Perspective-Aware CNN For Crowd Counting

Miaojing Shi, Zhaohui Yang, Chao Xu, Qijun Chen

Perspective-Aware CNN For Crowd Counting

Miaojing Shi, Zhaohui Yang, Chao Xu, *Member, IEEE,* and Qijun Chen, *Senior Member, IEEE*

***Abstract*—Crowd counting is the task of estimating pedestrian numbers in crowd images. Modern crowd counting methods employ deep neural networks to estimate crowd counts via crowd density regressions. A major challenge of this task lies in the drastic changes of scales and perspectives in images. Representative approaches usually utilize different (large) sized filters and conduct patch-based estimations to tackle it, which is however computationally expensive. In this paper, we propose a perspective-aware convolutional neural network (PACNN) with a single backbone of small filters (e.g. 3×3). It directly predicts a perspective map in the network and encodes it as a perspective-aware weighting layer to adaptively combine the density outputs from multi-scale feature maps. The weights are learned at every pixel of the map such that the final combination is robust to perspective changes and pedestrian size variations. We conduct extensive experiments on the ShanghaiTech, WorldExpo'10 and UCF_CC_50 datasets, and demonstrate that PACNN achieves state-of-the-art results and runs as fast as the fastest.**

***Index Terms*—Crowd Counting, Perspectives, Scales, PACNN, Fastest.**

I. Introduction

The rapid growth of the world's population has led to fast urbanization and resulted in more frequent crowd gatherings, e.g. sport events, music festivals, political rallies. Accurate and fast crowd counting therefore becomes essential in order to handle large crowds for public safety.

Traditional crowd counting methods estimate crowd counts via the detection of each individual pedestrian [2], [3], [4], [5]. Recent methods conduct crowd counting via the regression of density maps [6], [7], [8], [9]; the problem of crowd counting is casted as estimating a continuous density function whose integral over an image gives the count of pedestrians within that image [7], [10], [11], [12], [1], [13], [14] (see Fig. 1: Density Map). Handcrafted features were firstly employed in the density regression [7], [10], [11] and soon outperformed by deep representations [12], [1], [13].

A major challenge of this task lies in the drastic perspective changes in crowd images. The perspective problem is related to camera calibration which estimates a camera's 6 degrees-of freedom (DOF) [15]. Besides the camera DOFs, it also represents the 3d scene in a simplified form similar to a depth map [16]. For instance, in Fig. 1, the so-called perspective map signifies the pedestrian size and scale variation within an image [6], [17], [18], [1], [13], [14], [19]. To address the varying pedestrian scales in modern convolutional neural networks (CNN) [20], researchers 1) employ different filter sizes [13], [21], [12], [22], [23], [14], [24], [25], [19] and 2) adopt patch-based estimations [1], [12], [22], [14], [24], [25]. Filters with different sizes are utilized in a multi-column fashion [26], where different outputs are merged in the end to adapt the network to various pedestrian sizes [13], [21], [12]. While the patch-based estimation divides a crowd image into patches and infers them from the network with different global/local contexts; it is often integrated with the multi-column architecture [22], [14].

Despite the significant improvements achieved by [1], [13], [12], [22], [14], [22], [24], [25], [19], the computational complexity has increased dramatically by using large sized filters and patch-based inference. We decide to not follow this trend. We notice that in spite of the direct relation between perspective value and object size, perspective information in previous works has only been encoded implicitly into the ground truth density maps [1], [13], [14], [19]; the networks were unaware of any perspective information during training. Building upon this observation, we propose a perspective-aware CNN (PACNN) to explicitly take advantage of the perspective information in the network.

We first introduce the ground truth perspective map for network training. It is justified as a linear function of the vertical axis of the image plane by using the projective geometry. We fit this linear function based on perspective values computed at sampled pixels. Previous works [1], [6] labeled pedestrian heights to compute the sampled perspective values; while we propose a new way without incurring additional annotations. Afterwards, we regress density maps from two different scales of a single backbone network. The network has small filters (e.g. 3×3) only. We adaptively combine the two density outputs via a perspective-aware weighting layer, where the weights are learned through a nonlinear transform of a predicted perspective map in the network. The final output is robust to the varying perspectives and scales; we thereby infer the crowd density directly over an entire image.

We conduct extensive experiments on several standard benchmarks i.e. ShanghaiTech [13], WorldExpo'10 [1] and UCF_FF_50 [9]. We show that our PACNN outperforms the state-of-the-art in accuracy and runs as fast as the fastest.

II. Related work

We categorize the literature in crowd counting into either shallow or deep methods. Deep methods refer to those employ CNNs while shallow methods do not.

A. Shallow methods

Detection-based methods. These methods consider a crowd as a group of detected individual pedestrians [27], [2], [28],

M. Shi is with the Linkmedia team, Inria Rennes, Rennes Cedex 35042, France (e-mail: miaojing.shi@inria.fr).

Z. Yang and C. Xu are with the Key Laboratory of Machine Perception, Ministry of Education, Cooperative Medianet Innovation Center, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China (e-mail: zhaohuiyang@pku.edu.cn; xuchao@cis.pku.edu.cn).

Q. Chen is with the Key Laboratory of Embedded System and Service Computing, School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China (e-mail: qjchen@tongji.edu.cn).

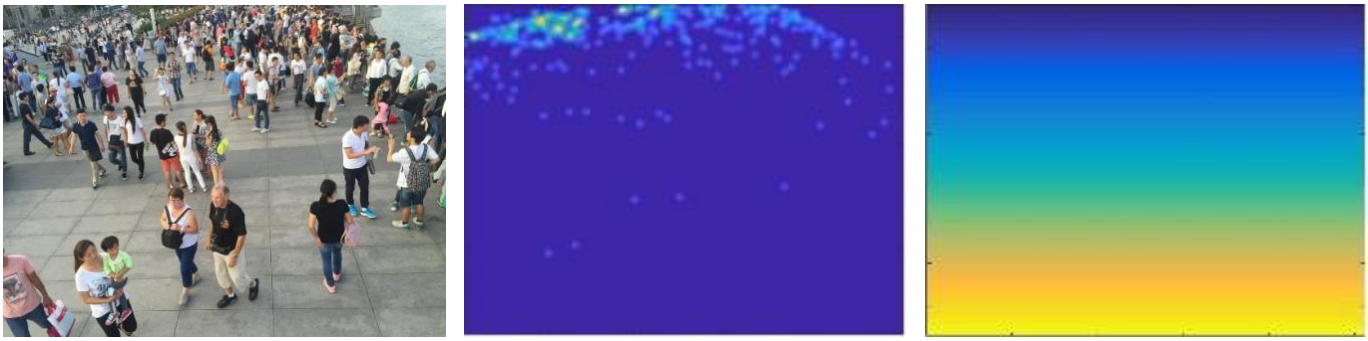

Fig. 1: A crowd image with its density map and perspective map. The density map shows the (spatially smoothed) crowd count at every location in the image. The perspective map shows the vertical scale at every location in the image, i.e. its value at each location indicates how many pixels correspond to a human height of one meter; values are set to be the same in the same row of the map [1]. Yellow in both maps indicates big values while blue corresponds to small values. We combine the information of density and perspective to estimate crowd count in the image.

[3], [4], [5]. They can be performed either in a monolithic manner or part-based. Monolithic approach typically refers to pedestrian detection that employs hand-crafted features like Haar [29] and HOG [30] to train an SVM or AdaBoost detector [28], [3], [4], [5]. These approaches often perform poorly in the dense crowds where pedestrians are heavily occluded or overlapped. Part-based detection is therefore adopted in many works [31], [27], [2] to count pedestrian from parts in images. Despite the improvements achieved, the detection-based crowd counting overall suffers severely in dense crowds with complex backgrounds.

Regression-based methods. These methods basically have two steps: first, extracting effective features from crowd images; second, utilizing various regression functions to estimate the crowd count. Regression features include edge features [6], [7], [8], [32], [17], texture features [7], [9], [33], [17] etc.. Regression methods include linear [32], [34], ridge [7] and Gaussian [6], [17] functions. Earlier works ignore the spatial information by simply regressing a scalar value (crowd count), later works instead learn a mapping from local features to a density map [7], [10], [11]. The crowd count is obtained by integrating over the density map. Spatial locations of pedestrians are encoded into the density map; regressing the map locally is in general easier than estimating the crowd count globally.

B. Deep methods

Due to the use of strong CNN features, recent works on crowd counting have shown remarkable progress [35], [36], [1], [37], [13], [21], [12], [38], [23], [22], [14], [39], [40], [24], [41], [25]. In order to deal with the varying head scales in one image, multi-column [13], [21], [12], [14], [37] or multi-scale [42], [41] network architecture is utilized for crowd density regression. The three-column architecture is particularly popular [26], [13]; it employs different sized filters corresponding to different columns; the large filters make the network hard to train. Sam et al. [14] introduce a switch classifier to relay the crowd patches from images to their best CNN columns with most suitable scales. The switching is costly and sometimes not correct. Zeng et al. [42] introduce a multi-scale blob to simulate the inception module in the network while Zhang et al. [41] concatenate three feature maps of different scales in the network for crowd counting. Sindagi et al. [22] design a system called contextual pyramid CNN. It consists of both a local and global context estimator to perform patch-based density estimation; particularly, local context is obtained using a sliding window strategy at the testing stage. To better exploit the temporal information in a video frame for crowd counting, Xiong et al. [24] propose a variant of the convolutional LSTM (ConvLSTM); they also use the sliding window strategy for test images. Huang et al. [19] formulate the key factors of crowd counting as semantic scene models, which includes the body part map and the structured density map. The crowd count is estimated based on the two semantic scene models. Liu et al. [25] design a so-called DecideNet to estimate the crowd density by generating detection- and regression-based density maps separately; the final crowd count is obtained with the guidance of an attention module which adopts suitable estimation from the two different kinds of density maps.

Additional annotations are provided to obtain perspective maps, and are utilized in the ground truth generalization of density maps [1], [13], [14], [19]. [1] performs a multi-task optimization to regress both the density map and the crowd count. It uses fully connected layers with fixed input required; network inference on a test image is conducted in an exhaustive manner with sliding windows. PACNN is also performed in a multi-task manner like in [1], [38], [41], [25], [19], but differently, it regresses two density maps and one perspective map with fully convolutional network. The perspective map is used to adaptively combine the two-scale density outputs from the network. Our network is aware of the perspective change in an image and is therefore able to cope with it. Our method, without using large filters [13], [21], [12], [22], [14], [24], [25], [19] and patch-based inference [1], [12],

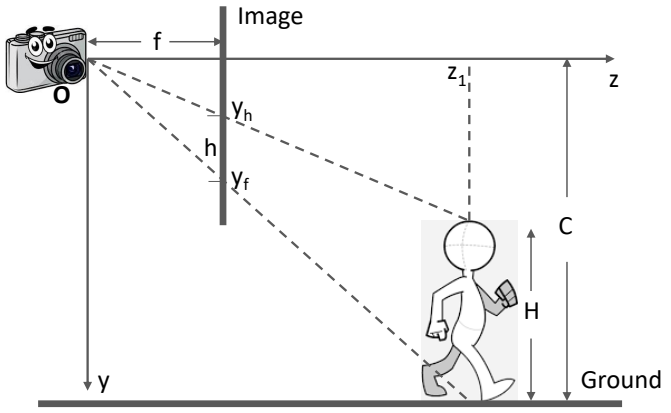

Fig. 2: The perspective geometry of a pinhole camera as seen from the x-axis.

[22], [14], [24], [25], outperforms the state-of-the-art with high efficiency.

Perspective provides additional information regarding the pedestrian size and scale in an image. Many other works of different objectives also try to learn or leverage additional cues to address the perspective effects in images [16], [43], [44]. For instance, [16] utilizes a depth map to predict the size of objects in the wild and count them; [44] designs scale-adaptive convolutions to learn the scaling ratios of the convolutional patches and resize them for scene parsing.

III. Perspective-aware CNN

In this section we first generate ground truth density maps and perspective maps; then we introduce the network architecture; finally we present the network training protocol.

A. Ground truth (GT) generation

GT density map generation. Pedestrian head center annotations are available for public datasets. The GT density map D^g can be generated using kernel density estimation (KDE) [45]. Following [13], [14], [22], a Gaussian kernel G_σ is employed to smooth at every head location z_j,

$$D_g = \sum_{j=1}^{Y^g} G_\sigma(z - z_j), \tag{1}$$

where Y^g denotes the total number of pedestrians in an image. The integral of the density map is equivalent to Y^g (see Fig. 1).

GT perspective map generation. The *perspective map* was first proposed in [6] and then used in [6], [17], [1], [13], [14]. The perspective value at every pixel of a GT map $P^g = \{p_j^g\}$ is defined as the number of pixels in the image representing one meter at that location in the real scene [1].

Two assumptions are made here as in previous works to create the perspective map: **a**. there is no in-plane rotation; **b**. the ground in the captured scene is flat. In the standard crowd counting benchmarks, the first assumption holds for the vast majority of the images; only a few were shot with a rotation of less than 15 degrees (roughly). The second assumption holds for almost all. Building upon this, we visualize the perspective geometry of a pinhole camera in Fig. 2: the Cartesian coordinate system starts from origin **O**, with y-axis representing the vertical direction while z-axis the optical axis (depth). The original 3d geometry of the pinhole camera is thus seen from x-axis in a 2d figure. Supposing a person with true height H is walking on the ground, he was shot by a camera located at **O** where the camera aperture is. The pedestrian's head top and feet bottom are mapped on the image plane at y_h and y_f, respectively. The distance from the camera aperture to the image plane is f, which is also known as the focal length; while the camera height from the ground is C. Solving the similar triangles we have:

$$\begin{aligned} y_h &= \frac{f(C-H)}{z_1} \\ y_f &= \frac{fC}{z_1}. \end{aligned} \tag{2}$$

The observed pedestrian height h is thereby,

$$h = y_f - y_h = \frac{fH}{z_1} \tag{3}$$

h is directly related to the predefined perspective value; dividing it by y_h will give us

$$h = \frac{H}{C-H} y_h. \tag{4}$$

In a crowd counting scenario, the height difference among pedestrians is relatively small compared to the distance range from the camera to the closest and remotest pedestrians. Likewise in [1], H is assumed to be the mean height of adults (1.75m) for every pedestrian while C is fixed for each image; we can therefore conclude that the pedestrian height in an image is linearly related to its head position along the y-axis (vertical). The perspective value, which is defined as the observed pedestrian height of one meter, is then a linear function of y: $f(y, \theta)$, where θ is related to the camera parameter (e.g. camera height C). To obtain θ, we fit $f(y, \theta)$ based on the perspective values computed at sampled pixels. Given a perspective map, **i**. its value are linearly increased from the top to the bottom; **ii**. the values in the same row are equal. These two constrains have been applied in previous works [6], [1], we provide a detailed justification of it. Note that the pinhole camera is illustrated upright in Fig. 2 without tilting. The camera's tilt will change the position of the horizon. For this more complicated case, we refer the readers to [46]: the linear relation between h and y still holds.

To compute the perspective values, in [6], [1] authors manually labeled several adult pedestrians from head to toe, such that the perspective value p_j^g at sampled head center j is given by:

$$p_j^g = \frac{h_j}{H} = \frac{h_j}{1.75}, \tag{5}$$

where h_j denotes the pedestrian height at pixel j in the image. A linear regression is employed afterwards to obtain the entire GT perspective map (Fig .1).

Equation (5) is effective when pedestrians are not occluded. Unfortunately, in many crowd counting scenarios pedestrians are densely distributed and only their heads can be perceived (Fig. 4, top row). To obtain the perspective information in

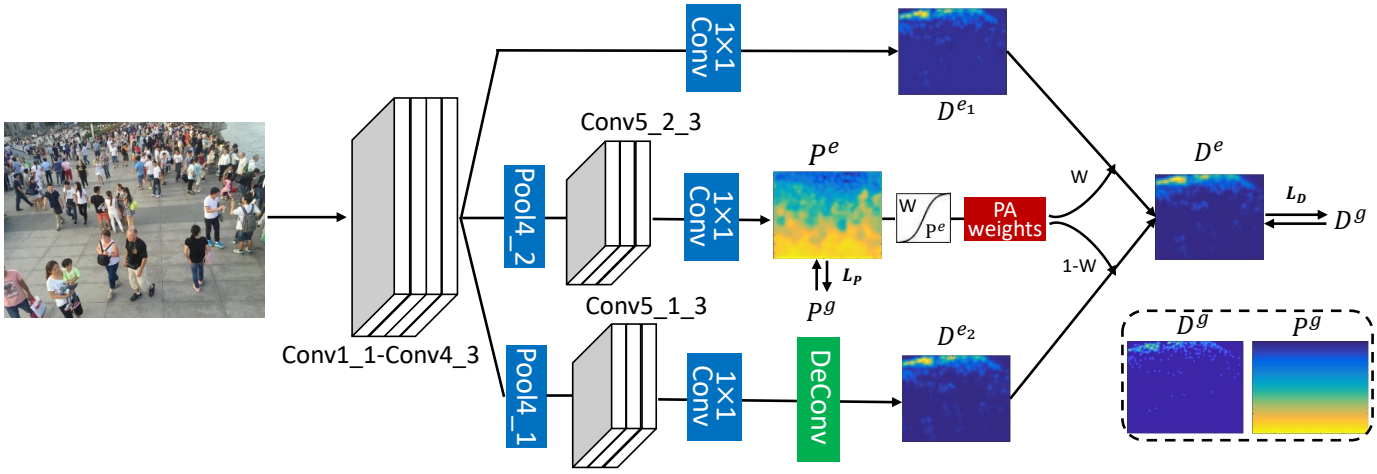

Fig. 3: The structure of the proposed perspective-aware convolutional neural network (PACNN) for crowd counting. D and P denote the density map and the perspective map, while e and g stand for estimation and ground truth. We build a single backbone network with 3×3 filters. We regress two density maps D^{e_1} and D^{e_2} from Conv4_3 and Conv5_1_3 and one perspective map P^e from Conv5_2_3 using 1×1 Conv with 1 output. D^{e_1} and D^{e_2} fire on crowds of different scales. We adaptively combine them via a perspective-aware (PA) weighting layer, where the PA weights are learned via a nonlinear transform of P^e. We optimize the network over the loss with respect to the ground truth of both D^g and P^g. The final output D^e is robust to the perspective change and close to the ground truth.

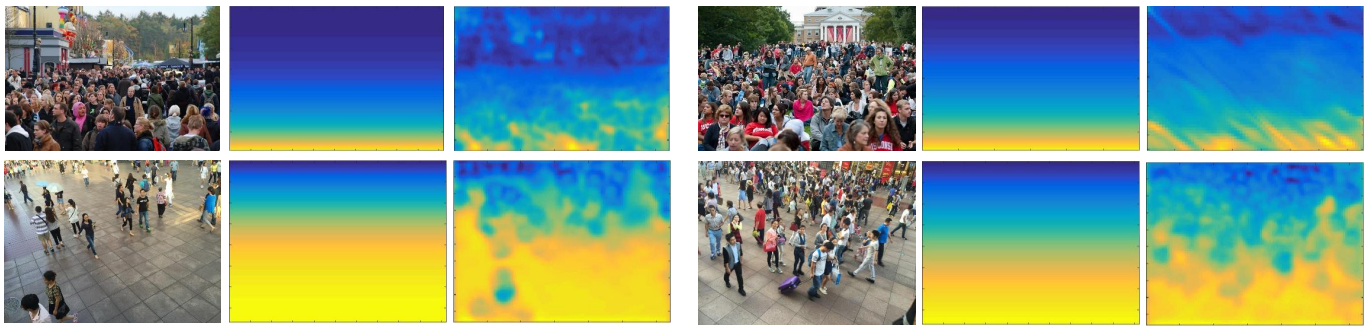

Fig. 4: Top row: samples from ShanghaiTech PartA [13]; bottom row: PartB. Each row has two blocks. In each block, left column: image; middle column: GT perspective map; right column: estimated perspective map. Blue in the heatmaps indicates small perspective values while yellow indicates large values. GT perspective maps are generated using (6) and (5) for ShanghaiTech PartA and PartB, respectively.

dense crowds, we instead interpret the perspective value with the pedestrian head size. Similarly to the pedestrian height, the head size also changes linearly along the y-axis of the image. Following [13], we relate the size of certain head at pixel j (the perspective value p_j^g) to the distances $dist_{kj}$ from this head to its K-nearest neighbors:

$$p_j^g = \frac{1}{K} \sum_{k=1}^{K} dist_{kj}. \tag{6}$$

We use linear regression to generate the whole perspective map.

Both (5) and (6) reflect the perspective and scale changes in crowd images. We illustrate some examples in Fig. 4. For datasets with relatively sparse crowds and visible pedestrian bodies (e.g. ShanghaiTech PartB [13], WorldExpo'10 [1]), (5) is adopted to compute the GT perspective maps (Fig. 4: bottom row); while for datasets with relatively dense crowds (e.g. ShanghaiTech PartA [13], UCF_CC_50 [9]), (6) is adopted (Fig. 4: top row). Note that although some areas in images contain no/few pedestrians, we can still generate the perspective values due to **i** and **ii**. With the help of (5,6), we are able to copy with images across different benchmarks.

B. Network architecture

The proposed PACNN employs a multi-task learning scheme to regress two density maps and one perspective map. These subtasks share the same backbone as shown in Fig. 3. Following [37], [41], we use 3×3 filters in the network convolutions. The computation required for 3×3 filters is far less than for large filters [13], [14], [23], [22]. Multiple stacked 3×3 filters have similar receptive fields as those of large sized filters. Our network with two density outputs indeed simulates the popular multi-column architecture with different sized filters [13], [22], [14]. PACNN requires less training time

in practice because of the shared low-level parameters. We adopt three convolutional layers after each pooling layer in the backbone; we use the same number of channels as with VGG [47] at each layer. Out of the output of Conv4_3, we branch off several data streams to perform the density and perspective regressions, which are described next.

Density map regression. We regress two density maps from the outputs of Conv4_3 and Conv5_1_3 simultaneously (see Fig. 3). The filter from the deeper layer Conv5_1_3 has a bigger receptive field than that from the shallower layer Conv4_3. Thus a combination of the two density maps is supposed to adapt to both the big and small pedestrian heads in an image.

Due to the pooling operations, the two density maps have different sizes: the output of Conv4_3 is of $1/8$ resolution of the input, while the output of Conv5_1_3 is of $1/16$ resolution of the input. To combine them, we use a deconvolutional layer [48] to upsample the density output from Conv5_1_3 to the same size with that of Conv4_3. A straightforward combination would be averaging the two outputs:

$$D^e = (D^{e_1} + D^{e_2})/2 \tag{7}$$

where we use $D^{e_1} = \{d_j^{e_1}\}$ and $D^{e_2} = \{d_j^{e_2}\}$ to denote the two estimated density maps; j signifies the j-th pixel in the map. They are regressed from Conv4_3 and Conv5_1_3 using 1×1 filters (1×1 Conv with 1 output in Fig. 3). The combined map D^e is of $1/8$ resolution of the input. We need to downsample the corresponding ground truth density map by three times. We illustrate this simple combination in Fig. 5(a).

Perspective map regression. The perspective regression is branched off from Pool4_2 with three more convolutional layers Conv5_2_1 to Conv5_2_3. We use $P^e = \{p_j^e\}$ to denote the regressed perspective map after Conv5_2_3. We set the stride of Pool4_2 as 1 such that P^e has $1/8$ resolution of the input. We need to downsample the ground truth perspective map by three times. We present some estimated perspective maps P^e and their corresponding ground truths P^g in Fig. 4.

Perspective-aware weighting. D^{e_1} is good at estimating small heads while D^{e_2} is good at detecting big heads. We know that the pedestrian size changes in accordance with the perspective value. To make use of the estimated perspective map P^e, we add a perspective-aware (PA) weighting layer in the network (see Fig. 3) so that in general it gives D^{e_1} higher weights on smaller heads and vice versa for D^{e_2}:

$$D^e = W \odot D^{e_1} + (1 - W) \odot D^{e_2}, \tag{8}$$

where $\odot$ denotes the element-wise (Hadamard) product. $W = \{w_j\}$ is the output of the perspective-aware weighting layer; it is obtained by applying a nonlinear transform $w_j = f(p_j^e)$ to the perspective values p_j^e (we also tried with linear transform, which does not work as well as nonlinear transform.). This nonlinear function needs to be differentiable and guarantee a positive relationship between w_j and p_j^e. We choose the sigmoid function:

$$w_j = f(p_j^e) = \frac{1}{1 + e^{-\alpha * (p_j^e - \beta)}}, \tag{9}$$

where α and β are the two parameters that can be learned via back propagation. $w_j \in (0, 1)$, it varies at every pixel of the density map. The backwards function of the PA weighting layer computes partial derivative of the loss function L with respect to α and β. We will discuss the loss function later. Here we write out the chain rule:

$$\begin{aligned} \frac{\partial L}{\partial \alpha} &= \frac{\partial L}{\partial D^e} \frac{\partial D^e}{\partial W} \frac{\partial W}{\partial \alpha} \\ &= \sum\nolimits_j \frac{\partial L}{\partial d_j^e} (d_j^{e_1} - d_j^{e_2}) \frac{\partial w_j}{\partial \alpha} \\ &= \sum\nolimits_j \frac{\partial L}{\partial d_j^e} (d_j^{e_1} - d_j^{e_2}) (p_j^e - \beta) f(p_j^e)(1 - f(p_j^e)); \end{aligned} \tag{10}$$

Similarly, we have

$$\frac{\partial L}{\partial \beta} = \sum\nolimits_j \frac{\partial L}{\partial d_j^e} (d_j^{e_1} - d_j^{e_2})(-\alpha) f(p_j^e)(1 - f(p_j^e)). \tag{11}$$

Compared to (7), where the average operator simply gives the same weights on D^{e_1} and D^{e_2} at every pixel, the proposed perspective-aware weighting scheme (8) encodes the perspective information into the density map combination. The weighted output is immune to the perspective change in an image. In Fig. 3 the crowd count integrated from D^e (352) is closer to the ground truth (355), compared to that of D^{e_1} (332), D^{e_2} (361), or their average (347).

C. Network training

We adopt the Euclidean distance to measure the difference between the estimated map and ground truth map in a specific task O, the loss function is defined as:

$$L_{\mathrm{O}}(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \|E(X_i; \Theta) - G_i\|_2^2, \tag{12}$$

where Θ is a set of learnable parameters in the proposed network. X_i is the input image and N is the number of training images. We denote by E and G the respective estimated map and ground truth map for task O. For the perspective regression task P, we obtain its loss L_{P} by substituting P^e and P^g into E and G, respectively; while for the density regression task D, we obtain its loss L_{D} by replacing E and G with D^e and D^g correspondingly.

We train the model from scratch using random initializations. The network is optimized with Stochastic Gradient Descent (SGD) in two phases. **Phase 1**: we optimize the density regression using (7): the two density maps D^{e_1} and D^{e_2} are simply averaged to produce the final map (see Fig. 5(a)); **Phase 2**: we finetune the model by adding the perspective-aware weighting scheme; the model is jointly optimized by using the predicted perspective map to combine the two density maps (8). The final loss function is a summation of the two losses L_{P} and L_{D}: $L = L_{\mathrm{P}} + L_{\mathrm{D}}$.

In some datasets/scenarios, perspective maps are provided for both training and test. It can be provided by the camera itself or through specific annotations. In either case, we no longer need to estimate the perspective maps for test images; the network architecture is simplified as in Fig. 5(b), where we use the ground truth perspective values to generate PA

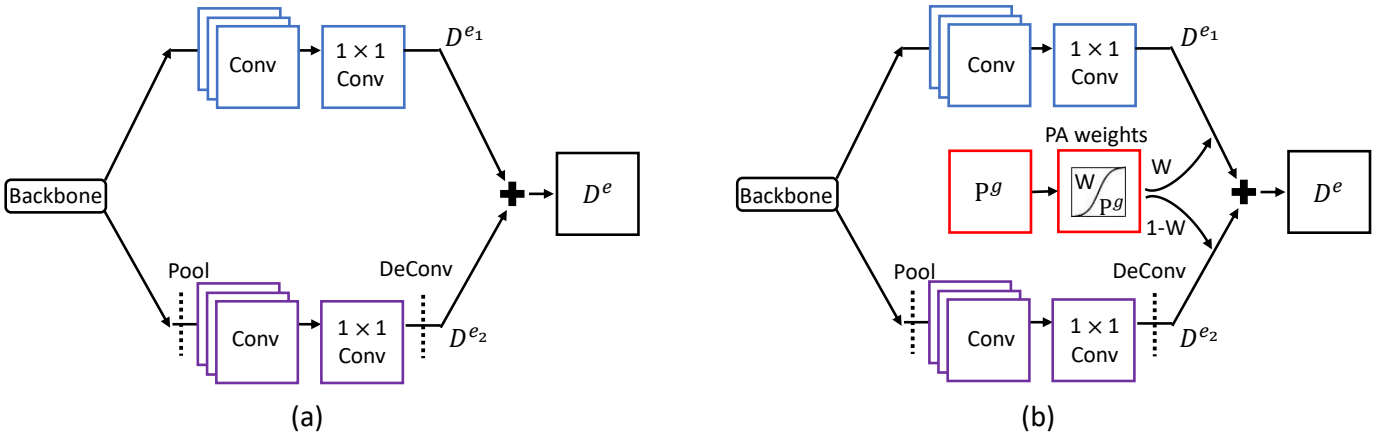

Fig. 5: Two variants of PACNN. (a) Density maps D^{e_1} and D^{e_2} are regressed and combined to produce $D^e = (D^{e_1} + D^{e_2})/2$; (b) Density maps D^{e_1} and D^{e_2} are regressed and adaptively combined via PA weights W, which are transformed from the ground truth perspective map P^g.

weights and combine the two density maps accordingly. The loss function will be only the density regression loss, $L = L_D$.

IV. Experiments

We first introduce three standard datasets for crowd counting. Afterwards, we evaluate our method on these datasets. Additionally, in the appendix we provide results for the perspective estimation task. We shall publish our code and data to reproduce our experiments.

A. Datasets

ShanghaiTech [13]. It consists of 1,198 annotated images with a total of 330,165 people with head center annotations. This dataset is split into two parts: PartA and PartB. The crowd images are sparser in PartB compared to PartA: the average crowd counts are 123.6 and 501.4, respectively. Following [13], we use 300 images for training and 182 images for testing in PartA; 400 images for training and 316 images for testing in PartB.

WorldExpo'10 [1]. It includes 3,980 frames, which are taken from the Shanghai 2010 WorldExpo. 3,380 frames are used as training while the rest are taken as test. The test set includes five different scenes and 120 frames in each one. Regions of interest (ROI) are provided in each scene so that crowd counting is only conducted in the ROI in each frame. The crowds in this dataset are relatively sparse in comparison to other datasets with an average pedestrian number of 50.2 per image.

UCF_CC_50 [9]. It has 50 images with 63,974 head annotations in total. The head counts range between 94 and 4,543 per image. The small dataset size and large variance in crowd count make it a very challenging dataset. Following [9], we perform 5-fold cross validations to report the average test performance.

B. Implementation details and evaluation protocol

Ground truth annotations for each head center in the standard benchmarks are publicly available. For WolrdExpo'10, the ground truth perspectives are provided for training. For ShanghaiTech PartB, we use (5) to generate the ground truth perspectives; while for PartA and UCF_CC_50 we use (6) since the crowds are much denser. Given a training set, we augment it by randomly cropping 9 patches from each image. Each patch is $1/4$ size of the original image. All patches are used to train our model PACNN. We set the batch size as 1, learning rate 1e-6 and momentum 0.9. Referring to Sec. III-C, we train 100 epochs in Phase 1 while 150 epochs in Phase 2. Network inference is on the entire image.

We evaluate the performance via the mean absolute error (MAE) and root mean square error (RMSE) commonly used in previous works [1], [13], [14], [12], [36]:

$$\begin{aligned} \text{MAE} &= \frac{1}{N}\sum_{i=1}^{N}|Y_i^e - Y_i^g|, \\ \text{RMSE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i^e - Y_i^g)^2}, \end{aligned} \tag{13}$$

where Y_i^e and Y_i^g denotes the estimated and the ground truth crowd counts, respectively. Y_i^e is obtained by integrating the density map D_i^e (8). Small MAE and RMSE values indicate good performance.

C. Results on ShanghaiTech

Ablation study. We conduct an ablation study to justify the utilization of multi-scale and perspective-aware weighting schemes in PACNN. Referring to Sec. III, PACNN is presented after introducing a direct average manner of the multi-scale density outputs (Fig. 5(a)). Perspective information is injected into the network to adaptively combine the two density outputs. The perspective map can either be provided as a ground truth (P^gACNN, Fig. 5(b)) or generated via the network (PACNN, Fig. 3).

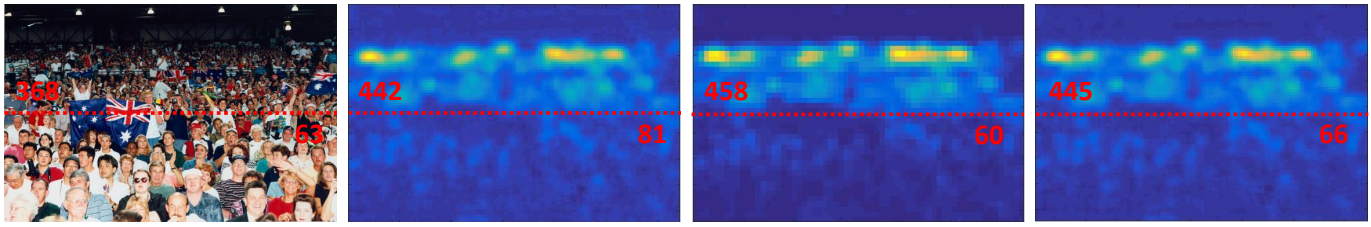

Fig. 6: An example for the ablation study. The numbers below the images indicate the estimated and ground truth crowd counts. The red number inside each image indicates the crowd count for its upper or lower half. The prediction of D^{e_2} is inferior to D^{e_1} on the dense area (upper half) with visible pixel blocks (large receptive fields); while on the sparse area (lower half), D^{e_2} is cleaner than D^{e_1}. Their combination D^{e} in PACNN takes advantage of both and results in a better prediction overall.

We first train a network following (7) to simply average the outputs of the two density maps D^{e_1} and D^{e_2}. We report the test result on ShanghaiTech in Table I. D^{e_1} is taken from Conv4_3 while D^{e_2} is taken from Conv5_1_3. Referring to Sec. III-B, D^{e_2} should fire more on bigger heads than D^{e_1} and vice versa. Having a look at Table I, the MAE for D^{e_1} and D^{e_2} on ShanghaiTech PartA are 90.8 and 93.3, respectively; on ShanghaiTech PartB they are 18.4 and 16.9, respectively. Crowds in PartA are much denser than in PartB (see Sec. IV-A), while pedestrian heads in PartA are smaller than in PartB (see Fig. 7). It reflects in Table I that D^{e_1} in general performs better on PartA while D^{e_2} performs better on PartB. Fig. 6 gives an example of this ablation study.

To justify the PA weighting scheme, we compare PACNN with $(D^{e_1}+D^{e_2})/2$ in Table I. Directly averaging over pixels of D^{e_1} and D^{e_2} produces a slightly better MAE (90.2) than that (90.8) of D^{e_1} on PartA, whilst a worse MAE (17.1) than that (16.9) of D^{e_2} on PartB. In contrast, using PA weights to adaptively combine D^{e_1} and D^{e_2} clearly decreases both the MAE and RMSE: they are 85.6 and 136.7 on PartA; 15.3 and 25.4 on PartB, respectively. PACNN assigns bigger weights on D^{e_1} in the small head area while bigger weights on D^{e_2} in the big head area. It is robust to perspective and scale changes in crowd images.

As mentioned in Sec. III-C, ground truth perspective information can be provided for test, PACNN is thereby simplified as in Fig. 5(b). We denote this modified version by P$^{\text{g}}$ACNN in Table I. It achieves the lowest MAE (84.5 and 14.2) and RMSE (132.5 and 24.1) on ShanghaiTech PartA and PartB, respectively. Notice PACNN in fact performs close to P$^{\text{g}}$ACNN, which demonstrates that the estimated perspective maps in PACNN are sufficient to combine the multi-scale density outputs.

Comparison to state-of-the-art. We compare PACNN with state-of-the-art [1], [13], [14], [22] in Table I. On ShanghaiTech PartA, the lowest MAE and RMSE are achieved by [22] (73.6 and 106.4); while ours are the second best. (P$^{\text{g}}$ACNN: 84.5 and 132.5). On ShanghaiTech PartB, the lowest MAE and RMSE are achieved by our PACNN (15.3 and 25.4) and P$^{\text{g}}$ACNN (14.2 and 24.1), which significantly outperform the state-of-the-art (20.1 and 30.1).

As mentioned in Sec. I, [1], [13], [14], [22] employ large filters (9×9, 7×7, 5×5) in their networks and [1], [14], [22] adopt patch-based inference on test images. They use a multi-column [13], [14], [22] or multi-task [1], [14], [22] architecture. In contrast, PACNN performs a multi-task optimization without using large filters and patch-based inference. To quantitatively compare the time complexity of different approaches, we report the average inference time per image on ShanghaiTech PartB in Table I: right part. We implement our method and those of [1], [13], [14], [22] in the same Caffe [49] framework. We use an image input size of 1024*768 and an Nvidia GTX Titan X GPU. Our PACNN has an inference time of 0.12s per image, which is comparable to the fastest method of [13] (0.11s). Note however that they use fewer channels in their network and we consistently outperform [13] in accuracy. [14] and [1] are 1.2x and 3.4x slower than our method. The state-of-the-art method of [22] is even 7.3x slower. Some methods use fully connected layers [1], [14], [22] and sliding window strategy [1], [22] in either their regression or classification task and hence require an additional preprocessing step to extract/resize the images/patches. Factoring in this additional cost, these methods are up to 25.9x slower than our method. Overall, we conclude that our model is effective and efficient to implement in a real-world surveillance scenario.

Fig. 6 and 7 show some examples on both PartA and PartB. The estimated density maps are visually similar to the crowd distributions in real images.

D. Results on WorldExpo'10

Referring to [1], training and test are both conducted within the ROI provided for each scene of WorldExpo'10. MAE is reported for each test scene and averaged to evaluate the overall performance. We compare our PACNN with other state-of-the-art [1], [13], [14], [22], [24], [19], [25] in Table II. It can be seen that PACNN produces the best MAE on average across scenes: 8.7.

The GT perspective maps for training images are provided in [1]. They are utilized to generate the GT density maps in [1], [13], [14], [19]: the Gaussian variance σ (1) is computed as a ratio of the perspective value $\sigma_j = \alpha * p_j^g$. [14] reports a better result without using perspective values; notations (w/o perspective) and (with perspective) in Table II signify either manner. Using perspective to compute σ (1) with a fixed parameter (e.g. $\alpha = 0.2$ [13]) might blur the original density in

ShanghaiTech	PartA		PartB		PartB (sec/image)	
Measures	MAE	RMSE	MAE	RMSE	Preprocessing	Inference
D^{e_1}	90.8	150.4	18.4	28.8	-	-
D^{e_2}	93.3	158.2	16.9	27.1	-	-
$(D^{e_1} + D^{e_2})/2$	90.2	153.6	17.1	27.6	-	-
PACNN	**85.6**	**136.7**	**15.3**	**25.4**	**0.00**	**0.12**
P^{g}ACNN	**84.5**	**132.5**	**14.2**	**24.1**	**0.00**	**0.09**
Zhang et al. [1]	181.8	277.7	32.0	49.8	2.11	0.41
Zhang et al. [13]	110.2	173.2	26.4	41.3	**0.00**	**0.11**
Sam et al. [14]	90.4	135.0	21.6	33.4	0.09	0.14
Sindagi et al. [22]	**73.6**	**106.4**	**20.1**	**30.1**	2.24	0.87

TABLE I: Ablation study of PACNN and its comparison with state-of-the-art on ShanghaiTech dataset. D^{e_1} denotes the density map regressed from Conv4_3 while D^{e_2} is regressed from Conv5_1_3 in Fig. 3. Refer to (7) and (8) for $(D^{e_1} + D^{e_2})/2$ and PACNN, respectively. P^{g}ACNN denotes the simplified version of PACNN with ground truth perspectives available (see Sec. III-C and Fig 5(b)). Left part: prediction errors in MAE and RMSE. Right part: preprocessing and inference time in seconds per image.

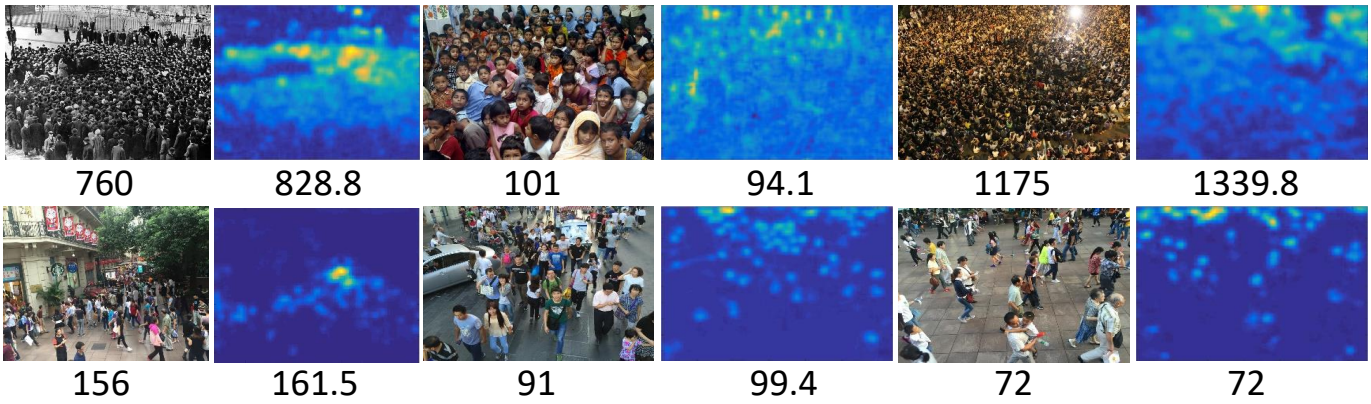

Fig. 7: Result on ShanghaiTech dataset. We present six test images and their estimated density maps on the right. The ground truth and estimated crowd counts are beneath the real images and the corresponding density maps, respectively.

WorldExpo'10	S1	S2	S3	S4	S5	Avg.
Zhang et al. [1]	9.8	14.1	14.3	22.2	3.7	12.9
Zhang et al. [13]	3.4	20.6	12.9	13.0	8.1	11.6
Sam et al. [14] (with perspective)	4.2	14.9	14.2	18.7	4.3	11.2
Sam et al. [14] (w/o perspective)	4.4	15.7	10.0	11.0	5.9	9.4
Sindagi et al. [22]	2.9	14.7	10.5	10.4	5.8	8.9
Xiong et al. [24]	6.8	14.5	14.9	13.5	3.1	10.6
Huang et al. [19]	4.1	21.7	11.9	11.0	3.5	10.5
Liu et al. [25]	2.0	13.1	8.9	17.4	4.8	9.2
PACNN	2.6	15.4	10.6	10.0	4.8	**8.7**

TABLE II: Comparison of PACNN with other state-of-the-art on WorldExpo'10 dataset. MAE is reported for each test scene and averaged in the end.

some occasions; it can result in a performance decrease [14]. In our case, the perspective information is utilized in an elaborate manner to adaptively combine the multi-scale density outputs at every pixel in PACNN.

E. Results on UCF_CC_50

We compare our method with other state-of-the-art on UCF_CC_50 [1], [13], [14], [22], [19] in Table III. Our method achieves the MAE 304.9 and RMSE 411.7, whereas Sindagi et al. achieve the lowest (295.8 and 320.9) in their work [22]. Our MAE is close to the best but RMSE is not, which means

UCF_CC_50	MAE	RMSE
Zhang et al. [1]	467.0	498.5
Zhang et al. [13]	377.6	509.1
Sam et al. [14]	318.1	439.2
Sindgai et al. [22]	**295.8**	**320.9**
Huang et al. [19]	409.5	563.7
PACNN	304.9	411.7

TABLE III: Comparison of PACNN with other state-of-the-art on UCF_CC_50 dataset.

that our method works well in general but has more extreme outliers. Recalling Table. I, the good result achieved in [22] is a tradeoff with time complexity.

V. Conclusion

In this paper we propose a perspective-aware convolutional neural network to automatically estimate the crowd counts in images. A novel way of generating GT perspective maps is introduced for PACNN training, such that at the test stage it predicts both the perspective map and the density map. The perspective map is encoded as a perspective-aware weighting layer to adaptively combine the multi-scale density outputs. The combined density map is demonstrated to be robust to the perspective changes in crowd images. Extensive experiments

ShanghaiTech	$\overline{\mathrm{MAE_P}}$	$\overline{\mathrm{RMSE_P}}$	$\overline{\mathrm{PSNR_P}}$
PartA	0.092	0.120	18.06
PartB	0.116	0.149	16.83

TABLE IV: Perspective estimations using PACNN. $\overline{\mathrm{MAE_P}}$, $\overline{\mathrm{RMSE_P}}$ and $\overline{\mathrm{PSNR_P}}$ are the average values of $\mathrm{MAE_P}$, $\mathrm{RMSE_P}$, and $\mathrm{PSNR_P}$ over the dataset.

Examples in Fig. 4	$\mathrm{MAE_P}$	$\mathrm{RMSE_P}$	$\mathrm{PSNR_P}$
top-left	0.093	0.129	17.80
top-right	0.075	0.089	20.50
bottom-left	0.090	0.121	18.31
bottom-right	0.126	0.152	16.40

TABLE V: Evaluations of Examples in Fig. 4. $\mathrm{MAE_P}$, $\mathrm{RMSE_P}$ and $\mathrm{PSNR_P}$ are reported.

on standard crowd counting benchmarks show the efficiency and effectiveness of the proposed method over the state-of-the-art. Possible direction for future work could be using the perspective map to directly combine more than two density outputs. We expect it to be more complex but the improvement might also be worthwhile.

APPENDIX

This appendix provides the experimental results for perspective estimations. We evaluate perspective estimations on ShanghaiTech dataset [13]. We follow the same implementation details as in Sec. IV-B. We first present the evaluation protocol and then offer the results.

Evaluation Protocol. The perspective maps produced by PACNN are of $\frac{1}{8}$ resolution of the ground truth maps. To compare with the ground truth, we downsample the ground truth maps to have the same resolution with the estimated maps. We normalize both the estimated and ground truth perspective values within each map for comparison.

For each estimated perspective map P^e, we employ three measurements to evaluate its similarity to the ground truth P^g: mean absolute perspective error ($\mathrm{MAE_P}$), root mean square perspective error ($\mathrm{RMSE_P}$), and peak perspective signal to noise ratio ($\mathrm{PSNR_P}$),

$$\begin{aligned} \mathrm{MAE_P} &= \frac{1}{S}\sum_{j=1}^{S}|p_j^e - p_j^g|, \\ \mathrm{RMSE_P} &= \sqrt{\frac{1}{S}\sum_{j=1}^{S}(p_j^e - p_j^g)^2}, \\ \mathrm{PSNR_P} &= 20\log_{10}(\frac{\mathrm{MAX_P}}{\mathrm{RMSE_P}}) \end{aligned} \tag{14}$$

where p_j^e and p_j^g denote the normalized estimated and ground truth perspective values at pixel j, respectively. We use S to denote the total number of pixels in each map. $\mathrm{MAX_P}$ is the maximum possible pixel value of the perspective map P. Since both P^e and P^g are normalized, $\mathrm{MAX_P} = 1$ in practice. We average the $\mathrm{MAE_P}$, $\mathrm{RMSE_P}$ and $\mathrm{PSNR_P}$ over the entire set, and denote by $\overline{\mathrm{MAE_P}}$, $\overline{\mathrm{RMSE_P}}$ and $\overline{\mathrm{PSNR_P}}$ our final measurements. Small $\overline{\mathrm{MAE_P}}$ and $\overline{\mathrm{RMSE_P}}$ and big $\overline{\mathrm{PSNR_P}}$ indicate good performance.

Results on ShanghaiTech. We show the perspective map estimation results in Table IV on ShanghaiTech PartA and PartB, respectively. $\overline{\mathrm{MAE_P}}$, $\overline{\mathrm{RMSE_P}}$ and $\overline{\mathrm{PSNR_P}}$ are similar between PartA and PartB. Their $\overline{\mathrm{PSNR_P}}$ are 18.06, and 16.83, respectively. We illustrate some examples in Fig. 4 and compute their $\mathrm{MAE_P}$, $\mathrm{RMSE_P}$ and $\mathrm{PSNR_P}$, respectively in Table V. The estimated perspective map in general keeps the color gradient as in the corresponding ground truth, albeit the noise. In Table. I we show the crowd counting accuracy using the estimated perspective maps is in fact very close to that using the ground truth perspective maps.

REFERENCES

[1] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015.

[2] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *ICCV*, 2005.

[3] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *IJCV*, vol. 63, no. 2, pp. 153–161, 2003.

[4] G. J. Brostow and R. Cipolla, "Unsupervised bayesian detection of independent motion in crowds," in *CVPR*, 2006.

[5] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *CVPR*, 2006.

[6] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *CVPR*, 2008.

[7] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting." in *BMVC*, 2012.

[8] D. Ryan, S. Denman, C. Fookes, and S. Sridharan, "Crowd counting using multiple local features," in *DICTA*, 2009.

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

[10] D. Kong, D. Gray, and H. Tao, "Counting pedestrians in crowds using viewpoint invariant training." in *BMVC*, 2005.

[11] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *NIPS*, 2010.

[12] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *ECCV*, 2016.

[13] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *CVPR*, 2016.

[14] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," in *CVPR*, 2017.

[15] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[16] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *ECCV*, 2016.

[17] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and bayesian regression," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2160–2177, 2012.

[18] N. C. Tang, Y.-Y. Lin, M.-F. Weng, and H.-Y. M. Liao, "Cross-camera knowledge transfer for multiview people counting," *IEEE Transactions on image processing*, vol. 24, no. 1, pp. 80–93, 2015.

[19] S. Huang, X. Li, Z. Zhang, F. Wu, S. Gao, R. Ji, and J. Han, "Body structure aware deep crowd counting," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1049–1059, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[21] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *ECCV*, 2016.

[22] V. A. Sindagi and V. M. Patel, "Generating high-quality crowd density maps using contextual pyramid cnns," in *ICCV*, 2017.

[23] ——, "Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in *AVSS*, 2017.

[24] F. Xiong, X. Shi, and D.-Y. Yeung, "Spatiotemporal modeling for crowd counting in videos," in *ICCV*, 2017.

[25] J. Liu, C. Gao, D. Meng, and A. G. Hauptmann, "Decidenet: Counting varying density crowds through attention guided detection and density estimation," in *CVPR*, 2018.

[26] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *CVPR*, 2012.

[27] M. Wang and X. Wang, "Automatic adaptation of a generic pedestrian detector to a specific traffic scene," in *CVPR*, 2011.

[28] R. Stewart, M. Andriluka, and A. Y. Ng, "End-to-end people detection in crowded scenes," in *CVPR*, 2016.

[29] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[31] S.-F. Lin, J.-Y. Chen, and H.-X. Chao, "Estimation of number of people in crowded scenes using perspective transformation," *TSMC-A*, vol. 31, no. 6, pp. 645–654, 2001.

[32] C. S. Regazzoni and A. Tesei, "Distributed data fusion for real-time crowding estimation," *Signal Processing*, vol. 53, no. 1, pp. 47–63, 1996.

[33] A. Marana, L. d. F. Costa, R. Lotufo, and S. Velastin, "On the efficacy of texture analysis for crowd monitoring," in *SIBGRAPI*, 1998.

[34] N. Paragios and V. Ramesh, "A mrf-based approach for real-time subway monitoring," in *CVPR*, 2001.

[35] M. Fu, P. Xu, X. Li, Q. Liu, M. Ye, and C. Zhu, "Fast crowd density estimation with convolutional neural networks," *Engineering Applications of Artificial Intelligence*, vol. 43, pp. 81–88, 2015.

[36] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, "Deep people counting in extremely dense crowds," in *ACM MM*, 2015.

[37] L. Boominathan, S. S. Kruthiventi, and R. V. Babu, "Crowdnet: a deep convolutional network for dense crowd counting," in *ACM MM*, 2016.

[38] Z. Zhao, H. Li, R. Zhao, and X. Wang, "Crossing-line crowd counting with two-phase deep neural networks," in *ECCV*, 2016.

[39] S. Kumagai, K. Hotta, and T. Kurita, "Mixture of counting cnns: Adaptive integration of cnns specialized to specific appearance for crowd counting," *arXiv:1703.09393*, 2017.

[40] M. Marsden, K. McGuiness, S. Little, and N. E. O'Connor, "Fully convolutional crowd counting on highly congested scenes," *VISAPP*, 2017.

[41] Z. Lu, M. Shi, and Q. Chen, "Crowd counting via scale-adaptive convolutional neural network," in *WACV*, 2018.

[42] L. Zeng, X. Xu, B. Cai, S. Qiu, and T. Zhang, "Multi-scale convolutional neural networks for crowd counting," *arXiv:1702.02359*, 2017.

[43] Z. Wei, Y. Sun, J. Wang, H. Lai, and S. Liu, "Learning adaptive receptive fields for deep image parsing network," in *CVPR*, 2017.

[44] R. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan, "Scale-adaptive convolutions for scene parsing," in *ICCV*, 2017.

[45] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[46] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3–15, 2008.

[47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[48] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015.

[49] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM MM*, 2014.