, through aggregating representations of its neighbors. Finally, the updated multi-scale features $\mathcal{H}_1 = \{\mathbf{h}_i^1\}_{i=1}^{|\mathcal{V}^1|}$ and $\mathcal{H}_2 = \{\mathbf{h}_i^2\}_{i=1}^{|\mathcal{V}^2|}$ are fused to produce the final representation $\mathbf{f}_d$ and $\mathbf{f}_l$, which is used to generate the outputs $\mathcal{D}$ and $\mathcal{L}$. Here, we only consider multi-scale relations between nodes of the same domain, and mutual beneficial relations between nodes of the same scale in our graph model. Considering that our graph model is designed to deal with two different nodes- and relations-types, we call it as Hybrid Graph Neural Network (HyGnn) which will be detailed in the following section.

Hybrid Graph Neural Network (HyGnn)

Overview. The key idea of our HyGnn is to perform K message propagation iterations over $\mathcal{G}$ to joint distill and reason all relations between crowd counting and auxiliary task (localization) across scales. Generally, as shown in Fig. 2, HyGnn map the given image $\mathcal{I}$ to the final predictions $\mathcal{D}$ and $\mathcal{L}$ through three phases. **First**, in the domain-specific feature extracting phase, HyGnn generates multi-scale deep density features $\mathcal{F}_d$ and localization features $\mathcal{F}_l$ for $\mathcal{I}$ through a Domain-specific Feature Learning Module (DFL), and represents these features as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. **Second**, a parametric message passing phase runs for K times to propagate message between nodes and also to update node representations according to the received messages within the graph $\mathcal{G}$. **Third**, a readout phase fuses the updated multi-scale features $\mathcal{H}_1$ and $\mathcal{H}_2$ to generate final representations (*i.e.*, $\mathbf{f}_d$ and $\mathbf{f}_l$), and map them to the outputs $\mathcal{D}$ and $\mathcal{L}$. Note that, as crowd counting is our main task, we emphasize the accuracy of $\mathcal{D}$ during the learning process.

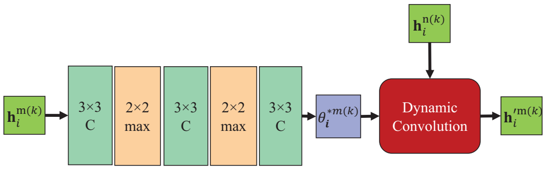

Figure 3: The architecture of the learnable adapter. The adapter takes the node representation of one (source) domain $\mathbf{h}_i^{m(k)}$ as input and outputs the adaptive convolution parameters $\theta_i^{*m(k)}$. The adaptive representation $\mathbf{h}'^{m(k)}_i$ is generated conditioned on $\mathbf{h}_i^{n(k)}$.

Domain-specific Feature Learning Module (DFL). The Domain-specific Feature Learning Module (DFL) is one of the major modules of our model, which extracts multi-scale, domain-specific features $\mathcal{F}_d = \{\mathbf{f}_d^{s_i}\}_{i=1}^N$ and $\mathcal{F}_l = \{\mathbf{f}_d^{s_i}\}_{i=1}^N$ from the input $\mathcal{I}$. DFL is composed of three major parts: one front-end and two domain-specific back-ends.

The front-end Fr is based on the well-known VGG-16, which maps the RGB images $\mathcal{I}$ to the shared underlying representations: $\mathbf{f}_{share} = Fr(\mathcal{I})$. More specifically, the first 10 layers from VGG-16 are deployed as the front-end and shared by these two tasks. Meanwhile, different dilated convolution layers are used as the back-ends, denoted as B_d and B_l, to enlarge receptive fields which are tailored for learning domain-specific features: $\mathbf{f}_d = B_d(\mathbf{f}_{share})$ and $\mathbf{f}_l = B_l(\mathbf{f}_{share})$. In addition, the Pyramid Pooling Module (PPM) (Zhao et al. 2017) is applied in each domain-specific back-end for extracting multi-scale features, followed by an interpolation layer R to ensure multi-scale feature maps to have the same size $H \times W$. Details of the DFL architecture can be found in the supplementary file.

Node Embedding. In our HyGnn, each node v_i^1 or $v_i^2 \in \mathcal{V}$ takes a unique value from $\{1, \cdots, |\mathcal{V}|\}$, and is associated with an initial node embedding (or node state), namely $\mathbf{v}_i^1$ or $\mathbf{v}_i^2$. We utilize our DFL to extract multi-scale domain-specific features as initial node representations. Take an arbitrary counting node $v_i^1 \in \mathcal{V}^1$ for example, its initial representations $\mathbf{h}_i^{1(0)}$ can be computed as:

$$\mathbf{h}_i^{1(0)} = \mathbf{v}_i^1 = R(P(\mathbf{f}_d, s_i)) \in \mathbb{R}^{H \times W \times C}, \tag{1}$$

where $\mathbf{h}_i^{1(0)} \in \mathbb{R}^{H \times W \times C}$ is 3D tensor features. $R(\cdot)$ and $P(\cdot)$ are the interpolation operation and pyramid pooling operation, respectively. The initial node representation for the localization node $v_i^2 \in \mathcal{V}^2$ is defined similarly as follows:

$$\mathbf{h}_i^{2(0)} = \mathbf{v}_i^2 = R(P(\mathbf{f}_l, s_i)) \in \mathbb{R}^{H \times W \times C}, \tag{2}$$

where $\mathbf{h}_i^{2(0)} \in \mathbb{R}^{H \times W \times C}$ denotes the initial node representation for the localization node $v_i^2 \in \mathcal{V}^2$.

Cross-scale Edge Embedding. A cross-scale edge $e_{i,j}^m \in \mathcal{E}_{i,j}$ connects two nodes v_i^m and v_j^m which are from the same domain $m \in \{1, 2\}$ but different scales $i, j \in \{1, \cdots, N\}$. The cross-scale edge embedding, denoted as $\mathbf{e}_{i,j}^m$, is used to distill the multi-scale relation on the two sides of the edge from v_i^m to v_j^m as edge's representation. To this goal, we employ a relation function $f_{rel}(\cdot, \cdot)$ to capture the multi-scale relations,

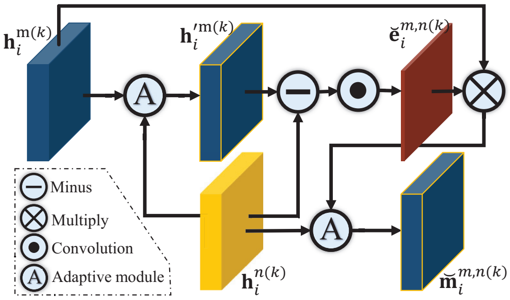

Figure 4: Detailed illustration of the cross-domain edge embedding and message aggregation. Please see text for details.

$$\mathbf{e}_{i,j}^{m(k)} = f_{rel}(\mathbf{h}_i^{m(k)}, \mathbf{h}_j^{m(k)}) = Conv(\mathrm{g}(\mathbf{h}_i^{m(k)}, \mathbf{h}_j^{m(k)})) \in \mathbb{R}^{H \times W \times C}, \tag{3}$$

where $\mathrm{g}(\cdot, \cdot)$ is a function to combine features $\mathbf{h}_i^{m(k)}$ and $\mathbf{h}_j^{m(k)}$. Following (Wang et al. 2019b), we model $\mathrm{g}(\mathbf{h}_i, \mathbf{h}_j) = \mathbf{h}_i - \mathbf{h}_j$, making the relations calculated based on the difference between the node embedding to alleviate the *symmetric* impact in feature combination. $Conv(\cdot)$ means the convolution operation that is used to learn the edge embedding in a data-driven way. Each element in $\mathbf{e}_{i,j}^{m(k)}$ reflects the pixel-level relations between the nodes of different scales from scale i to scale j. As a result, $\mathbf{e}_{i,j}^{m(k)}$ can be considered as the features that depict the *multi-scale relationships* between nodes.

Cross-domain Edge Embedding. Because our HyGnn is designed to fully exploit complementary knowledge contained in the nodes of different domains ($m, n \in \{1, 2\}$ & $m \neq n$), one major challenging is to overcome the "domain gap" between them. Rather than directly combining features as that used in cross-scale edge embedding, we first adapt the node representation of one (source) domain $\mathbf{h}_i^{m(k)}$ conditioned on the node representation of the other (target) domain $\mathbf{h}_i^{n(k)}$ to overcome the domain difference. Here, inspired by (Bertinetto et al. 2016), we integrate a learnable adapter $\mathcal{A}_m(\mathbf{h}_i^{m(k)} \| \mathbf{h}_i^{n(k)})$ into our HyGnn to transform the original node representation $\mathbf{h}_i^{m(k)}$ to the adaptive representation $\mathbf{h}'^{m(k)}_i$ as follows:

$$\begin{aligned} \mathbf{h}'^{m(k)}_i &= \mathcal{A}_m(\mathbf{h}_i^{m(k)} \| \mathbf{h}_i^{n(k)}) = \theta_i^{*m(k)} * \mathbf{h}_i^{n(k)}, \\ where\ \theta_i^{*m(k)} &= E_\phi(\mathbf{h}_i^{m(k)}). \end{aligned} \tag{4}$$

In the above function, $*$ is the convolution operation, and $\theta_i^{*m(k)}$ means the dynamic convolutional kernels. $E_\phi(\cdot)$ is a one-shot learner to predict the dynamic parameters $\theta_i^{*m(k)}$ from a single exemplar. Following (Nie et al. 2018), as shown in Fig. 3, we implement it by a small CNN with learnable parameters ϕ.

After achieving the adaptive representation $\mathbf{h}'^{m(k)}_i$, the cross-domain edge embedding $\breve{\mathbf{e}}_i^{m,n}$ for the edge $\breve{e}_i^{m,n} = (v_i^m, v_i^n) \in \mathcal{E}_{m,n}$ can be computed as:

$$\breve{\mathbf{e}}_i^{m,n(k)} = f_{rel}(\mathbf{h}'^{m(k)}_i, \mathbf{h}_i^{n(k)}) = Conv(\mathrm{g}(\mathbf{h}'^{m(k)}_i, \mathbf{h}_i^{n(k)})) \in \mathbb{R}^{H\times W\times C}, \tag{5}$$

where $\breve{\mathbf{e}}_i^{m,n} \in \mathbb{R}^{H\times W\times C}$ is a 3D tensor, which represents the hidden representation of a cross-domain relation. The detailed architecture can be found in Fig. 4.

Cross-scale Message Aggregation. In our HyGnn, we use different aggregation schemes for a node to aggregate feature messages from its neighbors. For the message $\mathbf{m}_{i,j}^m$ passed from node v_i^m to v_j^m within the same domain across different scales, we have:

$$\mathbf{m}_{i,j}^{m(k)} = M(\mathbf{h}_i^{m(k-1)}, \mathbf{e}_{i,j}^{m(k-1)}) = softmax(\mathbf{e}_{i,j}^{m(k-1)}) \cdot \mathbf{h}_i^{m(k-1)}, \tag{6}$$

where $M(\cdot)$ is the cross-scale message passing function (aggregator), and $softmax(\cdot)$ maps the edge's embedding into the link's weight. Note that because our HyGnn is designed to handle a pixel-level task, the link's weight between nodes is in the manner of a 2D map. Thus, $\mathbf{m}_{i,j}^m$ assigns the pixel-wise weighted features from node v_i^m to v_j^m to aggregate information.

Cross-domain Message Aggregation. Because the cross-domain discrepancy is significant in high-dimensional feature space and distribution, directly passing the learned representations of one node to its neighboring nodes for aggregation is a sub-optimal solution, which might damage the learned representations. Therefore, we formulate the message passing from node v_i^m to v_i^n as an adaptive representation learning process, conditioned on $\mathbf{h}_i^n$. Here, we use the similar idea with that used in cross-domain edge embedding process, *i.e.,* using a one-shot adapter to predict the message that should be passed:

$$\begin{aligned}\breve{\mathbf{m}}_i^{m,n(k)} &= \breve{M}(\mathbf{h}_i^{m(k-1)}, \breve{\mathbf{e}}_i^{m,n(k-1)} \| \mathbf{h}_i^{n(k-1)}) \\ &= \breve{\mathcal{A}}_m(softmax(\breve{\mathbf{e}}_i^{m,n(k-1)}) \cdot \mathbf{h}_i^{m(k-1)} \| \mathbf{h}_i^{n(k-1)}) \\ &= \breve{E}_\eta(softmax(\breve{\mathbf{e}}_i^{m,n(k-1)}) \cdot \mathbf{h}_i^{m(k-1)}) * \mathbf{h}_i^{n(k-1)} \\ &= \psi^{*m(k-1)}_i * \mathbf{h}_i^{n(k-1)},\end{aligned} \tag{7}$$

where $\breve{M}(\cdot)$ means the message passing function between nodes of two different domains. $\breve{\mathcal{A}}(\cdot)$ is the adapter which is conditioned on the node embedding of target domain $\mathbf{h}_i^{n(k-1)}$. $\breve{E}_\eta(\cdot)$ means a small CNN with learnable parameters η, which serves as one-shot learner to predict the dynamic parameters. $\psi^{*m(k-1)}_i$ is the predicted dynamic convolutional kernels, which includes guidance information that should be propagated from node v_i^m to node v_i^n.

Two-stage Node-state Update. In the k_{th} step, our HyGnn first aggregates information from the neighbor nodes of the other domain within the same scale i using Eq. 7. Therefore, v_i^n ($i \in \{1, \cdots, N\}$ & $n \in \{1, 2\}$) gets an intermediate state $\widetilde{\mathbf{h}}_i^{n(k)}$ by taking into account its received cross-domain message $\breve{\mathbf{m}}_i^{m,n(k)}$ and its prior state $\mathbf{h}_i^{n(k-1)}$. Here, following (Qi et al. 2018b), we apply Gated Recurrent Unit (GRU) (Ballas et al. 2015) as the update function,

$$\widetilde{\mathbf{h}}_i^{n(k)} = U_{GRU}(\mathbf{h}_i^{n(k-1)}, \breve{\mathbf{m}}_i^{m,n(k)}). \tag{8}$$

Then, HyGnn performs message passing across scales within the same domain n using Eq. 3 and aggregates message using Eq. 6. After that, v_i^n gets the new state $\mathbf{h}_i^{n(k)}$ of the k_{th} iteration by considering the cross-scale message $\mathbf{m}_{j,i}^{n(k)}$ and its intermediate state $\widetilde{\mathbf{h}}_i^{n(k)}$,

$$\mathbf{h}_i^{n(k)} = U_{GRU}(\widetilde{\mathbf{h}}_i^{n(k)}, \mathbf{m}_{j,i}^{n(k)}). \tag{9}$$

Readout Function. After K message passing iterations, the updated multi-scale features of two domains (node representations) $\mathcal{H}_1 = \{\mathbf{h}_i^1\}_{i=1}^{|\mathcal{V}^1|}$ and $\mathcal{H}_2 = \{\mathbf{h}_i^2\}_{i=1}^{|\mathcal{V}^2|}$ are firstly merged to form their final representations $\mathbf{f}_d$ and $\mathbf{f}_l$,

$$\mathbf{f}_d = C_d(\mathcal{H}_1) \ \ and \ \ \mathbf{f}_l = C_l(\mathcal{H}_2), \tag{10}$$

where $C_d(\cdot)$ and $C_l(\cdot)$ are the merge functions which are implemented with concatenation layers. Then, $\mathbf{f}_d$ and $\mathbf{f}_l$ are fed into a convolution layer to get the final per-pixel predictions,

Loss. Our HyGnn is implemented to be fully differentiable and end-to-end trainable. The loss for each task can be computed for the outputs of readout functions, and the error can propagate back according to chain rule. Here, we simply employ the Mean Square Error (MSE) loss to optimize network parameters for these two tasks:

$$L = L_1(\mathcal{D}^g, \mathcal{D}) + \lambda L_2(\mathcal{L}^g, \mathcal{L}), \tag{11}$$

where L_1 and L_2 are MSE losses, and λ is the combination loss. Because our main task is crowd counting, we set $\lambda = 0.001$ to emphasize the accuracy of crowd counting.

Experiments

In this section, we experimentally validate our HyGnn on four public counting benchmarks (*i.e.,* ShanghaiTech Part A, ShanghaiTech Part B, UCF_CC_50 and UCF_QNRF). First, we conduct an ablation experiment to prove the effectiveness of our hybrid graph model and the multi-task learning. Then, our proposed HyGnn is evaluated on all of these public benchmarks, while comparing the performance against other state-of-the-art approaches.

Datasets. We use Shanghai Tech (Zhang et al. 2016), UCF_CC_50 (Idrees et al. 2013) and UCF_QNRF (Idrees et al. 2018b) for benchmarking our HyGnn. Shanghai Tech provides 1,198 annotated images with more than 330K people with head center annotations. It includes two subsets: Shanghai Tech A and Shanghai Tech B. UCF_CC_50 provides 50 images with 63,974 head annotations in total. The small dataset size and large count variance make it a very challenging dataset. UCF_QNRF is the largest dataset to date, which contains 1,535 images which are divided into train and test sets of 1,201 and 3,34 images respectively. All of these benchmarks have been widely used for performance evaluation by existing approaches.

Implementation Details and Evaluation Protocol. To make the comparison fair, we use a truncated VGG as the backbone network. Specifically, the first 10 convolutional layers from VGG-16 are used as the front-end and shared by both two tasks. Following (Li, Zhang, and Chen 2018),

Table 1: Comparison with other state-of-the-art crowd counting methods on four benchmark crowd counting datasets using the MAE and MSE metrics.

Methods	Shanghai Tech A		Shanghai Tech B		UCF_CC_50		UCF_QNRF	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Crowd CNN (Zhang et al. 2015)	181.8	277.7	32	49.8	467	498	-	-
MC-CNN (Zhang et al. 2016)	110.2	173.2	26.4	41.3	377.6	509.1	277	426
Switching CNN (Sam, Surya, and Babu 2017)	90.4	135	21.6	33.4	318.1	439.2	228	445
CP-CNN (Sindagi and Patel 2017)	73.6	106.4	20.1	30.1	298.8	320.9	-	-
D-ConvNet (Shi et al. 2018)	73.5	112.3	18.7	26	288.4	404.7	-	-
L2R (Liu, van de Weijer, and Bagdanov 2018)	72	106.6	13.7	21.4	279.6	388.9	-	-
CSRNet (Li, Zhang, and Chen 2018)	68.2	115	10.6	16	266.1	397.5	-	-
PACNN (Shi et al. 2019)	66.3	106.4	8.9	13.5	267.9	357.8	-	-
RA2-Net (Liu, Weng, and Mu 2019)	65.1	106.7	8.4	14.1	-	-	116	195
SFCN (Wang et al. 2019a)	64.8	107.5	7.6	13	214.2	318.2	124.7	203.5
TEDNet (Jiang et al. 2019)	64.2	109.1	8.2	12.8	249.4	354.2	113	188
ADCrowdNet (Liu et al. 2019)	63.2	98.9	7.6	13.9	257.1	363.5	-	-
HyGnn (Ours)	**60.2**	**94.5**	**7.5**	**12.7**	**184.4**	**270.1**	**100.8**	**185.3**

Table 2: Analysis of the proposed method. Our results are obtained on Shanghai Tech A.

Methods	MAE↓	MSE↓
Baseline Model (a truncated VGG)	68.2	115.0
Baseline + PSP (Zhao et al. 2017)	65.3	106.8
Baseline + Bidirectional Fusion (Yang et al. 2018)	65.1	105.9
Single-task GNN	62.5	103.4
Multi-task GNN *w/o* adapter	62.4	101.8
HyGnn (N=2, K=3)	62.1	100.8
HyGnn (N=3, K=3)	60.2	94.5
HyGnn (N=5, K=3)	60.2	94.1
HyGnn (N=3, K=1)	65.4	109.2
HyGnn (N=3, K=3)	60.2	94.5
HyGnn (N=3, K=5)	60.1	94.4
HyGnn (full model)	**60.2**	**94.5**

our counting and localization back-ends are composed of 8 dilated convolutions with kernel of size 3×3.

We use Adam optimizer with an initial learning rate 10^{-4}. We set the momentum to 0.9, the weight decay to 10^{-4} and the batchsize to 8. For data augmentation, the training images and the corresponding groundtruths are randomly flipped and cropped from different locations to the size of 400×400. In the test phase, we simply feed the whole image to our HyGnn to get the counting and localization results.

For evaluation, we adopt Mean Absolute Error (MAE) and Mean Squared Error (MSE) to evaluate the performance. They are defined as follows:

$$\text{MAE} = \frac{1}{\text{N}}\sum_{\text{i}=1}^{\text{N}}|\text{C}_\text{i} - \text{C}_\text{i}^{\text{GT}}| \text{ and } \text{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}|^2}, \qquad (12)$$

where C_i and C_i^{GT} are the estimated count and the ground truth count of the i_{th} test image.

Ablation Study. Extensive ablation experiments are performed on ShanghaiTech A to verify impact of each component in our HyGnn. Results are summarized in Tab. 2.

Effectiveness of HyGnn. To show the importance of our HyGnn, we offer a baseline model without HyGnn, which gives the results from our backbone model, a truncated VGG with dilated back-ends. As shown in Tab. 2, our HyGnn significantly outperforms the baseline by 8.0 in MAE (68.2 ↦ 60.2) and 20.5 in MSE (115.0 ↦ 94.5). This is because our HyGnn can simultaneously modeling the multi-scale and cross-domain relationship which is important for achieving accurate crowd counting results.

Multi-task GNN vs. Single-task GNN. To evaluate the advantages of multi-task cooperation, we offer a single-task model which only formulate cross-scale relationship. According to our experiments, our HyGnn outperforms the single-task graph neural network by 2.3 in MAE (62.5 ↦ 60.2) and 8.9 in MSE (103.4 ↦ 94.5). This is because our HyGnn is able to distill mutual benefits between density and localization, while single-task graph neural network ignores this important information.

Effectiveness of the Cross-domain Edge Embedding. Our HyGnn carefully deals with the cross-domain information by a learnable adapter. To evaluate its effectiveness, we provide a multi-task GNN without the learnable adapter, that is we directly fuse features of different domains through the aggregation operation. As shown Tab. 2, our cross-domain edge embedding method achieves better performance in both MAE (60.2 vs. 62.4) and MSE (94.5 vs. 101.8), which indicates that our design of cross-domain edge embedding method is helpful for better leveraging information from other domain.

Node Numbers N in HyGnn. In our model, we have N numbers of nodes in each domain, *i.e.*, $|\mathcal{V}^1| = |\mathcal{V}^2| = N$. To investigate the impact of node numbers, we report the performance of our HyGnn with different N. We find that with more scales in the model (2 ↦ 3), the performance improves significantly (*i.e.,* 62.1 ↦ 60.2 according to MAE and 100.8 ↦ 94.5 according to MSE). However, when further considering more scales (3 ↦ 5), our model only achieves slight performance improvements, *i.e.,* 60.2 ↦ 60.2 according to MAE and 94.5 ↦ 94.1 according to MSE. This may be due to the redundant information in multi-scale features. Considering the tradeoff between efficiency and performance, we set $N = 3$ in the following experiments.

Message Passing Iterations K. To evaluate the impact of message passing iterations K, we report the performance of

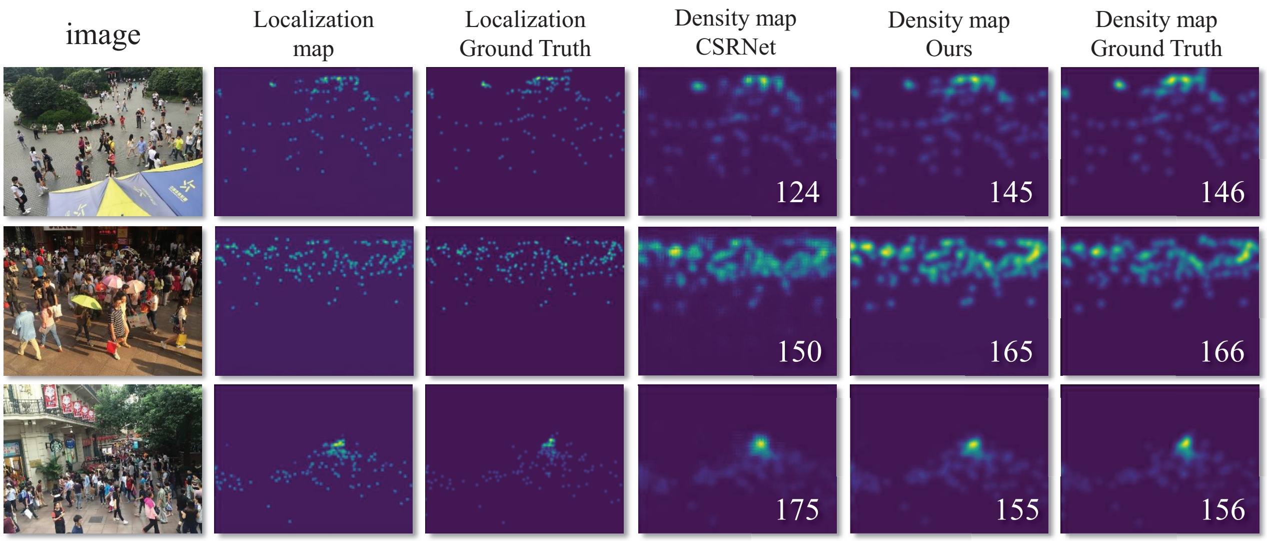

Figure 5: Density and localization maps generated by our HyGnn. We also show the counting map estimated by CSRNet for comparison. Clearly, our HyGnn produces more accurate results.

our model with different passing iterations K. Each message passing iteration in our HyGnn includes two cascade steps: i) the cross-scale message passing and ii) the cross-domain message passing. We find that with more iterations in the model (1 ↦ 3), the performance of our model improves. When further considering more iterations (3 ↦ 5), the performance improves slightly. Therefore, we find that our HyGnn converges to an optimal solution after three iterations.

GNN vs. Other Multi-feature Aggregation Methods. Here, we conduct an ablation to evaluate the superiority of GNN. We use a single-task GNN to fully exploit the underlying relationships between multi-scale features and compare our method with two well-known multi-scale feature aggregation methods (PSP (Zhao et al. 2017) and Bidirectional Fusion (Yang et al. 2018)) on Shanghai Tech A. As can be seen, our GNN-based method greatly outperforms other methods by a large margin.

Comparison with State-of-the-art. We compare our HyGnn with the state-of-the-art for the crowd counting.

Quantitative Results. As can be seen in Tab. 1, our HyGnn consistently achieves better results than other methods on four widely-used benchmarks. Specifically, our method greatly outperform previous best result by 3.0 in MAE and 4.4 in MSE on ShanghaiTech Part A. Although previous methods have worked well on ShanghaiTech Part B, our HyGnn also achieves the best performance. Compared with existing best algorithms like ADCrowdNet (Liu et al. 2019) and SFCN (Wang et al. 2019a), our HyGnn achieves performance gain by 0.1 in MAE and 1.2 in MSE and 0.1 in MAE and 0.3 in MSE respectively. On the most challenging UCF_CC_50, our HyGnn achieves considerable performance gain by decreasing the MAE from previous best 214.2 to 184.4 and MSE from 318.2 to 270.1. On UCF-QNRF dataset, our HyGnn also outperforms other methods with a large margin. As shown in Tab. 1, our HyGnn achieves a significant improvement of 10.8% in MAE over the existing best result produced by TEDNet (Jiang et al. 2019). Compared with other top-ranked methods, our HyGnn produce more accurate results. This is because HyGnn is able to leverage *free-of-cost* localization information and joint reason all relations between them.

Qualitative Results. Fig. 5 visualizes and compares the predicted density maps and counts of our HyGnn with CSRNet (Li, Zhang, and Chen 2018). In addition, we also show the localization results. We observe that our HyGnn is very powerful, and achieves much more accurate count estimations and reserve more consistency with the real crowd distributions. This is because our HyGnn can distill important information from the auxiliary task through a graph.

Conclusions

In this paper, we propose a novel method for crowd counting with a hybrid graph model. To best of our knowledge, it is the first deep model that can handle both multi-scale and mutual beneficial relations within a unified graph for crowd counting. The whole HyGnn is end-to-end differentiable and able to handle different relations effectively. Meanwhile, the domain gap between different tasks is also carefully considered in our HyGnn. According to our experiments, HyGnn achieves significant improvements compared to recent state-of-the-art methods on four benchmarks. We believe that our HyGnn can also incorporate other knowledge, *e.g.,* foreground information, for further performance improvements.

Acknowledgement. This work was supported in part by the National Key R&D Program of China (No.2017YFB1302300) and the NSFC (No.U1613223).

References

Ballas, N.; Yao, L.; Pal, C.; and Courville, A. 2015. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*.

Bertinetto, L.; Henriques, J. F.; Valmadre, J.; Torr, P.; and Vedaldi, A. 2016. Learning feed-forward one-shot learners. In *NeurIPS*.

Cao, X.; Wang, Z.; Zhao, Y.; and Su, F. 2018. Scale aggregation network for accurate and efficient crowd counting. In *ECCV*.

Chen, K.; Gong, S.; Xiang, T.; and Change Loy, C. 2013. Cumulative attribute space for age and crowd density estimation. In *CVPR*.

Dai, F.; Liu, H.; Ma, Y.; Cao, J.; Zhao, Q.; and Zhang, Y. 2019. Dense scale network for crowd counting. *CoRR* abs/1906.09707.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. *CoRR* abs/1704.01212.

Idrees, H.; Saleemi, I.; Seibert, C.; and Shah, M. 2013. Multi-source multi-scale counting in extremely dense crowd images. In *CVPR*.

Idrees, H.; Tayyab, M.; Athrey, K.; Zhang, D.; Al-Maadeed, S.; Rajpoot, N.; and Shah, M. 2018a. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*.

Idrees, H.; Tayyab, M.; Athrey, K.; Zhang, D.; Al-Maadeed, S.; Rajpoot, N.; and Shah, M. 2018b. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*.

Jiang, X.; Xiao, Z.; Zhang, B.; Zhen, X.; Cao, X.; Doermann, D.; and Shao, L. 2019. Crowd counting and density estimation by trellis encoder-decoder networks. In *CVPR*.

Kang, D.; Ma, Z.; and Chan, A. B. 2018. Beyond counting: Comparisons of density maps for crowd analysis tasks—counting, detection, and tracking. *TCSVT* 29(5):1408–1422.

Lempitsky, V., and Zisserman, A. 2010. Learning to count objects in images. In *NeurIPS*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2016. Gated graph sequence neural networks. In *ICLR*.

Li, X.; Yang, F.; Cheng, H.; Chen, J.; Guo, Y.; and Chen, L. 2017. Multi-scale cascade network for salient object detection. In *ACM MM*.

Li, X.; Yang, F.; Cheng, H.; Liu, W.; and Shen, D. 2018. Contour knowledge transfer for salient object detection. In *ECCV*.

Li, Y.; Zhang, X.; and Chen, D. 2018. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*.

Lian, D.; Li, J.; Zheng, J.; Luo, W.; and Gao, S. 2019. Density map regression guided detection network for rgb-d crowd counting and localization. In *CVPR*.

Liu, N.; Long, Y.; Zou, C.; Niu, Q.; Pan, L.; and Wu, H. 2019. Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding. In *CVPR*.

Liu, W.; Salzmann, M.; and Fua, P. 2019. Context-aware crowd counting. In *CVPR*.

Liu, X.; van de Weijer, J.; and Bagdanov, A. D. 2018. Leveraging unlabeled data for crowd counting by learning to rank. In *CVPR*.

Liu, C.; Weng, X.; and Mu, Y. 2019. Recurrent attentive zooming for joint crowd counting and precise localization. In *CVPR*.

Meng, Z.; Adluru, N.; Kim, H. J.; Fung, G.; and Singh, V. 2018. Efficient relative attribute learning using graph neural networks. In *ECCV*.

Nie, X.; Feng, J.; Zuo, Y.; and Yan, S. 2018. Human pose estimation with parsing induced learner. In *CVPR*.

Pham, V.-Q.; Kozakaya, T.; Yamaguchi, O.; and Okada, R. 2015. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *ICCV*.

Qi, S.; Wang, W.; Jia, B.; Shen, J.; and Zhu, S.-C. 2018a. Learning human-object interactions by graph parsing neural networks. In *ECCV*.

Qi, S.; Wang, W.; Jia, B.; Shen, J.; and Zhu, S.-C. 2018b. Learning human-object interactions by graph parsing neural networks. In *ECCV*.

Sam, D. B.; Surya, S.; and Babu, R. V. 2017. Switching convolutional neural network for crowd counting. In *CVPR*.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *TNNLS* 20(1):61–80.

Shen, Z.; Xu, Y.; Ni, B.; Wang, M.; Hu, J.; and Yang, X. 2018. Crowd counting via adversarial cross-scale consistency pursuit. In *CVPR*.

Shi, Z.; Zhang, L.; Liu, Y.; Cao, X.; Ye, Y.; Cheng, M.-M.; and Zheng, G. 2018. Crowd counting with deep negative correlation learning. In *CVPR*.

Shi, M.; Yang, Z.; Xu, C.; and Chen, Q. 2019. Revisiting perspective information for efficient crowd counting. In *CVPR*.

Si, C.; Jing, Y.; Wang, W.; Wang, L.; and Tan, T. 2018. Skeleton-based action recognition with spatial reasoning and temporal stack learning. In *ECCV*.

Sindagi, V. A., and Patel, V. M. 2017. Generating high-quality crowd density maps using contextual pyramid cnns. In *CVPR*.

Varior, R. R.; Shuai, B.; Tighe, J.; and Modolo, D. 2019. Scale-aware attention network for crowd counting. *CoRR* abs/1901.06026.

Viola, P.; Jones, M.; et al. 2001. Rapid object detection using a boosted cascade of simple features.

Viola, P.; Jones, M. J.; and Snow, D. 2005. Detecting pedestrians using patterns of motion and appearance. *IJCV* 63(2):153–161.

Wan, J.; Luo, W.; Wu, B.; Chan, A. B.; and Liu, W. 2019. Residual regression with semantic prior for crowd counting. In *CVPR*.

Wang, Q.; Gao, J.; Lin, W.; and Yuan, Y. 2019a. Learning from synthetic data for crowd counting in the wild. In *CVPR*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019b. Dynamic graph cnn for learning on point clouds. *TOG*.

Yang, F.; Li, X.; Cheng, H.; Guo, Y.; Chen, L.; and Li, J. 2018. Multi-scale bidirectional fcn for object skeleton extraction. In *AAAI*.

Zhang, C.; Li, H.; Wang, X.; and Yang, X. 2015. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*.

Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; and Ma, Y. 2016. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*.

Zhao, H.; Shi, J.; Qi, X.; Wang, X.; and Jia, J. 2017. Pyramid scene parsing network. In *CVPR*.

Zhao, M.; Zhang, J.; Zhang, C.; and Zhang, W. 2019. Leveraging heterogeneous auxiliary tasks to assist crowd counting. In *CVPR*.