# Kernel-based Density Map Generation for Dense Object Counting

Jia Wan, Qingzhong Wang, and Antoni B. Chan

**Abstract**—Crowd counting is an essential topic in computer vision due to its practical usage in surveillance systems. The typical design of crowd counting algorithms is divided into two steps. First, the ground-truth density maps of crowd images are generated from the ground-truth dot maps (density map generation), e.g., by convolving with a Gaussian kernel. Second, deep learning models are designed to predict a density map from an input image (density map estimation). The density map based counting methods that incorporate density map as the intermediate representation have improved counting performance dramatically. However, in the sense of end-to-end training, the hand-crafted methods used for generating the density maps may not be optimal for the particular network or dataset used. To address this issue, we propose an adaptive density map generator, which takes the annotation dot map as input, and learns a density map representation for a counter. The counter and generator are trained jointly within an end-to-end framework. We also show that the proposed framework can be applied to general dense object counting tasks. Extensive experiments are conducted on 10 datasets for 3 applications: crowd counting, vehicle counting, and general object counting. The experiment results on these datasets confirm the effectiveness of the proposed learnable density map representations.

**Index Terms**—Crowd counting, vehicle counting, object counting, density map generation, density map estimation, deep learning

## 1 Introduction

Dense object counting tasks, including crowd counting, vehicle counting, and general object counting, aim to estimate the number of objects in the image. Counting tasks have practical usage for understanding crowded scenes. For example, crowd counting can be used to prevent accidents caused by overcrowding and estimate the crowd flows in station. And vehicle counting can be used for traffic management on roads or in parking lots. General object counting is useful for the management of goods in the supermarket, farms and factories.

Although counting tasks are important and useful, the real usage is still limited since dense object counting is challenging. One of the main challenges is scale variation. Since the scale of people varies dramatically in images and across different images, it is difficult to extract features for density regression. Another challenge is the occlusions among people since only a small part of each person may be visible in crowd images. Complex backgrounds may also hurt the counting performance, and the domain gap between scenes in datasets and the real world scenes also limits the usage of counting algorithms.

Current state-of-the-art methods use crowd density maps to achieve superior counting performance [1, 2, 3]. Density maps are an *intermediate* representation, where the sum over any region in the density map indicates the number of people in the region. First, the density maps are generated from the dot annotations, where each dot indicates a person's location. Second, given the input image, algorithms are designed to predict the density map (see Figure 1), which is then summed to obtain the count. In this paper, we call these two steps density map *generation* and density map *estimation*, respectively. Most works focus on density map estimation and ignore density map generation. Many different deep networks have been proposed to improve density map estimation, e.g., using different kernel sizes [4] or image pyramids [5] to handle scale variations, or using context [6] or prior information [7] to handle occlusions. Although density map estimation is well-studied, the generation of density maps is often overlooked and uses handcrafted designs without adequate investigation and analysis. The simplest approach to obtain a density map is to convolve the annotation dot map with a Gaussian with fixed width [8], i.e., place a Gaussian on each dot. Other works scale the Gaussian bandwidth according to the scene perspective [9], or adaptively use the local congestion level (or distance to nearest neighbors) [4]. [9] uses human-shaped kernels, composed of two Gaussians, but is less popular since the body of the person is often occluded in crowd images.

In practice, the method for generating the density maps is crucial for crowd counting. Improperly generated density maps may dramatically hurt the counting performance – the choice of the kernel bandwidth or kernel shape used to generate the density map is often dataset dependent, and such choices often do not work across different datasets. In the era of deep learning, we may consider current density maps as a *hand-crafted intermediate representation*, which is used as a target for training deep networks to count. From the standpoint of end-to-end training, these hand-designed intermediate representations may not be optimal for the particular network architecture and particular dataset.

In this paper, we take the first step towards learnable density map representations, which are jointly trained with the density map estimator (counter). In particular, we first generate a unique normalized kernel for each object (e.g. a person or a vehicle) given an image as the input. The scale and shape of kernels are automatically learned during the joint optimization with a counter. The proposed method can potentially produce adaptive density

- *Jia Wan, Qingzhong Wang, and Antoni B. Chan (corresponding author) are with the Department of Computer Science, City University of Hong Kong.*
*E-mail: jiawan1998@gmail.com, qingzwang2-c@my.cityu.edu.hk, abchan@cityu.edu.hk.*

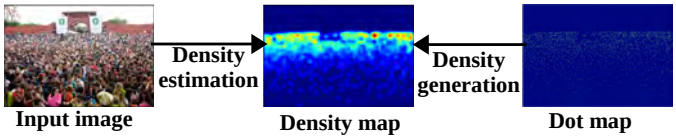


Fig. 1. Counting by crowd density maps: the ground-truth density map is generated from a dot annotation map. Then, algorithms are designed to predict the density map, which is summed to obtain the predicted global count. Current approaches treat the density map as a fixed intermediate representation, which is based on hand-crafted. The proposed algorithm jointly learns the density map generator and density map estimator.

maps for different counters, datasets and objects.

In summary, the contributions of this paper are:

1) We study the impact of density maps on different datasets, and confirm through experiments that proper selection of density maps is essential for counting.
2) To improve manually-generated density maps, we propose to refine traditional density maps and achieve superior performance, which confirms the intermediate representation of density maps can be improved.
3) We propose an adaptive density map generator, which takes the dot map as input, and produces a learnable density map representation. The density map generator and density map estimator (counter) are jointly trained.
4) We extend density map generation framework to general object counting problem. A novel generation method which can generate unique density kernels for individual objects is proposed. The proposed method is easy to analyze since we can visualize each individual kernel.
5) With the proposed framework, we achieve state-of-the-art performance on 10 datasets for 3 applications: crowd counting, vehicle counting, and general object counting without modifying the structure of the counter.

The remainder of this paper is organized as follows. The related works of crowd counting algorithms and density map generation methods are briefly reviewed in Section 2. The details of the proposed methods are introduced in Section 3, followed by experimental results and analysis in Section 4. The conclusion is presented in Section 5.

## 2 Related Works

We first briefly review related crowd counting algorithms, and then introduce the typical density map generation methods.

### 2.1 Crowd Counting

Traditional crowd counting algorithms are based on individual detection and tracking [10, 11], but these methods do not work well for dense scenes. Thus, global regression based methods are proposed to avoid the detection of individuals by directly regressing the number of people [12, 13, 14]. However, global regression ignores the spatial distribution of people, and thus density map based methods are proposed to further predict the spatial density map in images [8] and have achieve the outstanding performance [15].

#### 2.1.1 Detection-based counting

Individual detection and tracking based counting algorithms rely on the detection algorithms that do not work well under congested scenes. Most of these algorithms detect human heads and shoulders by detection or tracking algorithms. Li *et al.* [10] propose to count the number of people by detecting human heads and shoulders based on foreground segmentation. A shape based detection algorithm is proposed to detect individuals in [11]. Although the performance of detection algorithms have been improved significantly, the detection of individuals under very dense scenes is still challenging.

#### 2.1.2 Regression-based counting

To avoid explicit detection of individuals, regression methods are proposed to estimate the number of people directly from low-level features like texture, color, and gradient. Chan *et al.* [12] propose to count by directly regressing from global features to crowd number using Gaussian process regression. A prior distribution is proposed in [13] to estimate homogeneous crowds. Multiple features are combined in [14] to improve the performance of crowd counting. However, the performance of traditional counting algorithms are still limited due to scale variation and occlusion in crowd images.

#### 2.1.3 Density map based counting

Density map based methods are currently the most popular approach to crowd counting since the performance can be dramatically improved by utilizing spatial information. Density maps are typically generated by blurring dot maps in which each dot indicates a person in an image [8]. Since density maps are intermediate representations, most algorithms generate density maps beforehand by convolving the dot maps with Gaussian kernels with either fixed or adaptive bandwidths. Then, different network architectures are designed to handle various challenges, such as scale changes, improving the quality of density maps, encoding more contextual information, or adapting to new scenarios.

**Scale variation:** To handle with scale variation of people, [4] proposes a multi-column neural network (MCNN) where each column has different kernel sizes to extract multi-scale features. Similarly, SANet [16] proposes to extract multi-scale features with scale aggregation modules, while Kang and Chan [5] propose to use image pyramid to deal with scale variation in crowd counting. Instead of fusing multi-scale features, switch-CNN [17] proposes to select a proper column with appropriate receptive field. A tree-structured CNN is proposed to handle the diversity of people in crowds [18]. [19] proposes a hierarchical encoder-decoder framework to encode multi-scale features, while [20] proposes an attention based framework to filter background and [21] proposes a novel feature fusion strategy.

**Refinement-based:** Another way to improve the quality of density maps and the performance of crowd counting is to use refinement-based algorithms. Ranjan *et al.* [22] propose a two stage counting framework in which the high-resolution density map is estimated based on the low-resolution density map generated in the initial stage, while Sam and Babu [23] propose a feedback mechanism to refine the predicted density map. Besides image-based refinement framework, [24] proposes a region-based refinement method. Related to refinement based methods are ensemble methods, such as [25], which uses CNN boosting, or [26], which uses multiple negative correlated regressors.

**Context:** Contextual information is also useful for crowd counting. Sindagi and Patel [6] propose a contextual pyramid CNN, while Xiong *et al.* [27] propose to utilize temporal contextual information. To exploit unlabelled data, [7] propose a ranking based method. Other works have also shown that density maps

are useful for pedestrian detection and tracking in crowd scenes [28]. [29] proposes to localize individuals and estimate the density map simultaneously. [30] proposes a Context-Aware network by exploiting the difference between local features and global features, while [31] investigates the effectiveness of perspective map for crowd counting. [32] proposes to count crowd people by multi-camera settings.

**Multi-task:** Multi-task learning can improve counting performance through better learning of features. Idrees *et al.* [33] propose a composition loss for crowd counting, density map estimation and localization tasks. [34] proposes to deal with counting and localization using RGB-D data. [35] proposes to improve counting performance in cooperating with auxiliary tasks.

**Cross-scene:** One challenge with applying counting algorithms in real application is the domain gap between the training datasets and the new scene, i.e., the cross-scene problem. [9] proposes a cross-scene algorithm to adapt the trained model to new scenes. Wang *et al.* [36] propose a synthetic dataset and a domain adaptation algorithm to boost counting performance with the proposed dataset. More related works can be found in [37].

These density estimation methods are trained directly using hand-crafted density maps as supervision. Our proposed density map generator can be jointly trained with any of these density estimation methods in order to tune the density maps to the specific capabilities of the estimator.

### 2.2 Density Map Generation

Previous crowd counting research mainly concentrates on density map estimation but ignores density map generation. Current generation approaches convolve the dot map with a Gaussian kernel with fixed bandwidth [8] or an adaptive bandwidth based on crowdedness [4] or scene perspective [9]. However, the hyperparameters are manually selected without full investigation. In contrast to these hand-crafted methods, our proposed approach jointly learns the density map generator with the counting algorithm in an end-to-end manner.

A preliminary version of this work appears in [38]. The extensions over the conference version are three-fold. First, we propose a new generation method that can generate unique density kernels for individual objects, and outperforms our previous methods in [38] that are based on refining or fusing density maps based on Gaussian kernels. The new method is easy to analyze since it is possible to visualize each individual's kernel. Second, we apply our proposed methods to general object counting tasks, such as vehicle counting and general object counting. The new extensive experiment results show that the proposed method is effective at the dense object counting tasks. Third, analyzing the learned kernels, we discover that the kernels are sometimes flatter compared to traditional Gaussian kernels, which provides insight into better design of density map generation methods.

Finally, we note that some methods avoid using density maps altogether; [39] proposes a detection based framework without the use of density map, while [15] proposes a novel loss function based on point annotation.

## 3 Density Map Generators

In this section, we first briefly introduce traditional generation methods, and then present our density map generation framework which learns to make density maps from dot annotation maps as inputs. We consider three methods for density map generation: 1) density map refinement, which modifies a traditional density map; 2) adaptive density map generation, which adaptively fuses together traditional density maps with different kernel bandwidths; 3) kernel-based density map generation, which learns arbitrary kernels for each annotation and is suitable for general object counting.

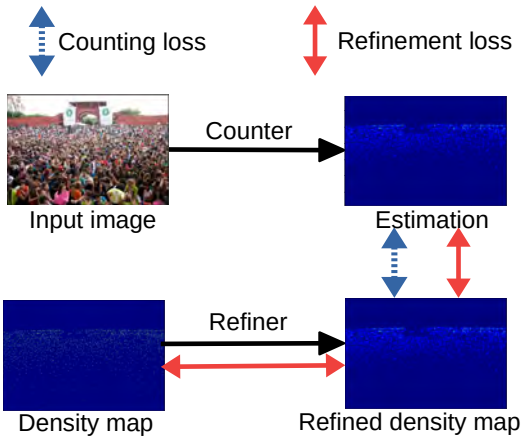


Fig. 2. Density map refinement framework. The *Counter* is a network that estimates the density map of an input image. The *Refiner* is another network that takes a density map as input and produces a better density map as the ground truth to train the Counter. Both the Counter and Refiner are trained jointly.

### 3.1 Traditional Density Map Generation

Traditional density map generation approaches treat generation and estimation as two separate steps. Usually, a dot annotation map (the input) is convolved with a Gaussian kernel to form a smooth heat map which is called a density map.

Formally, given a dot map $D$, where the spatial location of each annotated person takes the value 1, (and 0 otherwise), the density map $Y$ is the convolution of $D$ with a Gaussian kernel,

$$Y = D * k_\sigma, \tag{1}$$

where $k_\sigma$ is a 2D Gaussian kernel with bandwidth $\sigma$, and $*$ is 2D convolution. This is equivalent to placing a Gaussian on each dot annotation,

$$Y(p) = \sum_{\{p'|D(p')=1\}} \mathcal{N}(p|p', \sigma^2 I), \tag{2}$$

where $p$ is a pixel location in the image, $p'$ is the location of an annotated person, and $\mathcal{N}(p|\mu, \Sigma)$ is a multivariate Gaussian with mean $\mu$ and covariance $\Sigma$. For adaptive kernels, $\sigma$ changes with pixel location based on the perspective [4] or crowdedness [4]. The counter is then trained with images and the corresponding density maps.

### 3.2 Density Map Refinement

To confirm that traditional density maps can be improved to produce better counting performance, we first propose a density map refinement framework that jointly refines the density map and trains a counter from the refined density map (see Figure 2).

Formally, let $(X_i, Y_i)$ be the i-th image and traditional density map pair. We denote $f(X_i)$ as the predicted density map for image $X_i$, and $g(Y_i)$ as the refined density map for $Y_i$. The refiner $g$ is

**Algorithm 1** Training using density map refinement

**Input**: Set of image and density map pairs $\{(X_i, Y_i)\}_{i=1}^{N}$.
Initialize parameters of counter $f$ and refiner $g$.
**for** $epoch = \{1, \ldots, N_e\}$ **do**
  **for** $i = \{1, \ldots, N\}$ **do**
    Estimate density map $f(X_i)$ by the counter.
    Produce refined ground truth $g(Y_i)$ by the refiner.
    Update counter $f$ using the counting loss in (3).
    Update refiner $g$ using the refinement loss in (3).
  **end for**
**end for**
**Output:** a counter and a refiner.

a fully convolutional network. The counter $f$ and refiner $g$ are jointly trained using a combined loss,

$$\mathcal{L} = \overbrace{\underbrace{\sum_{i=1}^{N} \|f(X_i) - g(Y_i)\|^2}_{\text{refinement loss}} + \alpha \|g(Y_i) - Y_i\|^2}^{\text{counting loss}}, \quad (3)$$

where $N$ is the number of training pairs. The first term in (3) trains the counter to predict the refined density map, and vice versa, trains the refiner to produce density maps that favor the counter's architecture. The second term in (3) constrains the refined density map to be close to the original density map $Y_i$, so that the global count and spatial distribution of the crowd are preserved. In another interpretation, $g(\cdot)$ is trained like an auto-encoder, where the 2nd term is the reconstruction loss and the 1st term is a task loss that allows the "reconstruction" $g(Y_i)$ to deviate in order to better match the predicted density map $f(X_i)$. The joint training is summarized in Algorithm 1. Note that the refiner is only needed for training, i.e., to find the optimal intermediate representation – at inference time, only the counter is used to predict the density map from a novel image.

### 3.3 Adaptive Density Map Generation

The method in the previous section refines an existing density map. We next consider how to generate a density map directly from the dot annotation map. With this density map generation framework, the whole system can be end-to-end trained without any intermediate density maps (see Figure 3). This approach adaptively fuses together density maps constructed from different kernels, and then refines it to generate a new density map.

#### 3.3.1 Generation via Self-attention and Fusion

Given a dot map as the input, the generation of a density map is divided into 3 steps: Gaussian blurring, self-attention, and fusion. First, the input dot map $D_i$ is convolved with $k$ Gaussian kernels with different bandwidths, resulting in a stack of $k$ blurred density maps $B_i = \{B_i^j\}_j$,

$$B_i^j = D_i * k_{\sigma_j}, \quad (4)$$

which is equivalent to a convolutional layer with a different Gaussian kernel for each filter channel. Second, a self-attention module uses the blurred maps $B_i$ as input to select the best kernel size for each region, based on the density of people in the region. For example, this gives the generator flexibility to focus on the density maps using small-sized kernels for small people (in the background), and those using large-sized kernels for large people (in the foreground). In particular, the attention map is

$$A_i = F_a(B_i), \quad (5)$$

where $F_a$ is a small convolutional network, and each channel of $A_i$ is an attention map for the corresponding blurred density map. Third, the blurred density maps are adaptively fused based on the attention maps,

$$M_i = F_f(A_i \otimes B_i), \quad (6)$$

where $\otimes$ is the pixel-wise multiplication, $M_i$ is the final learned density map that is used to supervise the counter, and $F_f$ is the fusion network.

**Algorithm 2** Training using adaptive density map generation

**Input**: Set of image and dot map pairs $\{(X_i, D_i)\}_{i=1}^{N}$.
Initialize parameters of counter and refiner.
**for** $epoch = \{1, \ldots, N_e\}$ **do**
  **for** $i = \{1, \ldots, N\}$ **do**
    Estimate density map $\hat{M}_i$ by the counter.
    Produce ground truth $M_i$ by the generator.
    Update the counter using the counting loss in (7).
    {update generator every $N_g$ epochs.}
    **if** $\mod (epoch, N_g) = 0$ **then**
      Update parameters of generator using (7).
    **end if**
  **end for**
**end for**
**Output:** a counter and a generator.

#### 3.3.2 Loss Functions

Given a training set of images and corresponding ground-truth dot maps $\{(X_i, D_i)\}_{i=1}^{N}$, the density map generator and counter are jointly trained using the loss function,

$$\mathcal{L} = \overbrace{\underbrace{\sum_{i=1}^{N} \|\hat{M}_i - M_i\|^2}_{\text{generation loss}} + \beta (1^T M_i - 1^T D_i)^2}^{\text{counting loss}} \quad (7)$$

where $\hat{M}_i$ is the density map predicted by the counter, $M_i$ is the generated density map, and $1^T M$ is the sum over entries in $M$, i.e. the count from $M$. The first term in (7) trains the counter to predict the generated density map, while also training the generator to produce density maps that the counter can predict well. The second term in (7) encourages that the generated density maps have counts that are close to the ground-truth count. Algorithm 2 summarizes the training procedure for the counter and generator.

### 3.4 Kernel-based Density Map Generation

The adaptive generation method presented in the previous subsection has three disadvantages:

1) The summation of the generated density map is not always equal to the people count since the fusion network may cause the sum to deviate.
2) Gaussian kernels are not suitable for density map generation, especially for objects like vehicles since the shape of vehicles is not match with Gaussian kernels.
3) It is difficult to analyze the kernel shape of individuals since the density map is generated directly.

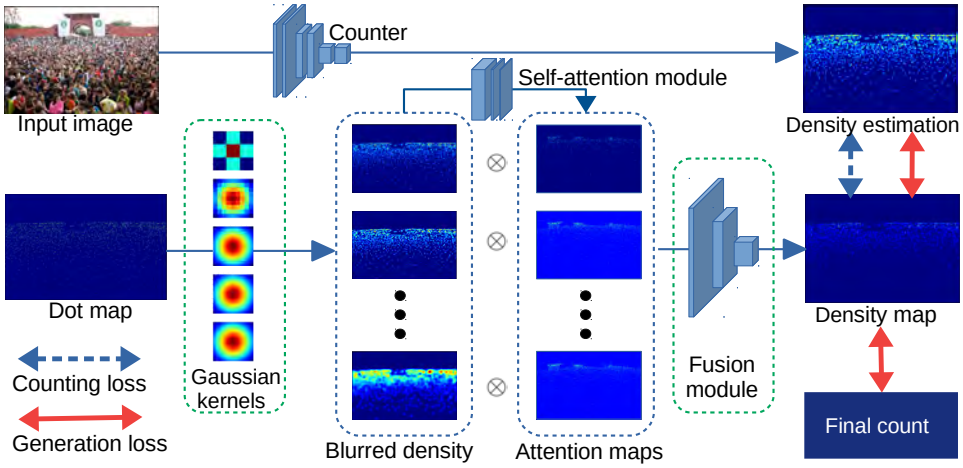


Fig. 3. Adaptive density map generation framework. The input dot map is convolved with different Gaussian kernels, yielding a set of blurred density maps. The blurred density maps are adaptively masked using a self-attention module, and then passed through a fusion module to produce the final density map. The generated density map serves as the ground truth for training the density map estimator (counter).

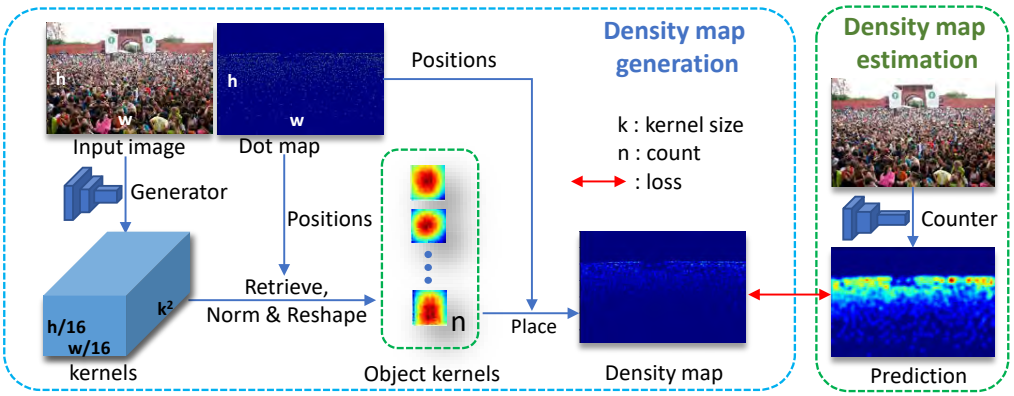


Fig. 4. Kernel-based density map generation framework. Given an image as the input, we first learn density kernels for all spatial locations (a $w \times h \times k^2$ tensor). Then, object kernels are retrieved, normalized and reshaped based on their coordinates. The density map is generated by sticking all the kernels together based on their locations. The generated density map then serves as the ground-truth to train a counting network.

We propose a novel generation framework to overcome these problems. In particular, we generate density kernels for each object and form the density map based on the positions of these objects. The kernels are always normalized (sum to 1) so that the summation of the learned density map is always equal to the number of objects. Furthermore, the kernel shapes are easy to analyze since they are generated explicitly for each object. Finally, the shape of kernels can adapt to different objects so the proposed framework can naturally apply to general object counting tasks.

The pipeline of the proposed method is shown in Figure 4. Given an input image $X$[1], kernels are learned for each spatial position by a generator, resulting in a $w \times h \times k^2$ kernel map $K$, where $w \times h$ is the image size and $k$ is the kernel width,

$$K = F_k(X), \tag{8}$$

where $F_k$ is the *Kernel Generator* which is modeled by a neural network. The output of the Kernel Generator is a set of predicted kernels $K$, which is a $w \times h \times k^2$ tensor. For each spatial location $p = (x, y)$, the vector $K_p$ in $K$ is a $k^2$ vector that represents a $k \times k$ kernel. Let $\mathcal{D} = \{p_j\}_{j=1}^{A}$ be the set of $A$ annotated 2D coordinates of the person/object, $p_j = (x_j, y_j)$, in image $X$. For each annotation in $\mathcal{D}$, we retrieve the corresponding kernel $K_{p_j}$ from $K$. We then reshape and normalize $K_{p_j}$ to sum to 1, resulting in the location-specific $k \times k$ kernel $\tilde{K}_{p_j} = \text{vec}^{-1}(K_{p_j})/\text{sum}(K_{p_j})$, where $\text{vec}^{-1}$ is the vector-to-matrix reshaping from $k^2$-dim vector to $k \times k$ matrix, and $\text{sum}(z)$ is the sum of the entries in the vector $z$. Finally, the generated density map is obtained by placing the location-specific kernels on top of each annotation,

$$M(p) = \sum_{j=1}^{A} \tilde{K}_{p_j}(p - p_j), \tag{9}$$

where the indexing of $\tilde{K}_{p_j}(p)$ is on $p \in \{-r, \cdots, r\} \times \{-r, \cdots, r\}$ where $r = (k-1)/2$. Note that (9) is analogous to placing Gaussians on each annotation to generate a traditional density map, except that now the kernels are learned, and could be different for each position and each image. The learned density map $M$ is then used as the ground-truth to train a counter.

### 3.4.1 Loss Functions

Given a training set of images and corresponding ground-truth dot maps $\{(X_i, D_i)\}_{i=1}^{N}$, the density map generator and counter are

1. Here we do not explicitly write the dependence on $i$ to reduce clutter.

**Algorithm 3** Training using kernel-based density map generation

1: **Input**: Set of image and dot map pairs $\{(X_i, D_i)\}_{i=1}^{N}$.
2: Initialize parameters of counter and refiner.
3: **for** $epoch = \{1, \ldots, N_e\}$ **do**
4:  **for** $i = \{1, \ldots, N\}$ **do**
5:  Estimate density map $\hat{M}_i$ by the counter.
6:  Produce ground truth $M_i$ by the generator.
7:  Update the counter using the counting loss in (10).
8:  Update parameters of generator using (10).
9:  **end for**
10: **end for**
11: **Output:** a counter and a generator.

jointly trained using the loss function,

$$\mathcal{L} = \overbrace{\sum_{i=1}^{N} \|\hat{M}_i - M_i\|^2 + \lambda(1 - CS(\hat{M}_i^{'}, M_i^{'}))}^{\text{generation loss}}, \qquad (10)$$

counting loss

where $\hat{M}_i$ is the density map predicted by the counter, and $M_i$ is the generated density map. $\hat{M}_i^{'}$ and $M_i^{'}$ are vectorized versions of $\hat{M}_i$ and $M_i$. $CS(\hat{M}_i^{'}, M_i^{'})$ is the cosine similarity between two vectors,

$$CS(\hat{M}_i^{'}, M_i^{'}) = \frac{\hat{M}_i^{'} \cdot M_i^{'}}{\max(\|\hat{M}_i^{'}\|_2 \cdot \|M_i^{'}\|_2, \epsilon)}, \qquad (11)$$

where $\epsilon = 10^{-8}$. We use the cosine similarity for spatial regularization of the density map. The normalized vectors $\hat{M}_i^{'}/||\hat{M}_i^{'}||_2$ and $M_i^{'}/||M_i^{'}||_2$ represents the spatial distribution of the crowd regardless of the count. Thus, the cosine similarity in (11) encourages similar spatial distributions of crowd and non-crowd between the GT and predicted density maps. Algorithm 3 summarizes the training procedure for the counter and generator. The generator and dot maps are only used to train the counter. At test time, the counter predicts the density map from the input image.

## 4 Experiments

In this section we present experiments evaluating our proposed density map generation frameworks. We first present detailed experimental setup including applications, datasets and evaluate metrics. Then, we compare the proposed framework with state-of-the-art approaches. Finally, we investigate hyper-parameter settings and analyze the proposed method through visualization of individual kernels and density maps.

TABLE 2
The architectures of the density map refiner, adaptive density map generation, and kernel-based density map generation. C(K,S) is a convolution layer with $K$ features and kernel size $S$. P is average pooling that decreases the spatial size by half, $k$ is the size of the generated kernels. Each conv layer is followed by a ReLU, except the last layer.

| Subnetwork | Architecture |
|---|---|
| Refiner | C(512,3)-C(512,3)-C(256,3)-C(128,3)-C(64,3)-C(1,3) |
| Self-attention | C(128,3)-C(32,3)-C(5,3)-Softmax |
| Fusion | C(128, 3)-P-C(32, 3)-P-C(8, 3)-P-C(1,3)-PReLU[2] |
| Kernel Generator | C(32, 3)-P-C(64, 3)-P-C(128, 3)-P-C(128,3)-P-C($k^2$,3) |

2. In practice, we find that occasionally the output of the Fusion network is always 0 and the network cannot be optimized properly if we use ReLU as the final activation. Therefore, we change the last layer activation to PReLU to mitigate this problem.

TABLE 3
Experiment results on ShanghaiTech. MAE and RMSE are used to evaluate the performance.

| Method | ShanghaiTech_A | | ShanghaiTech_B | |
|---|---|---|---|---|
| | MAE ↓ | MSE ↓ | MAE ↓ | MSE ↓ |
| Cross-scene [9] | 181.8 | 277.7 | 32.0 | 49.8 |
| MCNN [4] | 110.2 | 173.2 | 26.4 | 41.3 |
| FCN [46] | 126.5 | 173.5 | 23.8 | 33.1 |
| Cascaded-MTL [47] | 101.3 | 152.4 | 20.0 | 31.1 |
| Switching-CNN [17] | 90.4 | 135.0 | 21.6 | 33.4 |
| CP-CNN [6] | 73.6 | 106.4 | 20.1 | 30.1 |
| ASACP [48] | 75.7 | 102.7 | 17.2 | 27.4 |
| Top-Down [23] | 97.5 | 145.1 | 20.7 | 32.8 |
| L2R [7] | 73.6 | 112.0 | 14.4 | 23.8 |
| IG-CNN [22] | 72.5 | 118.2 | 13.6 | 21.1 |
| ic-CNN [22] | 68.9 | 117.3 | 10.7 | 16.0 |
| SANet [16] | 67.0 | 104.5 | 8.4 | <u>13.6</u> |
| SCNet [49] | 71.9 | 117.9 | 9.3 | 14.4 |
| Spatial-Aware [24] | 69.3 | **96.4** | 11.1 | 18.2 |
| Image Pyramid [5] | 80.6 | 126.7 | 10.2 | 18.3 |
| CSRNet† [50] | 68.2 | 115.0 | 10.6 | 16.0 |
| CSRNet | 67.8 | 109.3 | 9.5 | 15.9 |
| DMR (ours) [38] | <u>64.2</u> | 99.7 | 9.1 | 14.4 |
| ADMG (ours) | 64.7 | <u>97.1</u> | <u>8.1</u> | <u>13.6</u> |
| KDMG (ours) | **63.8** | 99.2 | **7.8** | **12.7** |

TABLE 4
Experiment results on UCF-QNRF.

| Method | MAE↓ | MSE↓ |
|---|---|---|
| Multi-sources [14] | 315 | 508 |
| MCNN [4] | 277 | 426 |
| Encoder-decoder [51] | 277 | 426 |
| Cascaded-MTL [47] | 252 | 514 |
| Switching-CNN [17] | 228 | 445 |
| Resnet101 [52] | 190 | 277 |
| Densenet201 [53] | 163 | 226 |
| Composition Loss [33] | 132 | 191 |
| SFCN†[36] | 115 | 192 |
| CSRNet | 148 | 234 |
| DMR (ours) [38] | 111 | 189 |
| ADMG (ours) | <u>101</u> | <u>176</u> |
| KDMG (ours) | **99.5** | **173** |

### 4.1 Applications & Datasets

We conduct experiments on three applications: crowd counting, vehicle counting, and general object counting. For **crowd counting**, five datasets are used for evaluation, including ShanghaiTech (ShTech) A and B [4], UCF-QNRF [33], JHU-CROWD++ [40], and NWPU-Crowd [41]. ShanghaiTech A contains 482 crowd images with crowd numbers varying from 33 to 3139, and ShanghaiTech B contains 716 high-resolution images with crowd numbers from 9 to 578. UCF-QNRF and JHU-CROWD++ are two large-scale datasets that contain 1,535 and 4,250 high resolution images with very large crowds. NWPU-Crowd is the largest benchmark and has 5,109 high-resolution images with large crowd variation. For **vehicle counting**, TRANCOS [42], PUCPR+ [43], and CAPRK [43] are used for evaluation. TRANCOS contains 1244 images in the traffic with vehicle varying from 9 to 107. PUCPR+ and CARPK are used to count parking cars. PUCPR+ has 125 images with vehicle number from 0 to 331, while CARPK has 1448 images with vehicle number from 1 to 188. We also evaluate the proposed framework on **general counting** tasks on

TABLE 1
Datasets used for evaluation.

| Applications | Datasets | Min. | Avg. | Max. | Total | # of Image | # of Class | Avg. Resolution |
|---|---|---|---|---|---|---|---|---|
| Crowd Counting | ShanghaiTech A [4] | 33 | 501 | 3,139 | 241,667 | 482 | 1 | 589×868 |
| | ShanghaiTech B [4] | 9 | 123 | 578 | 88,488 | 716 | 1 | 768×1024 |
| | UCF-QNRF [33] | 49 | 815 | 12,865 | 1,251,642 | 1,525 | 1 | 2013×2902 |
| | JHU-CROWD++ [40] | 0 | 262 | 7,286 | 1,114,785 | 4,250 | 1 | 1450×900 |
| | NWPU-Crowd [41] | 0 | 418 | 20,033 | 2,133,238 | 5,109 | 1 | 2311×3383 |
| Vehicle Counting | TRANCOS [42] | 9 | 38 | 107 | 46,734 | 1,244 | 1 | 480×640 |
| | PUCPR+ [43] | 0 | 135 | 331 | 16,915 | 125 | 1 | 720×1280 |
| | CARPK [43] | 1 | 62 | 188 | 89,772 | 1,448 | 1 | 720×1280 |
| Object Counting | SKU-110K [44] | 1 | 147 | 718 | 1,733,708 | 11,762 | 110,712 | 3212×2473 |
| | DOTA [45] | 10 | 57 | 618 | 39,450 | 690 | 6 | 1010×996 |

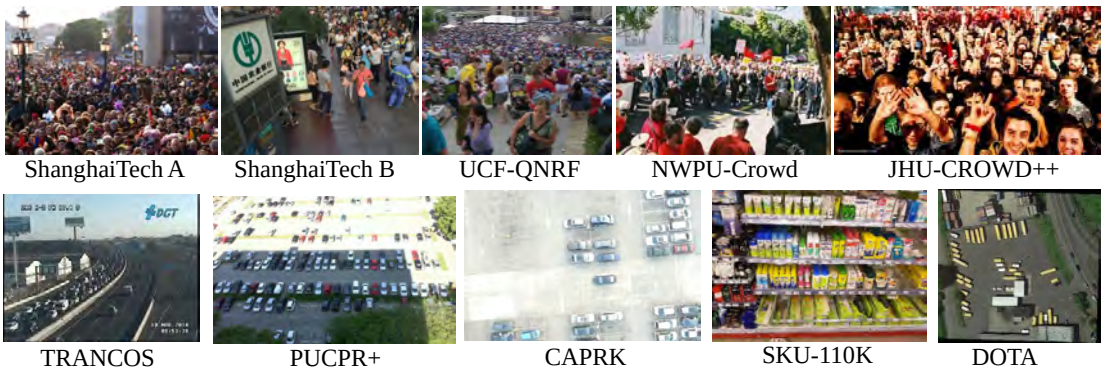


Fig. 5. Example images from the datasets.

TABLE 5
Experiment results on large-scale datasets NWPU-Crowd and JHU-CROWD++.

| | NWPU-Crowd | | JHU-CROWD++ | |
|---|---|---|---|---|
| | MAE | MSE | MAE | MSE |
| CMTL [47] | - | - | 157.8 | 490.4 |
| LSCCNN [54] | - | - | 112.7 | 454.4 |
| MCNN [4] | 232.5 | 714.6 | 188.9 | 483.4 |
| CSRNet [50] | 120.9 | 404.1 | 81.8 | 274.0 |
| SANet [16] | 190.6 | 491.4 | 91.1 | 320.4 |
| DSSINet [55] | - | - | 133.5 | 416.5 |
| MBTTBF [21] | - | - | 81.8 | 299.1 |
| BL [15] | 105.4 | 454.2 | 75.0 | 299.9 |
| ADMG (Ours) | 152.8 | 907.3 | 75.47 | 270.68 |
| KDMG (Ours) | **100.5** | 415.5 | **69.74** | **268.27** |

TABLE 6
Experiment results of vehicle counting on TRANCOS dataset.

| Methods | GAME(0) | GAME(1) | GAME(2) | GAME(3) |
|---|---|---|---|---|
| Victor *et al.* [56] | 13.76 | 16.72 | 20.72 | 24.36 |
| Onoro *et al.* [8] | 10.99 | 13.75 | 16.09 | 19.32 |
| CSRNet† [50] | 3.56 | 5.49 | 8.57 | 15.04 |
| CSRNet | 3.67 | 5.82 | 8.53 | 14.40 |
| PSDDN [39] | 4.79 | 5.43 | 6.68 | **8.40** |
| ADMG (ours) | 3.79 | 5.93 | 8.49 | 13.51 |
| KDMG (ours) | **3.13** | **4.79** | **6.20** | 8.68 |

SKU-110K [44] and DOTA [45], which contains more than one semantic categories. SKU-110K contains 11,762 high-resolution images with 110,712 classes. The object number varies from 1 to 718. For DOTA, we only use 690 images with 6 classes (Large-vehicle, Helicopter, Plane, Ship, Small-vehicle, and Storage tank) with object numbers larger than 10. Table 1 presents a summary of the datasets, while Figure 5 shows a few example images.

TABLE 7
Experiment results on PUCPR+, CARPK, and SKU110k.

| Method | PUCPR+ | | CARPK | | SKU110k | |
|---|---|---|---|---|---|---|
| | MAE↓ | MSE↓ | MAE↓ | MSE↓ | MAE↓ | MSE↓ |
| Faster R-CNN [57] | 39.88 | 47.67 | 24.32 | 37.62 | 107.46 | 113.42 |
| YOLO (2016) [58] | 156.00 | 200.42 | 48.89 | 57.55 | 84.17 | 97.81 |
| One-Look (2016) [59] | 21.88 | 36.73 | 59.46 | 66.84 | - | - |
| LPN Counting (2017) [43] | 22.76 | 34.46 | 23.80 | 36.79 | - | - |
| YOLO9000$^{opt}$ [60] | 130.43 | 172.46 | 45.36 | 52.02 | - | - |
| RetinaNet (2018) [61] | 24.58 | 33.12 | 16.62 | 22.30 | 16.58 | 30.70 |
| IEP Counting (2019) [62] | 15.17 | - | 51.83 | - | - | - |
| Densely Packed (2019) [44] | 7.16 | 12.00 | 6.77 | 8.52 | 14.52 | 23.99 |
| ADMG (ours) | 3.57 | 5.02 | 7.14 | 8.59 | 12.69 | **20.72** |
| KDMG (ours) | **3.01** | **4.38** | **5.17** | **6.94** | **11.20** | **19.88** |

## 4.2 Experiment Setup

Our baseline counters include CSRNet [50], SFCN [36], MCNN [4], and FCN [5]. The training procedures for the counters follow the original papers: SGD [63] is used to train CSRNet with learning rate set to 5e-7; The Adam optimizer [64] is used to train SFCN, FCN and MCNN with learning rate 1e-5. Traditional Gaussian density maps are generated following [50]. The fixed bandwidth is set to 4 and 16.

Our methods are denoted as Density Map Refinement (DMR, Section 3.2), Adaptive Density Map Generation (ADMG, Sec-

tion 3.3), and Kernel-based Density Map Generation (KDMG, Section 3.4).[3] The architectures of DMR, ADMG, and KDMG are summarized in Table 2, and are designed as standard Fully Convolutional Neural Networks. We use 3 pooling layers for the fusion network so that the resolution of the generator output is the same as the counter. We use 4 pooling layers for the Kernel Generator to reduce computational cost. For our methods, Adam optimizer is used for training with learning rate of 1e-7.

The methods are evaluated using mean absolute error (MAE) and root mean squared error (RMSE):

$$MAE = \frac{1}{N}\sum_i |\hat{y}_i - y_i|, \quad RMSE = \sqrt{\frac{1}{N}\sum_i \|\hat{y}_i - y_i\|^2},$$

where $N$ is the number of samples and $\hat{y}_i$, $y_i$ are the predicted and ground truth counts. For vehicle counting, we also use the GAME metric [42] on TRANCOS,

$$GAME(L) = \frac{1}{N}\sum_i \sum_{l=1}^{4^L} |\hat{y}_i^l - y_i^l|,$$

where the images are divided into $2^L$ non-overlapping regions.

### 4.3 Crowd Counting

We compare our proposed density map refinement and generation frameworks with state-of-the-art methods on ShanghaiTech A [4], ShanghaiTech B [4], UCF-QNRF [33], JHU-CROWD++ [40], and NWPU-Crowd [41]. Here we use CSRNet [50] as the baseline counting model to be trained with DMR, ADMG, and KDMG. The experiment results are shown in Tables 3, 4, and 5. Note that CSRNet$^\dagger$ refers to results reported in [50], while CSRNet refers to our baseline implementation.

On the ShanghaiTech A dataset, both proposed generation frameworks (ADMG and KDMG) achieve superior performance compared with state-of-the-arts on MAE. The proposed generation methods also outperform the baseline model, CSRNet [50], which further confirms the effectiveness of learning the density map representation. Similarly, On ShanghaiTech B, both of our frameworks achieve better performance than the baseline CSRNet. UCF-QNRF is a challenging crowd counting dataset, and our proposed methods achieve the best performance on both MAE and MSE. On the two large-scale datasets, NWPU-Crowd and JHU-CROWD++, KDMG significantly outperforms ADMG by a large margin; KDMG also performs favorably versus other state-of-the-art methods such as BL [15]. Comparing the two generators, in general, KDMG has lower MAE than ADMG. In summary, these experiments demonstrate that the proposed density map generation frameworks can produce learnable density map representations that improve counting performance, especially on large datasets such as ShanghaiTech A/B, UCF-QNRF, NWPU-Crowd and JHU-CROWD++.

### 4.4 Vehicle Counting

We also evaluate the performance of the proposed method on vehicle counting. The experiment results are shown in Tables 6 and 7. The proposed method outperforms CSRNet in both global counting (GAME(0)) and local counting (GAME(1), GAME(2) and GAME(3)), since the proposed framework can generate better kernels. PSDDN [39] is a detection based approach, and thus the precise localization performance (GAME(3)) is better than the proposed approach. However, we achieve superior counting performance, GAME(0), and rough localization, i.e., GAME(1) and GAME(2). For the parked car counting on PUCPR+ and CARPK, the proposed framework achieves the best performance on both MAE and MSE, which confirms that the proposed method is effective to count vehicle number on both road and parking lot accurately.

3. Code available at http://visal.cs.cityu.edu.hk/research/kdmg/

TABLE 8
Experiment results on DOTA.

| Method | MAE↓ | MSE↓ |
|---|---|---|
| CSRNet + Fixed kernel ($\sigma$=4) | 4.82 | 10.17 |
| CSRNet + Fixed kernel ($\sigma$=16) | 5.10 | 9.02 |
| CSRNet + Adaptive kernel | 6.05 | 8.95 |
| ADMG (ours) | 4.42 | 8.38 |
| KDMG (ours) | **3.65** | **7.44** |

### 4.5 General Object Counting

We evaluate general object counting performance on SKU-110K and DOTA. For general object counting, following [62], we count all objects at the same time, regardless of object type. The task is more challenging since the shape and size varies dramatically across different objects. We use the same setup for all comparison methods. The results are shown in Tables 7 and 8. The density map based approaches are better than detection based approaches, demonstrating the advantages of density maps for dense counting. The new KDMG achieves better performance than ADMG, which demonstrates the necessity of learning the kernels directly.

### 4.6 Ablation Studies

We next present ablation studies to justify the design choices of our framework.

#### 4.6.1 Density Maps

We first compare the effectiveness of different traditional density maps and our generated density maps. For fixed bandwidth kernels, the bandwidths are set to 4 and 16. As shown in Table 9, the learned density maps from ADMG/KDMG generally outperform manually generated density maps including fixed kernels and adaptive kernels. KDMG achieves superior performance over ADMG, since the former preserves the true count in the learned density map.

#### 4.6.2 Kernel Size

The effect of kernel size $k$ on the density maps produced by KDMG is investigated on ShTech A and B, and the results are presented in Figure 6 (a). For ShTech A, smaller kernels tend to yield better performance since the crowd is dense. In ShTech B, the best performance is achieved when kernel size is 7 since the crowd is less dense. We thus set kernel size to 5 and 7 for ShTech A and B, respectively. On the other crowd datasets, we use $k$=5 since they have similar average head size to ShTech A. We also set $k$=5 for the vehicle and general object counting datasets.

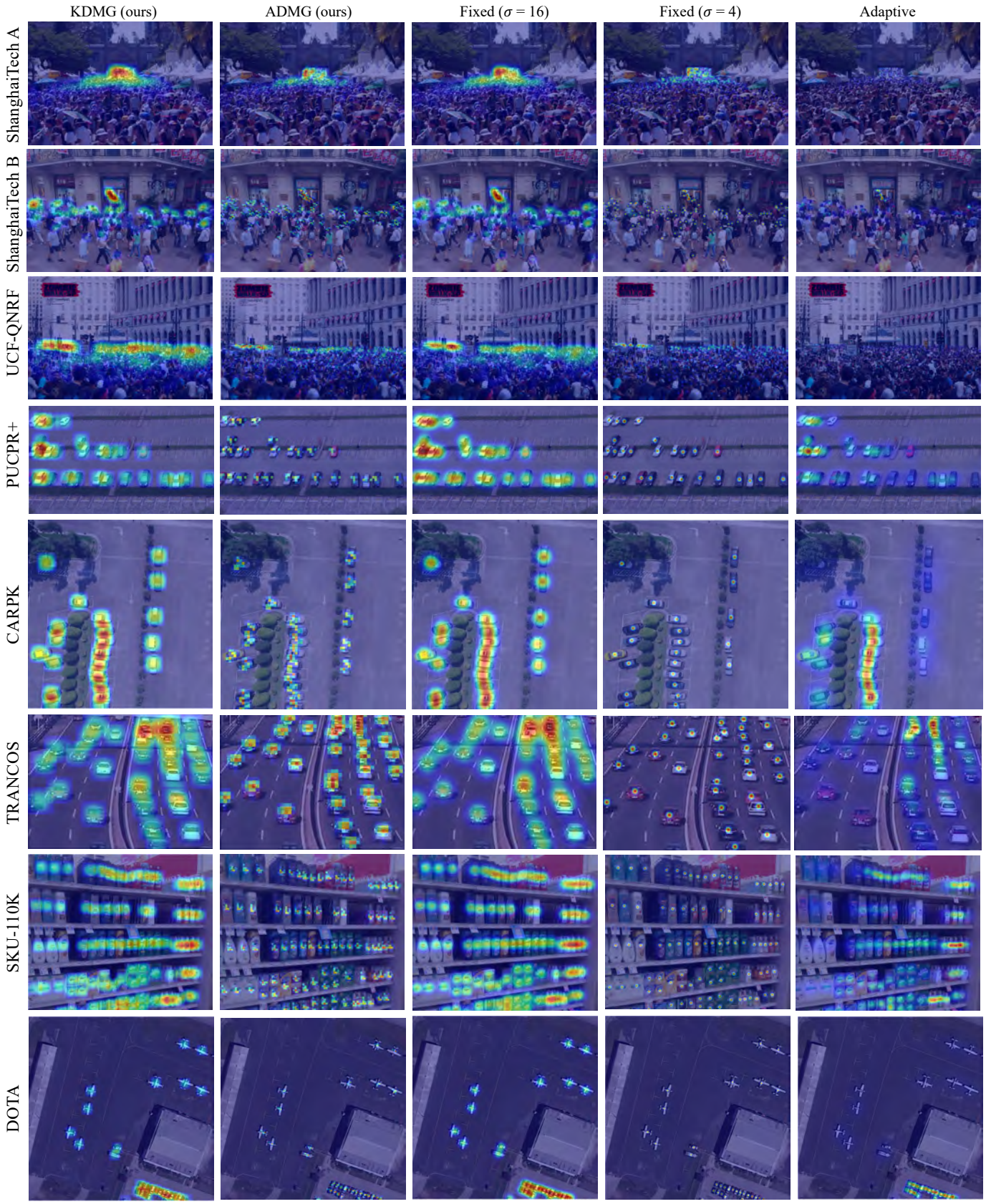


Fig. 7. Comparison of different density maps. From left to right: learned density maps from KDMG and ADMG, density maps generated by fixed kernel ($\sigma = 16$), density maps generated by fixed kernel ($\sigma = 4$), and density maps generated by adaptive-bandwidth Gaussian kernels.

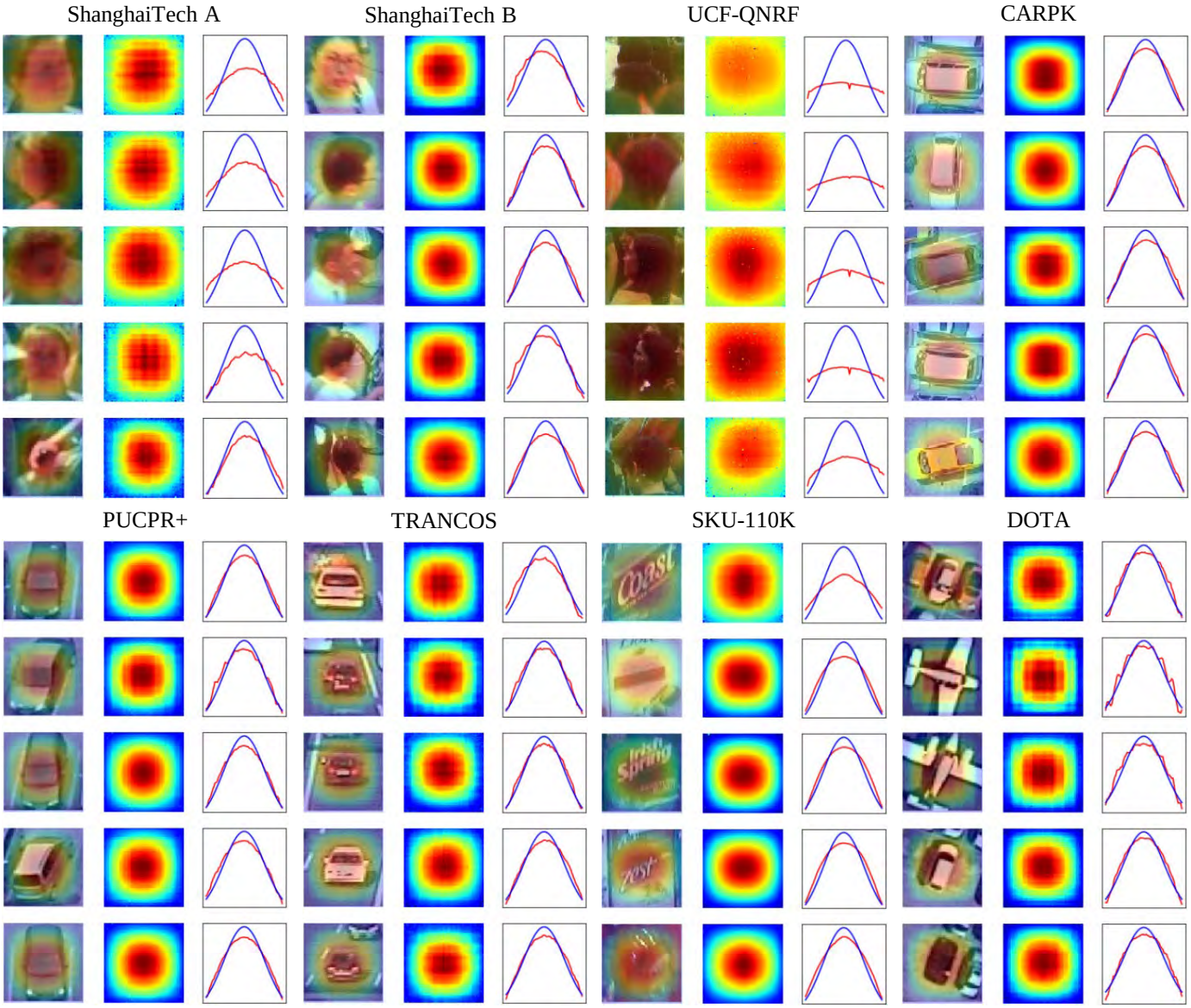


Fig. 8. Comparison of learned kernels from KDMG (red) and fixed kernel with bandwidth 16 (blue). Best viewed in color.

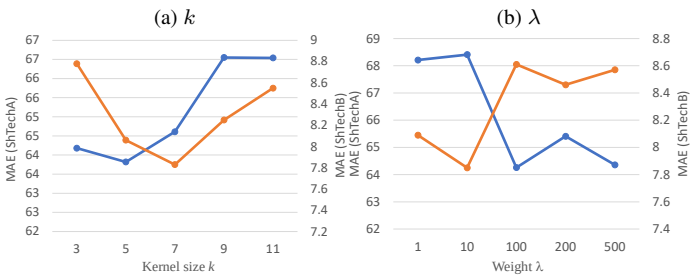


Fig. 6. MAE vs. (a) kernel size and (b) weight for cosine regularizer.

#### 4.6.3 Regularization

The cosine similarity is used as a regularizer for spatial consistency. The effect of the weight $\lambda$ in (10) is investigated on ShanghaiTech A and ShanghaiTech B, and the results are shown in Figure 6 (b). On ShanghaiTech A, larger weights tend to generate better performance since people are close to each other in ShanghaiTech A. Under this circumstance, spatial regularization is effective to learn the spatial density distribution. On ShanghaiTech B, smaller weights tend to achieve better performance since people are separated from each other.

#### 4.6.4 Generalization Ability

To evaluate the generalization ability of the learned density maps, we conducted an experiment on ShanghaiTech A and ShanghaiTech B using the generated density maps trained for CSRNet as the ground-truth density maps to train MCNN, FCN, and SFCN. The results are shown in Table 10, where "ADMG (jointly trained)" is the generator-estimator jointly trained together and "ADMG for CSRNet" uses the density maps generated for CSRNet to train other estimators. The results show that density maps generated for one estimator (CSRNet) do not generalize well to other estimators (MCNN, FCN, SFN), as the error for "ADMG for CSRNet" is higher than the jointly-trained generator, and in general similar to fixed/adaptive kernels. This demonstrates that the generated density maps are matching particular properties (e.g., receptive field size, network depth) of the estimators to improve the counting accuracy. In this case, as CSRNet is a large complex network, the density maps for CSRNet might be too complex for simpler networks (MCNN, FCN) to predict correctly.

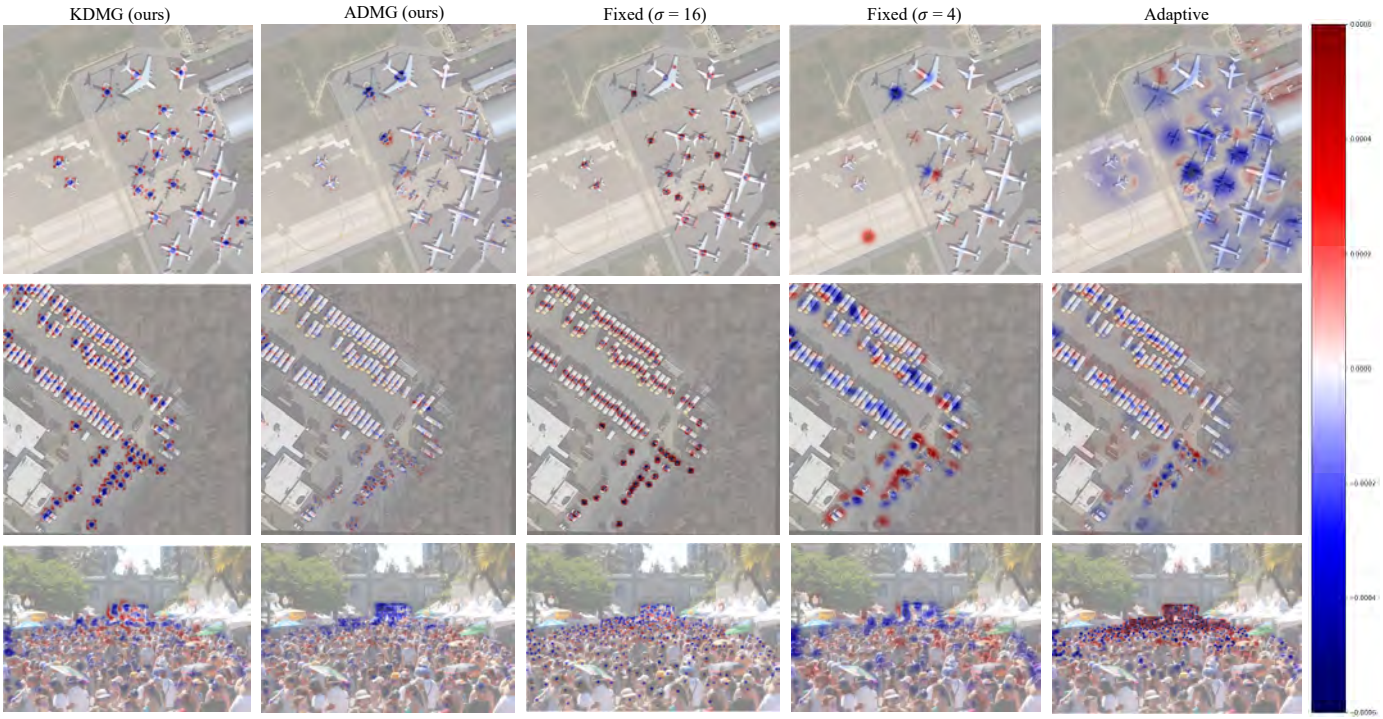


Fig. 9. Error maps (prediction - truth) for different types of density maps.

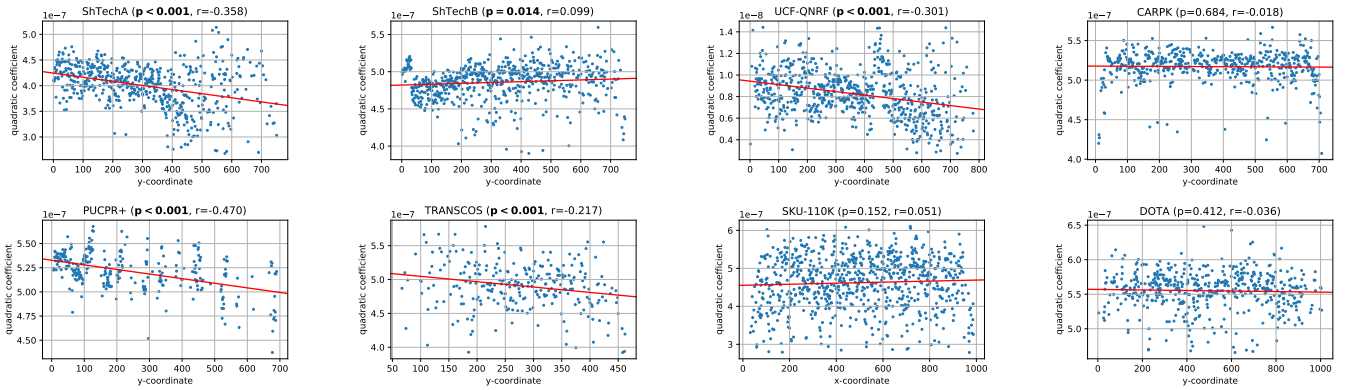


Fig. 10. The quadratic coefficient magnitudes of kernels vs. their x-coordinate (SKU-110K dataset) and y-coordinate (other datasets). The best-fit line is in red, and the title shows the p-values for testing for significant correlations.

#### 4.6.5 *Self-attention module*

To evaluate the effectiveness of the self-attention module in ADMG, we compare using the self-attention module with three variants: 1) "image-att" generates attention from the input image; 2) "direct fusion" directly fuses without attention; 3) "naive-fusion" directly sums all density maps. As shown in Tab. 11, the fusion is more effective with self-attention ("self-att") than with the input image ("image-att"). The possible reason is that the crowd information is directly obtainable from the density maps, whereas this information needs to be decoded and interpreted from the image, which introduces additional complexity and noise.

#### 4.6.6 *Local Counting Performance*

To investigate the local counting performance, we evaluate the frameworks based on GAME metric on UCF-QNRF. The result is shown in Table 12. The local counting performance of ADMG is worse than the traditional method, while KDMG achieves better local counting performance than the baseline CSRNet. KDMG has better local performance because it composites a set of kernels with fixed $k \times k$ size, which keeps the local density regions smooth. In contrast, ADMG uses a series of convolution layers that tends to make the density regions more compact (see Fig. 7), which cause errors near boundaries of the GAME image patches.

#### 4.6.7 *Visualization*

To better understand the generation framework, we compare the learned density maps on different datasets with traditional density maps in Figure 7. For small-bandwidth density maps (fourth column) and ADMG density maps (second column), the density only appear on part of the object, which results in sparse density maps. For density maps generated by adaptive kernels (last column), the density is too smooth for sparse objects. The density maps generated by fixed kernel with bandwidth 16 (3rd column) are similar to KDMG density maps (1st column), which can better cover the whole objects with less leakage to the background.

We next visualize the learned individual kernels for KDMG and fixed kernels with bandwidth 16 in Figure 8. Overall, the learned kernels are more flat than fixed kernels especially for very dense images in UCF-QNRF. Since the counters use max-pooling layers which will produce translation invariant features, it will be better to have the same density value for shifted patches.

TABLE 9
Comparison of traditional and generated density maps, evaluated with MAE. $\sigma$ is the bandwidth for the fixed kernel.

| Counter | Density Map | ShTech A | ShTech B |
|---|---|---|---|
| MCNN | Fixed kernel ($\sigma$=16) | 95.4 | 18.7 |
| | Fixed kernel ($\sigma$=4) | 96.0 | 17.9 |
| | Adaptive kernel | 103.3 | 17.9 |
| | ADMG (ours) | 93.5 | 17.7 |
| | KDMG (ours) | **91.0** | **16.6** |
| FCN | Fixed kernel ($\sigma$=16) | 90.7 | 18.8 |
| | Fixed kernel ($\sigma$=4) | 88.9 | 13.8 |
| | Adaptive kernel | 95.4 | 16.0 |
| | ADMG (ours) | 87.1 | 13.9 |
| | KDMG (ours) | **84.7** | **12.8** |
| SFCN | Fixed kernel ($\sigma$=16) | 70.8 | 9.9 |
| | Fixed kernel ($\sigma$=4) | 70.8 | 10.6 |
| | Adaptive kernel | 73.1 | 9.7 |
| | ADMG (ours) | 68.4 | 8.4 |
| | KDMG (ours) | **67.5** | **8.3** |
| CSRNet | Fixed kernel ($\sigma$=16) | 67.8 | 12.1 |
| | Fixed kernel ($\sigma$=4) | 70.1 | 9.5 |
| | Adaptive kernel | 66.4 | 10.6 |
| | ADMG (ours) | 64.7 | 8.1 |
| | KDMG (ours) | **63.7** | **7.9** |

TABLE 10
The experiment results on the generalization ability of generated density maps. Results are for MAE on ShanghaiTech.

| Density map | ShanghaiTech A | | | ShanghaiTech B | | |
|---|---|---|---|---|---|---|
| | MCNN | FCN | SFCN | MCNN | FCN | SFCN |
| Fixed kernel ($\sigma$=16) | 95.4 | 90.7 | 70.8 | 18.7 | 18.8 | 9.9 |
| Fixed kernel ($\sigma$=4) | 96.0 | 88.9 | 70.8 | 17.9 | 13.8 | 10.6 |
| Adaptive kernel | 103.3 | 95.4 | 73.1 | 17.9 | 16.0 | 9.7 |
| ADMG (jointly trained) | 93.5 | 87.1 | 68.4 | 16.6 | 12.8 | 8.3 |
| ADMG for CSRNet | 95.2 | 94.8 | 70.1 | 17.0 | 14.0 | 8.8 |

We also show the error maps (prediction − ground-truth) for different types of density maps in Figure 9. Density maps that are too smooth, such as "Fixed ($\sigma = 16$)" and "Adaptive", usually have more false positive since many background pixels are treated as the positive samples during the training of the counter. Density maps that are too sparse, such as "ADMG (ours)" and "Fixed ($\sigma = 4$)", will have more false negatives since many object pixels are treated as the negative samples during the training. For the learned kernels of KDMG, most errors occur inside the kernel with smaller values at the center and larger values at the boundary. These errors will complement each other and result in accurate global count. This also explains why flat kernels achieve better performance than tradition Gaussian kernels.

We also fit the profiles of the learned kernels from KDMG to quadratic functions. In particular, we randomly selected 70 images for each dataset. Then, the images are split into several regions based on the height or width, and the average coefficient is calculated for each region. The correlation statistics and p-value are calculated using data from the 70 images. The magnitude of the quadratic coefficient (the coefficient is always negative) can be used to represent the flatness of a kernel. For each dataset, the magnitude of the quadratic coefficient of a kernel versus its x-coordinate (SKU-110K dataset) or y-coordinate (other datasets) is shown in Figure 10, where the origin (0,0) is the top-left corner of the image. We also plot the best fit line (via linear regression), and test whether there is a significant correlation between the coefficient magnitude and the y-coordinate (or x-coordinate), i.e., whether the line's slope is significantly different from zero.

When the images contain dense crowds under perspective effects (as in ShTech A, UCF-QNRF, PUCPR+, and TRANSCOS), we found that the quadratic coefficient's magnitude is negatively correlated with kernel's y-coordinate. In other words, for these scenes, the curvature of the kernel adapts to placement of people in the scene. For small people far from the camera (small y-coordinates), the quadratic coefficient has larger magnitude, yielding a sharper kernel. In contrast, for large people close to the camera (large y-coordinates), the coefficient has smaller magnitude, yielding a flatter kernel. A possible explanation for using flatter kernels closer to the camera is that flat structures are easier to predict through the max-pooling layer (as discussed above), and the kernels are less likely to overlap since close people are more spread out in the image space. On the opposite, far people are more likely to have overlapping kernels, and thus a sharper kernel is used so that the densities in overlapping regions does not get too large, compared to the peak value on the person. For the overhead-view images (DOTA, CARPK) or side-on view (SKU-110K), there was no significant correlation between the kernel shape and its location, mainly because the object density and inter-object distance do not change significantly within the image. Finally, for ShTech B, there was a small positive correlation between the coefficient magnitude and the y-coordinate – people closer to the camera (large y-coordinates) had slightly sharper kernels (larger coefficient magnitude), compared to people further away. Perhaps this is a side-effect caused by the large variations in camera orientations and heights for images in ShTech B (compared to the other datasets that are more homogenous), which also explains the small effect size (r=0.099). In summary, we found that the kernel-shape changes within the image, based on properties of the scene, and thus KDMG is able to optimize the kernel shape to match the scene and the particular density estimation network.

TABLE 11
Ablation study of self-attention module on ShanghaiTech A. MAE↓ is used as the metric.

| | self-att | image-att | direct-fusion | naive-fusion |
|---|---|---|---|---|
| MAE↓ | **64.7** | 66.9 | 67.5 | 68.6 |

TABLE 12
The evaluation of local counting performance on UCF-QNRF.

| | GAME0 | GAME1 | GAME2 | GAME3 |
|---|---|---|---|---|
| MCNN [4] | 177.38 | 220.01 | 257.56 | 296.33 |
| CSRNet [50] | 148.12 | 179.73 | 220.66 | 267.98 |
| ADMG | 100.99 | 129.66 | 166.54 | 207.71 |
| KDMG | **99.54** | **116.91** | **134.95** | **160.19** |

## 5 Conclusion

In this paper, a general density map generation framework is proposed to learn effective density kernels for different objects. Traditional density maps are manually generated by either fixed Gaussian kernels or adaptive-bandwidth Gaussian kernels, which is not optimal. A density map generation framework is proposed to jointly learn a density map generator and density map estimator. The proposed framework can potentially learn effective density maps matching specific capabilities of the counter and specific properties of the objects in the dataset. We evaluate the proposed

framework on 10 datasets for 3 applications. The best performance is achieved on these datasets, which confirms the effectiveness of the proposed methods.

## ACKNOWLEDGEMENTS

We thank the reviewers for their helpful comments. This work was supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. [T32-101/15-R] and CityU 11212518), and by a Strategic Research Grant from City University of Hong Kong (No. 7004887).

## REFERENCES

[1] J. Gao, Q. Wang, and X. Li, "Pcc net: Perspective crowd counting via spatial convolutional network," *IEEE Trans. CSVT*, 2019.

[2] J. Gao, W. Lin, B. Zhao, D. Wang, C. Gao, and J. Wen, "C^ 3 framework: An open-source pytorch code for crowd counting," *arXiv preprint arXiv:1907.02724*, 2019.

[3] J. Wan, W. Luo, B. Wu, A. B. Chan, and W. Liu, "Residual regression and semantic prior for crowd counting," in *CVPR*, 2019, pp. 4036–4045.

[4] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *CVPR*, 2016, pp. 589–597.

[5] D. Kang and A. B. Chan, "Crowd counting by adaptively fusing predictions from an image pyramid," in *BMVC*, 2018, p. 89.

[6] V. A. Sindagi and V. M. Patel, "Generating high-quality crowd density maps using contextual pyramid cnns," in *ICCV*, 2017, pp. 1879–1888.

[7] X. Liu, J. van de Weijer, and A. D. Bagdanov, "Leveraging unlabeled data for crowd counting by learning to rank," in *CVPR*, 2018.

[8] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *NeurIPS*, 2010, pp. 1324–1332.

[9] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015, pp. 833–841.

[10] M. Li, Z. Zhang, K. Huang, and T. Tan, "Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection," in *ICPR*, 2008, pp. 1–4.

[11] W. Ge and R. T. Collins, "Marked point processes for crowd counting," in *CVPR*, 2009, pp. 2913–2920.

[12] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *CVPR*, 2008, pp. 1–7.

[13] A. B. Chan and N. Vasconcelos, "Bayesian poisson regression for crowd counting," in *ICCV*, 2009, pp. 545–551.

[14] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *CVPR*, 2013, pp. 2547–2554.

[15] Z. Ma, X. Wei, X. Hong, and Y. Gong, "Bayesian loss for crowd count estimation with point supervision," in *ICCV*, 2019, pp. 6142–6151.

[16] X. Cao, Z. Wang, Y. Zhao, and F. Su, "Scale aggregation network for accurate and efficient crowd counting," in *ECCV*, 2018, pp. 734–750.

[17] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," in *CVPR*, 2017, pp. 4031–4039.

[18] D. Babu Sam, N. N. Sajjan, R. Venkatesh Babu, and M. Srinivasan, "Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn," in *CVPR*, 2018, pp. 3618–3626.

[19] X. Jiang, Z. Xiao, B. Zhang, X. Zhen, X. Cao, D. Doermann, and L. Shao, "Crowd counting and density estimation by trellis encoder-decoder networks," in *CVPR*, 2019, pp. 6133–6142.

[20] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, and H. Wu, "Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding," in *CVPR*, June 2019.

[21] V. A. Sindagi and V. M. Patel, "Multi-level bottom-top and top-bottom feature fusion for crowd counting," in *ICCV*, 2019, pp. 1002–1012.

[22] V. Ranjan, H. Le, and M. Hoai, "Iterative crowd counting," in *ECCV*, 2018, pp. 278–293.

[23] D. B. Sam and R. V. Babu, "Top-down feedback for crowd counting convolutional neural network," in *AAAI*, 2018, pp. 7323–7330.

[24] L. Liu, H. Wang, G. Li, W. Ouyang, and L. Lin, "Crowd counting using deep recurrent spatial-aware network," in *IJCAI*, 2018, pp. 849–855.

[25] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *ECCV*. Springer, 2016, pp. 660–676.

[26] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, and G. Zheng, "Crowd counting with deep negative correlation learning," in *CVPR*, 2018, pp. 5382–5390.

[27] F. Xiong, X. Shi, and D.-Y. Yeung, "Spatiotemporal modeling for crowd counting in videos," in *ICCV*, 2017, pp. 5161–5169.

[28] D. Kang, Z. Ma, and A. B. Chan, "Beyond counting: comparisons of density maps for crowd analysis tasks-counting, detection, and tracking," *IEEE Trans. CSVT*, 2018.

[29] C. Liu, X. Weng, and Y. Mu, "Recurrent attentive zooming for joint crowd counting and precise localization," in *CVPR*, 2019, pp. 1217–1226.

[30] W. Liu, M. Salzmann, and P. Fua, "Context-aware crowd counting," in *CVPR*, 2019, pp. 5099–5108.

[31] M. Shi, Z. Yang, C. Xu, and Q. Chen, "Revisiting perspective information for efficient crowd counting," in *CVPR*, 2019, pp. 7279–7288.

[32] Q. Zhang and A. B. Chan, "Wide-area crowd counting via ground-plane density maps and multi-view fusion cnns," in *CVPR*, 2019, pp. 8297–8306.

[33] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, and M. Shah, "Composition loss for counting, density map estimation and localization in dense crowds," in *ECCV*, 2018, pp. 532–546.

[34] D. Lian, J. Li, J. Zheng, W. Luo, and S. Gao, "Density map regression guided detection network for rgb-d crowd counting and localization," in *CVPR*, 2019, pp. 1821–1830.

[35] M. Zhao, J. Zhang, C. Zhang, and W. Zhang, "Leveraging heterogeneous auxiliary tasks to assist crowd counting," in *CVPR*, 2019, pp. 12 736–12 745.

[36] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from synthetic data for crowd counting in the wild," in *CVPR*, 2019.

[37] V. A. Sindagi and V. M. Patel, "A survey of recent advances

in cnn-based single image crowd counting and density estimation," *Pattern Recognition Letters*, vol. 107, pp. 3–16, 2018.
[38] J. Wan and A. B. Chan, "Adaptive density map generation for crowd counting," in *ICCV*, 2019.
[39] Y. Liu, M. Shi, Q. Zhao, and X. Wang, "Point in, box out: Beyond counting persons in crowds," in *CVPR*, 2019, pp. 6469–6478.
[40] V. A. Sindagi, R. Yasarla, and V. M. Patel, "Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method," *arXiv preprint arXiv:2004.03597*, 2020.
[41] Q. Wang, J. Gao, W. Lin, and X. Li, "Nwpu-crowd: A large-scale benchmark for crowd counting," *arXiv preprint arXiv:2001.03360*, 2020.
[42] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Onoro-Rubio, "Extremely overlapping vehicle counting," in *IbPRIA*. Springer, 2015, pp. 423–431.
[43] M.-R. Hsieh, Y.-L. Lin, and W. H. Hsu, "Drone-based object counting by spatially regularized regional proposal network," in *ICCV*, 2017, pp. 4145–4153.
[44] E. Goldman, R. Herzig, A. Eisenschtat, J. Goldberger, and T. Hassner, "Precise detection in densely packed scenes," in *CVPR*, June 2019.
[45] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *CVPR*, 2018, pp. 3974–3983.
[46] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Fully convolutional crowd counting on highly congested scenes," in *VISIGRAPP*, 2017, pp. 27–33.
[47] V. A. Sindagi and V. M. Patel, "Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in *AVSS*, 2017, pp. 1–6.
[48] Z. Shen, Y. Xu, B. Ni, M. Wang, J. Hu, and X. Yang, "Crowd counting via adversarial cross-scale consistency pursuit," in *CVPR*, 2018, pp. 5245–5254.
[49] Z. Wang, Z. Xiao, K. Xie, Q. Qiu, X. Zhen, and X. Cao, "In defense of single-column networks for crowd counting," in *BMVC*, 2018, p. 78.
[50] Y. Li, X. Zhang, and D. Chen, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *CVPR*, 2018, pp. 1091–1100.
[51] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. PAMI*, 2017.
[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[53] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.
[54] D. B. Sam, S. V. Peri, M. N. Sundararaman, A. Kamath, and V. B. Radhakrishnan, "Locate, size and count: Accurately resolving people in dense crowds via detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[55] L. Liu, Z. Qiu, G. Li, S. Liu, W. Ouyang, and L. Lin, "Crowd counting with deep structured scale integration network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1774–1783.
[56] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt, "Where are the blobs: Counting by localization with point supervision," in *ECCV*, 2018, pp. 547–562.
[57] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015, pp. 91–99.
[58] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.
[59] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," in *ECCV*. Springer, 2016, pp. 785–800.
[60] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *CVPR*, 2017, pp. 7263–7271.
[61] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.
[62] T. Stahl, S. L. Pintea, and J. C. van Gemert, "Divide and count: Generic object counting by image divisions," *IEEE Trans. IP*, vol. 28, no. 2, pp. 1035–1044, 2018.
[63] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *COMPSTAT*. Springer, 2010, pp. 177–186.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

**Jia Wan** received the B.Eng. degree in software engineering from Northwestern Polytechnical University, Xi'an, China, and M.Phil. degree from School of Computer Science and the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an, China, in 2015 and 2018, respectively. He is currently working towards the Ph.D. degree in Computer Science at the City University of Hong Kong. His research interests include congestion analysis and crowd counting.

**Qinzhong Wang** received the B.Eng. and M.Eng. degrees in control science and engineering from Harbin Engineering University, Harbin, China, in 2013 and 2016. Now he is a Ph.D candidate in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision, natural language processing and generative models.

**Antoni B. Chan** received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.