Rethinking Spatial Invariance of Convolutional Networks for Object Counting

Zhi-Qi Cheng[1], Qi Dai[2], Hong Li[3*], Jingkuan Song[4], Xiao Wu[3], Alexander G. Hauptmann[1]
[1]Carnegie Mellon University [2]Microsoft Research Asia [3]Southwest Jiaotong University
[4]University of Electronic Science and Technology of China
{zhiqic,alex}@cs.cmu.edu, qid@microsoft.com,
{hl1997.work,jingkuan.song}@gmail.com, wuxiaohk@home.swjtu.edu.cn

Abstract

Previous work generally believes that improving the spatial invariance of convolutional networks is the key to object counting. However, after verifying several mainstream counting networks, we surprisingly found too strict pixel-level spatial invariance would cause overfit noise in the density map generation. In this paper, we try to use locally connected Gaussian kernels to replace the original convolution filter to estimate the spatial position in the density map. The purpose of this is to allow the feature extraction process to potentially stimulate the density map generation process to overcome the annotation noise. Inspired by previous work, we propose a low-rank approximation accompanied with translation invariance to favorably implement the approximation of massive Gaussian convolution. Our work points a new direction for follow-up research, which should investigate how to properly relax the overly strict pixel-level spatial invariance for object counting. We evaluate our methods on 4 mainstream object counting networks (i.e., MCNN, CSRNet, SANet, and ResNet-50). Extensive experiments were conducted on 7 popular benchmarks for 3 applications (i.e., crowd, vehicle, and plant counting). Experimental results show that our methods significantly outperform other state-of-the-art methods and achieve promising learning of the spatial position of objects[1].

1. Introduction

Object counting has been widely studied since it can potentially solve crowd flow monitoring, traffic management, etc. The previous works [8, 28, 62] believe that the latchkey to improving the object counting is to improve the spatial invariance of CNNs. Based on this starting point, more and more networks (such as dilated CNNs [3, 13, 39], deformable CNNs [17, 34] and multi-column CNNs [11, 13, 71]) are studied for object counting.

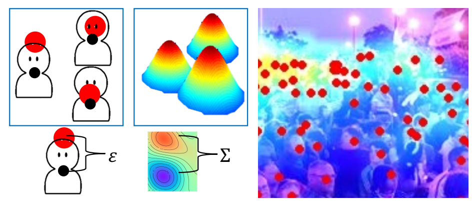

Figure 1. The left shows the idea of density map generation, and the right is an example from SHTech-PartA dataset [76], where the red dot is the annotation in groundtruth, and the black dot is the real center position. The density map is generated by smoothing the center points with the multi-dimensional Gaussian distribution. There are two main types of noise: 1) the error ϵ between the true center points and the annotations and 2) the overlap $\mathbf{\Sigma}$ caused by multiple Gaussian kernels. Note that the left is merely an example. The center point of crowd counting usually refers to the center of the head. [**Best view in color**].

However, this research direction has appeared performance bottlenecks. We noticed that the counting accuracy had not been significantly improved with further continuously optimizing the network architectures. Some recent studies [6, 10, 30, 59] also witnessed a lot of noise during density generation and conjecture that this might be the reason for the performance bottleneck. Although these efforts have made some progress, we are still ignorant of the following questions. *1) Is blindly improving spatial invariance valuable for object counting tasks? 2) How does density noise affect performance?*

Before answering these questions, let's briefly introduce the generation process of the density map. Figure 1 takes crowd counting as an example. The density map is generated by smoothing the center point with multiple Gaussian kernels. This preprocessing converts the discrete counting problem into a continuous density regression, but inevitably brings some noise. In general, there are two types of noise. 1) The error between the actual center point and annotation (i.e., ϵ between the red and black dots). 2) The overlay of Gaussian kernels (i.e., $\mathbf{\Sigma}$)[2]. More formal mathematical description is in Sec. 3.1 and 3.2.

*Work done during a remote research collaboration with CMU.
[1]Code is at https://github.com/zhiqic/Rethinking-Counting

[2]Note that we have some abuse symbols here.

To answer these problems, we have thoroughly verified four mainstream object counting methods (MCNN [77], CSRNet [28], SANet [4] and ResNet-50 [18]) in three different tasks (crowd, vehicles and plants counting). Extensive verification experiments reveal that too strict pixel-level spatial invariance will not only cause the large prediction variances, but also overfitting to the noise in the density map as Sec. 4.2. We observed that the existing models 1) cannot be generalized, even impossible within the same crowd counting task and 2) essentially impossible to learn the actual object position and distribution in the density maps. In general, these experiments provide the following answers. *1) Solely increasing the spatial invariance is not beneficial to object counting tasks. 2) The pixel-level spatial invariance makes the model easier to overfit the density map noise.*

To solve these problems, inspired by the previous works [15, 19, 26, 57], we try to replace the traditional convolution operation with Gaussian convolution. The motivation behind is to mimic the Gaussian-style density generation throughout the whole feature learning, rather than merely generating the final density map. To a certain extent, this modification is equivalent to a relaxation of the pixel-level spatial invariance. After the pixel-grid filters are revised with Gaussian kernels, we can jump out of the over-strict pixel-level restrictions. Fortunately, the experimental result of Sec. 4.4 proved that this relaxation could allow us to avoid overfitting to the density map noise and promisingly learn the object position and distribution law.

Technically, we propose a novel low-rank approximation to simulate the process of Gaussian-style density map generation during the feature extraction. Although previous work [59] uses a multivariate Gaussian approximation to optimize the density map in the loss function, it is unclear how to explicitly model this approximation during the convolution process. Note that the approximation in [59] only imposes the constraints on predicted density maps, while leaving the density estimation unchanged. In contrast, our approach employs Gaussian convolution to replace standard convolution, where our low-rank approximation uses finite Gaussian kernels (Eq. 10) to approximate the massive Gaussian kernel convolution (Eq. 7). It is worth noting that our method concentrates on the density estimation process, while [59] only focuses on the generated density maps.

As shown in Figure 3, we replace the standard convolution operation with Gaussian convolution to provide a novel way to generate the density map. We first propose a *Low-rank Approximation module* to approximate the massive Gaussian convolution. Specifically, we sample a few Gaussian kernels from the groundtruth density map as input, and then employ Principal Component Analysis (PCA) to select some representative Gaussian kernels. Through a simple attention mechanism, the correlation between the selected Gaussian kernels is learned, which is operated to approximate the massive Gaussian convolution. Correspondingly, we also propose a *Translation Invariance Module* to accelerate the inference. On the input side, we adopt the translation invariance to decouple the Gaussian kernel operation to accelerate the convolution operation. At the output side, we utilize the weights obtained from the low-rank approximation module to accomplish approximation. Note that all of our implementations are based on CUDA. It can be seamlessly applied to mainstream CNNs and is end-to-end trainable. To conclude, our contributions are mainly three folds:

- We reveal that the overly restrictive spatial invariance in object counting is unnecessary or even harmful when facing the noises in the density maps.
- A low-rank Gaussian convolution is proposed to handle the noises in density map generation. Equipped with low-rank approximation and translation invariance, we can favorably replace standard convolutions with several Gaussian kernels.
- Extensive experiments on seven datasets for three counting tasks (i.e. crowd, vehicle, plant counting) fully demonstrate the effectiveness of our method.

2. Related works

We divide the literature into two directions as follows.

2.1. Increase the spatial invariance with CNNs

Different from traditional manually designed counting detectors [2, 5, 40, 47], existing mainstream methods convert counting problems into density regression [8,27,62,74]. The main research direction is to improve the spatial invariance of CNNs. The mainstream technical routes include Multi-Column CNNs [11–13, 71, 77], Dilation CNNs [3, 13, 17, 28, 39, 70], Deformable CNNs [17, 34], Residual CNNs [29, 43, 78], Graph CNNs [38], Attention Mechanism [14,25,44,52,72,73], Pyramid Pooling [9,21,50], and Hierarchy/Hybrid Structures [38,51]. With the further optimization of parameters and structures, performance bottlenecks have appeared in these approaches, which makes us have to investigate the underlying reasons behind them.

As shown in Figure 2, we briefly visualized the ideas of these methods. From the point of view of convolution, the accuracy can be improved by 1) relaxing the pixel-level spatial invariance (e.g., Dilation/ Deformable CNNs), 2) fusing more local features (e.g., Multi-Column CNNs and Spatial Pyramid Pooling), and 3) exploiting Attention/ Perspective information. Inspired by this, we utilize a set of low-rank Gaussian kernels with the attention mechanism to relax spatial invariance and fuse local features by replacing standard convolutions. Here we only offer one solution, and follow-up work can continue to explore how to properly relax the spatial invariance.

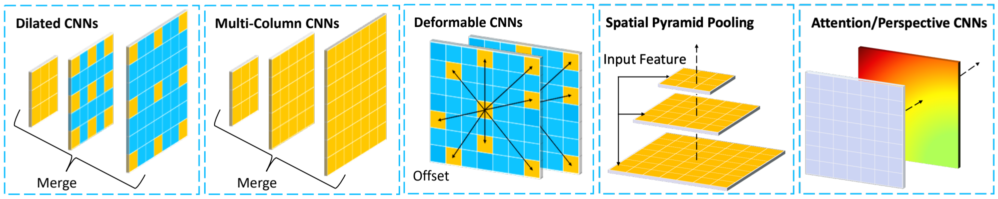

Figure 2. Overview of research directions of CNNs in object counting. 1) Dilated CNNs gradually increase the strides of convolution filters to adapt to different sizes. 2) Multi-Column CNNs utilize different filters to merge features in different branches. 3) Deformable CNNs optimize the shape of filters to handle multi-scale densities. 4) Spatial pyramid pooling performs pyramid scaling on input features. 5) Attention/perspective uses perspective/attention maps through feature extraction. [**Best viewed in color**].

2.2. Dealing with noise in the density map

Similar to our findings, some studies have also shown notable label noise in density maps [53, 59, 69, 75]. The mainstream approaches to overcome noise are to propose loss functions [6, 10, 30, 35, 42, 48, 60], optimize measurement metrics [30, 54], update matching rules [54, 61], fine-grained noise regions [1, 4, 36, 55], strengthen regular constraints [36, 45, 59, 68, 69], combine extra labels [7, 53, 64, 75, 80], and optimize training processes [1, 4, 32, 79]. Some recent studies have also started to use adversarial [46, 65, 81, 82] and reinforcement learning [33] to handle noise in the density learning.

In summary, these approaches do not reveal the correlation between the spatial invariance and the noise of density maps. Most of them only minimize noise by optimizing the loss or regularization term [23, 31, 42, 59, 68]. For example, a recent work called AutoScale [68] attempts to normalize the densities of different image regions to within a reasonable range. Our work is inspired by previous work [59]. Unlike it only focuses on optimizing the loss, our method attempts to modify the convolution operation to overcome noise during the feature learning.

3. Methods

To better understand our method, we first briefly review the traditional density map generation to reveal the labeling noises of the object counting task.

3.1. Traditional density map generation

The recent mainstream approach turns the object counting task into a density regression problem [27, 50, 63]. For N objects of image $\mathcal{I}$, the center points of all objects are labeled as $\left\{\tilde{\mathbf{D}}_1, ..., \tilde{\mathbf{D}}_i, ...\tilde{\mathbf{D}}_N\right\}$. The Gaussian kernel can effectively overcome the singularity in the prediction process. Thus the density of any pixel in an image, $\forall \mathbf{p}_i \in \mathcal{I}$, is generated by multiple Gaussian kernels as,

$$y\left(\mathbf{p}_i\right) = \sum_{i=1}^{N} \mathcal{N}\left(\mathbf{p}_i; \tilde{\mathbf{D}}_i, \beta\mathbf{I}\right) \tag{1}$$

$$= \sum_{i=1}^{N} \frac{1}{2\pi\beta} \exp(-\frac{1}{2}\left\|\mathbf{p}_i - \tilde{\mathbf{D}}_i\right\|^2_{\beta\mathbf{I}}), \tag{2}$$

where $\mathcal{N}(\tilde{\mathbf{D}}_\mathbf{i}, \beta\mathbf{I})$ is the multivariate Gaussian kernel, the mean $\tilde{\mathbf{D}}_\mathbf{i}$ and the covariance $\beta\mathbf{I}$ respectively depict the center point position and shape of the object. β is the variance of the Gaussian kernel and $\|\mathbf{x}\|^2_{\beta\mathbf{I}} = \mathbf{x}^T(\beta\mathbf{I})^{-1}\mathbf{x}$ is the square Mahalanobis distance.

3.2. Noise in object counting task

However, similar to the previous work [53, 59, 69, 75], we found that there are naturally two kinds of unavoidable noises in density map as Figure 1.

1. The error ϵ between the true position of the object $\mathbf{D}_\mathbf{i}$ and the labeled center point $\tilde{\mathbf{D}}_\mathbf{i}$;
2. The error $\mathbf{\Sigma}$ between object occlusion and overlapping of multiple Gaussian kernel approximation $\sum_{i=1}^{N} \mathcal{N}(\mathbf{p}_i; \tilde{\mathbf{D}}_i, \beta\mathbf{I})$;

Suppose the labeling error ϵ of the center point position is independent and identically distributed (i.i.d) and also obeys the Gaussian distribution. Similar to Eq. 1, the density map of any pixel $\forall \mathbf{p}_i \in \mathcal{I}$ with the true center point $\mathbf{D}_i = \tilde{\mathbf{D}}_i - \epsilon_i$ can also be computed as,

$$y\left(\mathbf{p}_i\right) = \sum_{i=1}^{N} \mathcal{N}\left(\mathbf{p}_i; \tilde{\mathbf{D}}_i - \epsilon_i, \beta\mathbf{I}\right) \tag{3}$$

$$= \sum_{i=1}^{N} \mathcal{N}\left(\mathbf{q}_i; \epsilon_i, \beta\mathbf{I}\right), \tag{4}$$

where we have made some equivalent changes to the equations. Further replacing $\mathbf{p}_i$ with $\mathbf{q}_i = \tilde{\mathbf{D}}_i - \mathbf{p}_i$, the density map is still as the combination of the Gaussian distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$. The values of mean μ and variance $\mathbf{\Sigma}$ are respectively estimated as,

$$\mu \approx \mathbb{E}\left[\sum_{i=1}^{N} \mathcal{N}\left(\epsilon_i, \beta\mathbf{I}\right)\right] \approx \sum_{i}^{N} \epsilon_i, \tag{5}$$

$$\mathbf{\Sigma} \approx \sum_{i=1}^{N} \frac{1}{2\pi\gamma} \mathcal{N}(\mathbf{0}, \delta\mathbf{I}) - \sum_{i=1}^{N} \mu_i^2, \tag{6}$$

where β, γ, δ are the variance parameters of the Gaussian function[3].

[3]We simply reformulate the parameter by $\gamma = 2\beta$ to make a concise expression. For more details, please refer to the previous work [59].

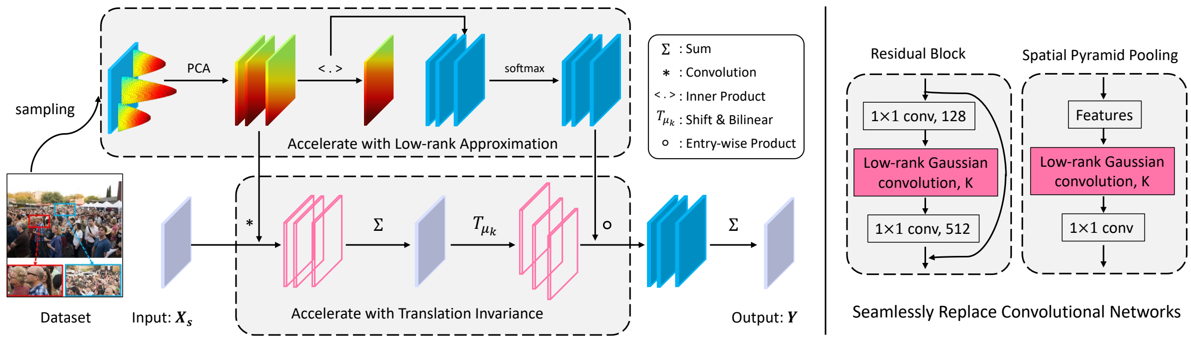

Figure 3. Illustration of Low-rank Gaussian convolutional layer. Our proposed layer mainly contains two acceleration modules. *Low-rank approximation module* has two steps: 1-Principal Component Analysis (PCA) is used to select the Gaussian kernels, 2-Inner product and softmax are used to get the fusion weights. *Translation Invariance module* also splits the Gaussian kernel operation into two steps: 1-Convolution with Gaussian kernels of zero means, 2-Translation result with other unique means. Our proposed layer can replace any standard convolutional layer, where the right part is two application examples of residual blocks and pyramid pooling. [**Best view in color**].

Although the updated density map still obeys a Gaussian distribution, according to Eq. 5 and 6, the mean μ (depicting the center point) and variance $\mathbf{\Sigma}$ (representing shape and occlusion) have more complex forms. This mathematically sheds light on why strict pixel-level spatial invariance leads to severe overfitting label errors. As shown in Sec. 4.2, some state-of-the-art networks still fail to predict actual occlusion in high-density regions, and overestimate the density in low-density regions. Obviously, this is due to overfitting to noise, thereby completely ignoring the position and shape of objects. Below we will present our solution.

3.3. Low-rank Gaussian convolutional layer

Inspired by the previous works [15, 19, 26, 56], we try to replace the standard convolution filters with Gaussian kernels. In this way, the feature extraction can simulate the process of density map generation. After pixel-grid filters are replaced with Gaussian kernels, we can jump out of the strict pixel-level spatial constraints and learn the density map in a more relaxed spatial manner. The modified convolution is as,

$$\mathbf{Y}_s = \sum_{i=0}^{N} G(\mu_i, \mathbf{\Sigma}_i) * \mathbf{X}_s + \mathbf{b}_s, \tag{7}$$

where $*$ and $\mathbf{b}_s$ are convolution operation and offsets. $\mathbf{X}_s$ and $\mathbf{Y}$ are two-dimensional features. Here we only take the features of channel s as an example. Since we want to simulate the density map generation, all N Gaussian kernels $G(\mu_i, \Sigma_i)$ have to be used for convolution. The position and shape of the objects are respectively stipulated by the mean μ_i and the variation $\mathbf{\Sigma}_i$.

However, Eq. 7 cannot be implemented because it requires massive Gaussian convolutions. Fortunately, previous work [59] uses low-rank Gaussian distributions to approximate the density map. Inspired by this, we proposed a low-rank approximation module (Sec. 3.3.1) to achieve the approximation to Gaussian convolution, and accordingly equipped a translation invariance module (Sec. 3.3.2) to accelerate computation. As shown in Figure 3, we will present these two modules below.

3.3.1 Accelerate with Low-rank approximation

Low-rank approximation module uses a small number of Gaussian kernels with the low-rank connection to approximate an almost infinite Gaussian convolution (Eq. 7). It has been proven [59] that a density map generated by aggregating N Gaussian kernels (N can be hundreds to thousands[4]) could be approximated by K Gaussian kernels $\{G_1(\mathbf{\Sigma}_1), ..., G_K(\mathbf{\Sigma}_K)\}$, where $K \ll N$. Although previous work [59] uses the low-rank approximation to optimize the density map in the loss function, it is still unclear how to approximate the massive Gaussian convolution.

To this end, we try to approximate the nite Gaussian convolution by learning a few Gaussian kernels, as well as their correlations with an attention mechanism. During the approximation, $2K$ Gaussian kernels are randomly sampled. After the Principal Component Analysis (PCA), the eigenvectors $\{G(\mathbf{\Sigma}_k)\}_{k=1}^{K}$ corresponding to K non-zero eigenvalues are obtained. Then we initialize the coefficients of picked K Gaussian kernels as,

$$\mathbf{w}_k = \langle G(\mathbf{\Sigma}_k), G(\mathbf{\Sigma}_I) \rangle \,, \tag{8}$$

where $< . >$ is the inner product, and $\mathbf{\Sigma}_I$ represents the identity matrix. Because we will further decompose the Gaussian kernel to speed up the computation, the mean μ of the Gaussian kernels is ignored here. Finally, we perform normalization operations,

$$\sigma(\mathbf{w}_k) = \frac{\exp(\mathbf{w}_k)}{\sum_{l=1}^{K} \exp(\mathbf{w}_l)}, \tag{9}$$

where $\mathbf{w}_k$ are also updated during training. In addition to fusing the local features, it can also help restrict the spatial information in the gradient back-propagation.

[4] N is the number of objects in image as shown in Table 1

Based on this improvement, the optimized Gaussian convolutional layer is computer as,

$$\mathbf{Y}_s = \sum_{k=0}^{K}\sum_{j=0}^{K}(\mathbf{w}_k \circ \sum_{i=0}^{K}(G(\mu_i, \mathbf{\Sigma}_j) * \mathbf{X}_s)) + \mathbf{b}_s, \quad (10)$$

where $\circ$ is the entry-wise product. We utilize the low-rank Gaussian kernels to complete the approximation process. Following we will continue to apply the translation invariance module to further optimize our method.

3.3.2 Accelerate with translation invariance

Translation invariance module aims to decompose the convolution operation between the Gaussian kernel and the input feature map to accelerate the inference. Accomplishing convolution operations of all Gaussian kernels in Eq. 10 requires a lot of computational resources. Using the translation invariance of Gaussian kernel, the convolution operation between the Gaussian kernel and the input features can be efficiently implemented as,

$$G(\mu_k, \mathbf{\Sigma}_k) * \mathbf{x} = \mathcal{T}_{\mu_k}[G(\mathbf{0}, \mathbf{\Sigma}_k)] * \mathbf{x} \quad (11)$$

$$= \mathcal{T}_{\mu_k}[G(\mathbf{0}, \mathbf{\Sigma}_k) * \mathbf{x}], \quad (12)$$

where $\mathcal{T}_{\mu_k}[\mathbf{y}] = g(\mathbf{y} - \mu_k)$ is the translation of the function $g()$. $G(\mathbf{0}, \mathbf{\Sigma}_k)$ is Gaussian kernels with zero mean. The benefit of this is that we can ignore the mean of Gaussian kernels in the convolution operation. Since Eq. 12 is only accurate for discrete μ_k, we treat the translation function $g()$ as bilinear interpolation in the actual implementation,

$$\tilde{\mathcal{T}}_{\mu_k}[\mathbf{y}] = \sum_i \sum_j \mathbf{a}_{ij} \cdot g(\mathbf{y} - \lfloor \mu_k \rfloor + \lfloor i, j \rfloor), \quad (13)$$

where $\mathbf{a}_{ij}$ are the weights in bilinear interpolation, which allow computing subpixel displacements and can be implemented efficiently in CUDA.

Finally, our proposed low-rank Gaussian convolutional layer can be computed as,

$$\mathbf{Y}_s = \sum_{k=0}^{K}\sum_{j=0}^{K}(\mathbf{w}_k \circ \tilde{\mathcal{T}}_{\mu_k}[G(\mathbf{\Sigma}_j) * \mathbf{X}_s]) + \mathbf{b}_s, \quad (14)$$

where all implementations are based on CUDA. Thus our proposed layer can be applied to mainstream CNNs. In most cases, we replace all the convolutional layers (or 3×3 convolutional layers in all residual and pyramid pooling blocks) with our Gaussian convolutional layers.

Complexity Analysis. Theoretically, considering input $\mathbf{X} = [H, W, C_i]$ and output $\mathbf{Y} = [H, W, C_o]$, supposing N Gaussian kernels are used in density map generation, the complexity of the initial Gaussian convolution (Eq. 7) is $\mathcal{O}(C_iC_oHWNk_wk_h)$, where k_w, k_h indicate the upper bound of the size of Gaussian kernels. When utilizing low-rank approximation, the complexity of Eq. 10 is $\mathcal{O}(KC_iC_oHWKk_wk_h)$, where K is the number of the sampled kernels, $K \ll N$. By further applying translation invariance, the complexity of Eq. 14 is $\mathcal{O}(4KC_iC_oHW)$, where 4 is related to the bilinear interpolation. Table 2 also shows experimental time cost of our method, which demonstrates the effectiveness of two acceleration components.

Table 1. Object counting benchmarks. [Min, Max] and #images are the range of objects per image and the number of images.

	Datasets	[Min, Max]	#Images
Crowd	SHTech-PartA [76]	[33, 3,139]	482
	UCF_CC_50 [22]	[94, 4,543]	50
	UCF-QNRF [24]	[49, 12,865]	1,525
	JHU-CROWD++ [49]	[0, 7,286]	4,250
	SHTech-PartB [76]	[9, 578]	716
Vehicle	TRANCOS [16]	[9, 107]	1,244
Plant	MTC [37]	[0, 100]	361

4. Experiments

4.1. Experiment settings

Dataset. We evaluate our method on three application, i.e., crowd, vehicle, and plant counting. For crowd counting, five datasets are used for evaluation, including ShanghaiTech (SHTech) PartA and PartB [76], UCF_CC_50 [22], UCF-QNRF [24] and JHU-CROWD++ [49]. For vehicle and plant counting, two datasets, i.e., TRANCOS [16] and MTC [37]) are used, respectively. Table 1 gives a summary of these datasets.

Baseline Networks. We evaluate our method by integrating it with four baselines including MCNN [77], CSRNet [28], SANet [4], and ResNet-50 [18]. The training procedures follow third-party Github repositories[5]. Training details are slightly different from the original paper. For example, batch processing and other functions are included. Following previous works [10,44,61], MCNN and CSRNet are tested on the whole images, while SANet is evaluated on image patches. Additionally, Mean Absolute Error (MAE) and Mean Square Error (MSE) are used as evaluation measurements.

4.2. Reveal the label noise of object counting

We verified the prediction variance on four mainstream object counting methods (i.e., MCNN [77], CSRNet [28], SANet [4], and ResNet-50 [18]).

Large variance in prediction. As shown in Figure 5, four object counting methods have a large prediction variance on SHTech-PartA and UCF-QNRF datasets. Even more surprising is that the variance does not decrease as the performance (spatial invariance) increases. The results in Figure 5 meaningfully reveal its hidden reason, namely that the overly strict pixel-level spatial invariance makes the model severely overfit to the density map noise.

[5] https://github.com/gjy3035/C-3-Framework

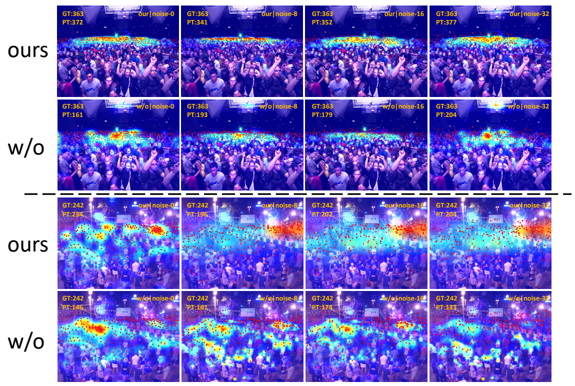

Figure 4. Visualization of robustness to annotation noise, where red dots are groundtruth annotations. Here we generate the noisy dataset by randomly moving the annotation points by {0, 8, 16, 32} pixels. Visualization results exhibit the results of two examples with/without our proposed Gaussian convolutional layer.

Table 2. Cost on MCNN (batchsize 1, image size 256^2). LRA and TI refer to Low-Rank Approximation and Translation Invariance. The Vanilla setting uses 256 Gaussian kernels per layer.

Time (milliseconds)	Vanilla	LRA	LRA+TI
Forward	51.3	13.3	4.1
Backward	160.0	44.1	12.6

Underestimation of high-density areas. We performed a similar validation for high-density regions to find out the reasons for the large prediction variance. In the second column of Figure 5, we noticed that the prediction variance in high-density areas is more severe than the entire image. The overall statistics prove that the model severely underestimates density in high-density areas. What is even more surprising is that this variance appears to increase as the performance (spatial invariance) increases.

Overestimated in low density areas. Likewise, in the third column, we analyzed the low-density area. Overall, the variance is reduced compared to high-density areas. We speculate that fewer Gaussian kernels are in the low-density area, which inherently has lower annotation noise. Although the variance is slight than the high-density area, the overall variance is still more severe than the entire image. We guess this is because the high and low-density areas compensate each other to reduce the variance.

Ignorance of position and shape. To further clarify the large prediction variance, we visualized some examples. Figure 4 shows the obvious difference between the predicted density maps and the true position of the object (indicated by the red dot). In some low-density areas, the prediction results ignore many objects (i.e., the density map does not cover many red dots). Likewise, in some high-density regions, the crowd is poorly estimated (that is, the clustering on the density map is inconsistent with the trend of the red dots). To sum up, these visualizations show that blindly improving spatial invariance does not learn the location and shape of objects.

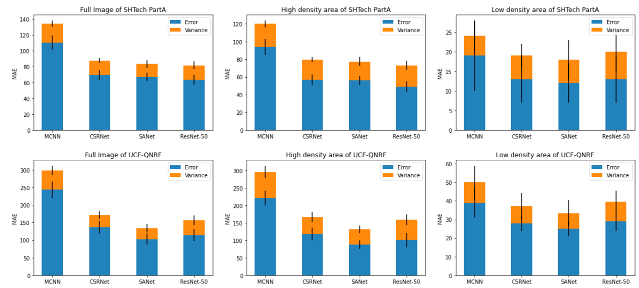

Figure 5. Comparative analysis of the prediction variance. The variance refers to the difference in the results at different convergence states. The error refers to the difference between the prediction and the groundtruth. Left to right are the analysis results of the full image, high-density area, and low-density area. The results clearly show that there is a huge variance in prediction results.

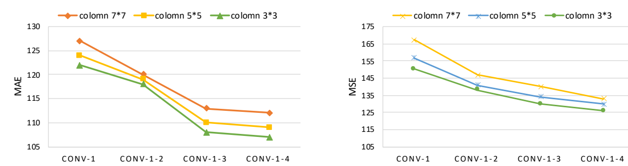

Figure 6. The ablation study of MCNN [77]. The number after CONV indicates the range of using our proposed layer.

4.3. Ablation study

We perform ablation studies with our method. Due to space limitations, we only uses MCNN [77] as an example.

Effectiveness of accelerated modules. We conduct ablation studies to verify the effectiveness of low-rank approximation and translation invariance modules. Table 2 shows the experimental time cost of our proposed layer. Compared with the original Gaussian convolution, our offered two acceleration modules can significantly improve the computational efficiency.

Where should it be replaced? As shown in Figure 6, we performed ablation studies on the three column convolutional structures of MCNN. Overall, the three column structures have roughly the same results. We noticed that replacing our layers in the first three convolutional layers will achieve larger improvements. We also got similar results in other baselines. Our method has fewer parameters than the original convolutional layer. Thus in most cases, we replace all the convolutional layers (or 3×3 convolutional layers in all residual blocks and pyramid pooling blocks) with our Gaussian convolutional layers.

How to set the Gaussian kernels? Our method has three hyperparameters, i.e., the mean μ, variance σ, and the number of Gaussian kernels K. The mean value can be instantly set according to the stride of the original convolutional layer. Thus we will only discuss variance $\mathbf{\Sigma}$ and the number of K in the experiment.

As shown in Figure 8, we have carried out studies on the three column structures of MCNN. When the value of K is

Table 3. Comparison with the state-of-the-art methods on SHTech-PartA [76], UCF_CC_50 [22], UCF-QNRF [24] and JHU-CROWD++ [49] datasets. The best results are shown in bold. This also applies to the following tables.

Methods	Venue	SHTech-PartA		UCF_CC_50		UCF-QNRF		JHU-CROWD++	
		MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ADSCNet [3]	CVPR'20	55.4	97.7	198.4	267.3	**71.3**	**132.5**	-	-
AMSNet [20]	ECCV'20	56.7	93.4	208.4	297.3	101.8	163.2	-	-
MNA [59]	NeurIPS'20	61.9	99.6	-	-	85.8	150.6	67.7	258.5
DM-Count [61]	NeurIPS'20	59.7	95.7	211.0	291.5	85.6	148.3	66.0	261.4
GLoss [60]	CVPR'21	61.3	95.4	-	-	84.3	147.5	59.9	259.5
URC [69]	ICCV'21	72.8	111.6	294.0	443.1	128.1	218.1	129.7	400.5
SDNet [41]	ICCV'21	55.0	92.7	197.5	264.1	80.7	146.3	59.3	248.9
MCNN [77]	CVPR'16	110.2	173.2	377.6	509.1	277.0	426.0	188.9	483.4
CSRNet [28]	CVPR'18	68.2	115.0	266.1	397.5	119.2	211.4	85.9	309.2
SANet [4]	ECCV'18	67.0	104.5	258.4	334.9	-	-	91.1	320.4
MCNN (ours)		94.2	141.8	282.6	387.2	204.2	280.4	165.3	486.6
CSRNet (ours)		61.2	97.8	215.4	296.4	84.2	152.4	69.4	262.4
SANet (ours)		59.2	95.4	209.2	278.4	86.6	162.8	68.9	270.6
ResNet-50 (ours)		**54.8**	**89.1**	**186.3**	**256.5**	81.6	153.7	**58.2**	**245.1**

Table 4. Results on SHTech-PartB [76] dataset.

Methods	Venue	SHTech-PartB	
		MAE	MSE
ADSCNet [3]	CVPR'20	6.4	11.3
AMSNet [20]	ECCV'20	6.7	10.2
DM-Count [61]	NeurIPS'20	7.4	11.8
GLoss [60]	CVPR'21	7.3	11.7
URC [69]	ICCV'21	12.0	18.7
MCNN [77]	CVPR'16	26.4	41.3
CSRNet [28]	CVPR'18	10.6	16.0
SANet [4]	ECCV'18	8.4	13.2
MCNN (ours)		17.6	24.7
CSRNet (ours)		7.6	12.7
SANet (ours)		7.1	11.2
ResNet-50 (ours)		**6.2**	**9.9**

Table 5. Results on TRANCOS [16] and MTC [37] dataset.

Methods	TRANCOS		MTC	
	MAE	MSE	MAE	MSE
ADMG [58]	2.6	3.89	-	-
TasselNetv2 [66]	-	-	5.4	8.8
S-DCNet [67]	-	-	5.6	9.1
CSRNet [28]	3.56	-	9.4	14.4
CSRNet(ours)	2.2	**2.6**	3.2	4.6
MCNN (ours)	7.7	7.4	8.7	12.3
SANet (ours)	2.5	2.8	3.4	4.5
ResNet (ours)	**2.1**	**2.6**	**3.1**	**4.3**

large enough ($K = 100$), we estimate the change of Gaussian kernel variance in each convolution layer. We observe that the variance merely changed in the first convolutional layer. Inspired by this, we usually set K to 16 with variance from $[-0.5, 0.5]$ in the first two convolutional layers, and set K to 2 or 4 in the successive convolutional layers with variance from $[-0.1, 0.1]$).

4.4. Compare with state-of-the-art methods

We compared our method with state-of-the-art methods in three applications (crowd, vehicle, plant counting).

Table 6. Robustness to annotation noise. Both [59] and CSRNet adopt VGG backbone. Results of VGG are from Figure 4 of [59].

MAE (↓)	0	4	8	16	32
CSRNet (w/o)	119.2	125.4	133.7	142.5	166.2
VGG [59]	85.8	91.0	96.0	97.0	99.0
CSRNet (ours)	84.2	85.7	89.0	92.2	95.4

Result of crowd counting. Table 3 shows the results of crowd counting in the free camera perspective. We considered the prediction variance and selected the worst result for reporting. Except for MCNN, the other three modified baselines outperform other state-of-the-art methods. Compared to the original baselines, our variant has also achieved a huge improvement. The performance of the light MCNN is even close to some of the most advanced methods.

Table 4 shows the results in the surveillance scenarios. Like free views, our model surpasses other state-of-the-art approaches, but the improvement in surveillance scenarios is not as much as free perspective. We guess there is more noise in generating density maps in the free view. Due to the noisy label in the groundtruth of SHTech-PartB, our method cannot further improve performance.

Result of object counting. We also evaluated vehicle and plant counting. Table 5 shows that our model works well for vehicle scenarios. The improvement is minor compared to the crowd counting because the vehicle scene holds less noise. For plant counting, we got similar results. Our model outperforms other state-of-the-art methods. Notable is the improvement in the MSE metric, which shows that our method is more robust. The overall performance is very close to the groundtruth.

4.5. Robustness to annotation noise

We follow previous work [59] to verify robustness to annotation noise. We generate a noisy dataset by randomly moving the annotation points by $\{4, 8, 16, 32\}$ pixels. Then we train the model on noisy datasets with or without our

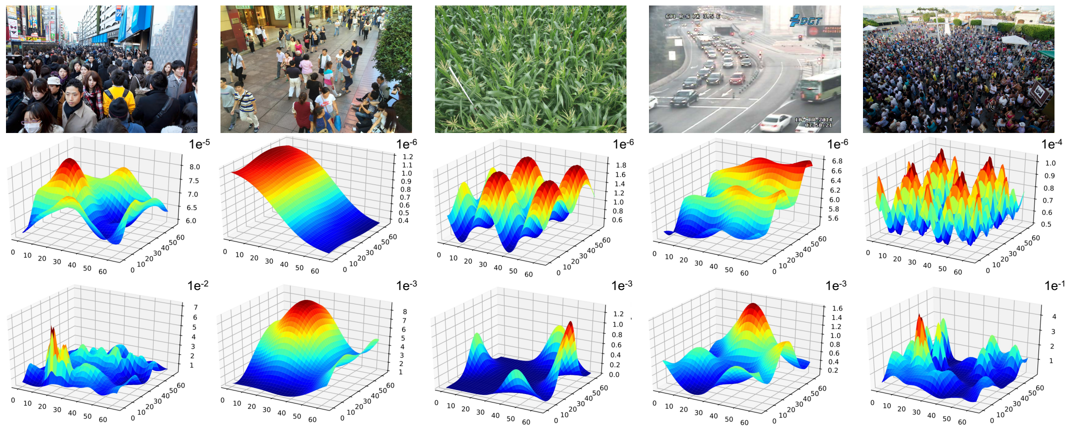

Figure 7. The visualization of the convolution filters. From left to right are the results from SHTech-PartA, SHTech-PartB, MTC, TRANCOS and UCF-QNRF. From top to bottom are example images, our revised SANet and the original SANet [4]. Intuitively, our approach can fully understand the spatial information of objects and the perspective law of views. More details are shown in Sec. 4.6.

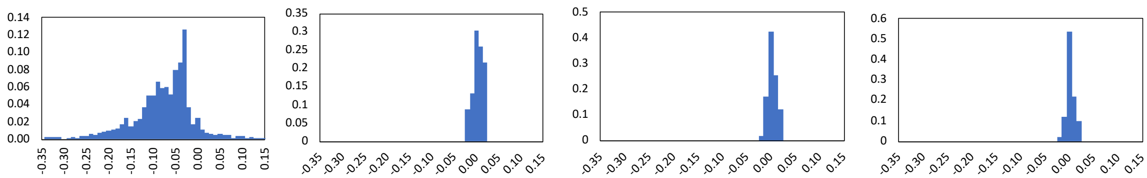

Figure 8. Ablation study on MCNN [77]. From left to right are the variance optimization changes in the 1-st to 4-th convolutional layers. The abscissa indicates the range of change, and the ordinate indicates the intensity of the change.

proposed Gaussian convolutions. Table 6 shows the comparison. Though the performances of all methods decrease as the annotation noise increases, our method is still more robust than other methods. Figure 4 also illustrates the predicted results of two examples with/without our method.

4.6. Visualization of convolution filters

We visualized the convolution filters to evaluate whether our model can simulate the density map generation and learn the spatial information of the objects. Figure 7 shows the results after visualization. In general, our method can effectively learn the perspective law of the distribution of objects. The results in the plant counting (column 3) are particularly obvious due to the more consistent scenarios. Our method learns the planting distribution and even reflects the planting interval. In contrast, the original SANet [4] only shows some noise in the image (e.g., marking Poles). Similarly, our method also learns the distribution of pedestrians and vehicles by counting pedestrians and vehicles under the surveillance viewing angle (columns 2 and 4). On the contrary, the original SANet blindly guesses high-density areas or overestimates low-density regions. We also found similar results under the free perspective (columns 1 and 5), where our method can approximate crowd density distribution in pedestrian streets and squares.

5. Conclusion

We reveal the relationship between spatial invariance and density map noise. Extensive experiments prove that if only instinctively improve the spatial invariance of CNNs, the model will easily overfit to the density map noise. Inspired by this, we utilize a set of locally connected multivariate Gaussian kernels for replacing the convolution filters. Unlike the pixelized-based filter, our proposed variant can approximately simulate the process of density map generation. Considering the characteristics of object counting, we try to use translation invariance and low-rank approximation to improve the efficiency. Extensive experiments show that our method outperforms other state-of-the-art methods. Our work points out the direction for future research. It can avoid wildly improving the spatial invariance for the object counting. In the future, we will further analyze the relationship between the Gaussian kernel and spatial invariance.

Acknowledgements. This work was partially supported by the Air Force Research Laboratory under agreement number FA8750-19-2-0200; the financial assistance award 60NANB17D156 from U.S. Department of Commerce, National Institute of Standards and Technology (NIST); the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00340; the Defense Advanced Research Projects Agency (DARPA) grant funded under the GAILA program (award HR00111990063). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References

[1] Shahira Abousamra, Minh Hoai, Dimitris Samaras, and Chao Chen. Localization in the crowd with topological constraints. In *AAAI*, 2021. 3

[2] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *ECCV*, pages 504–518, 2014. 2

[3] Shuai Bai, Zhiqun He, Yu Qiao, Hanzhe Hu, Wei Wu, and Junjie Yan. Adaptive dilated network with self-correction supervision for counting. In *CVPR*, pages 4594–4603, 2020. 1, 2, 7

[4] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *ECCV*, pages 734–750, 2018. 2, 3, 5, 7, 8

[5] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, pages 1–7, 2008. 2

[6] Chen Change Loy, Shaogang Gong, and Tao Xiang. From semi-supervised to transfer counting of crowds. In *ICCV*, pages 2256–2263, 2013. 1, 3

[7] Binghui Chen, Zhaoyi Yan, Ke Li, Pengyu Li, Biao Wang, Wangmeng Zuo, and Lei Zhang. Variational attention: Propagating domain-specific knowledge for multi-domain learning in crowd counting. In *ICCV*, pages 16065–16075, 2021. 3

[8] Ke Chen, Shaogang Gong, Tao Xiang, and Chen Change Loy. Cumulative attribute space for age and crowd density estimation. In *CVPR*, pages 2467–2474, 2013. 1, 2

[9] Xinya Chen, Yanrui Bin, Nong Sang, and Changxin Gao. Scale pyramid network for crowd counting. In *WACV*, pages 1941–1950, 2019. 2

[10] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, and Alexander G Hauptmann. Learning spatial awareness to improve crowd counting. In *ICCV*, pages 6152–6161, 2019. 1, 3, 5

[11] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, Jun-Yan He, and Alexander G Hauptmann. Improving the learning of multi-column convolutional neural network for crowd counting. In *ACM MM*, pages 1897–1906, 2019. 1, 2

[12] Zhi-Qi Cheng, Xiao Wu, Siyu Huang, Jun-Xiu Li, Alexander G Hauptmann, and Qiang Peng. Learning to transfer: Generalizable attribute learning with multitask neural model search. In *ACM MM*, pages 90–98, 2018. 2

[13] Diptodip Deb and Jonathan Ventura. An aggregated multicolumn dilated convolution network for perspective-free counting. In *CVPR Workshops*, pages 195–204, 2018. 1, 2

[14] Haoran Duan, Shidong Wang, and Yu Guan. Sofa-net: Second-order and first-order attention network for crowd counting. *arXiv*, 2020. 2

[15] Gamaleldin Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. In *ICML*, pages 2868–2879, 2020. 2, 4

[16] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *ICPRIA*, pages 423–431, 2015. 5, 7

[17] Dan Guo, Kun Li, Zheng-Jun Zha, and Meng Wang. Dadnet: Dilated-attention-deformable convnet for crowd counting. In *ACM MM*, pages 1823–1832, 2019. 1, 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2, 5

[19] Zhezhi He and Deliang Fan. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *CVPR*, June 2019. 2, 4

[20] Yutao Hu, Xiaolong Jiang, Xuhui Liu, Baochang Zhang, Jungong Han, Xianbin Cao, and David Doermann. Nas-count: Counting-by-density with neural architecture search. In *ECCV*, 2020. 7

[21] Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, and Alexander Hauptmann. Stacked pooling for boosting scale invariance of crowd counting. In *ICASSP*, pages 2578–2582, 2020. 2

[22] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *CVPR*, pages 2547–2554, 2013. 5, 7

[23] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, pages 532–546, 2018. 3

[24] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, pages 532–546, 2018. 5, 7

[25] Xiaoheng Jiang, Li Zhang, Mingliang Xu, Tianzhu Zhang, Pei Lv, Bing Zhou, Xin Yang, and Yanwei Pang. Attention scaling for crowd counting. In *CVPR*, pages 4706–4715, 2020. 2

[26] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds. *IEEE TPAMI*, 2020. 2, 4

[27] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *NeurIPS*, 23:1324–1332, 2010. 2, 3

[28] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, pages 1091–1100, 2018. 1, 2, 5, 7

[29] Dongze Lian, Jing Li, Jia Zheng, Weixin Luo, and Shenghua Gao. Density map regression guided detection network for rgb-d crowd counting and localization. In *CVPR*, pages 1821–1830, 2019. 2

[30] Hui Lin, Xiaopeng Hong, Zhiheng Ma, Xing Wei, Yunfeng Qiu, Yaowei Wang, and Yihong Gong. Direct measure matching for crowd counting. *arXiv*, 2021. 1, 3

[31] Bo Liu and Nuno Vasconcelos. Bayesian model adaptation for crowd counts. In *ICCV*, pages 4175–4183, 2015. 3

[32] Lingbo Liu, Jiaqi Chen, Hefeng Wu, Guanbin Li, Chenglong Li, and Liang Lin. Cross-modal collaborative representation learning and a large-scale rgbt benchmark for crowd counting. In *CVPR*, 2021. 3

[33] Liang Liu, Hao Lu, Hongwei Zou, Haipeng Xiong, Zhiguo Cao, and Chunhua Shen. Weighing counts: Sequential crowd counting by reinforcement learning. In *ECCV*, pages 164–181, 2020. 3
[34] Ning Liu, Yongchao Long, Changqing Zou, Qun Niu, Li Pan, and Hefeng Wu. Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding. In *CVPR*, pages 3225–3234, 2019. 1, 2
[35] Xinyan Liu, Guorong Li, Zhenjun Han, Weigang Zhang, Yifan Yang, Qingming Huang, and Nicu Sebe. Exploiting sample correlation for crowd counting with multi-expert network. In *ICCV*, pages 3215–3224, 2021. 3
[36] Xiyang Liu, Jie Yang, and Wenrui Ding. Adaptive mixture regression network with local counting map for crowd counting. In *ECCV*, 2020. 3
[37] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen. Tasselnet: counting maize tassels in the wild via local counts regression network. *Plant methods*, 13(1):1–17, 2017. 5, 7
[38] Ao Luo, Fan Yang, Xin Li, Dong Nie, Zhicheng Jiao, Shangchen Zhou, and Hong Cheng. Hybrid graph neural networks for crowd counting. In *AAAI*, volume 34, pages 11693–11700, 2020. 2
[39] Yu-Jen Ma, Hong-Han Shuai, and Wen-Huang Cheng. Spatiotemporal dilated convolution with uncertain matching for video-based crowd estimation. *IEEE TMM*, 2021. 1, 2
[40] Zheng Ma and Antoni B Chan. Crossing the line: Crowd counting by integer programming with local features. In *CVPR*, pages 2539–2546, 2013. 2
[41] Zhiheng Ma, Xiaopeng Hong, Xing Wei, Yunfeng Qiu, and Yihong Gong. Towards a universal model for cross-dataset crowd counting. In *ICCV*, pages 3205–3214, 2021. 7
[42] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Bayesian loss for crowd count estimation with point supervision. In *ICCV*, pages 6142–6151, 2019. 3
[43] Mark Marsden, Kevin McGuinness, Suzanne Little, and Noel E O'Connor. Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification. In *AVSS*, pages 1–7, 2017. 2
[44] Yunqi Miao, Zijia Lin, Guiguang Ding, and Jungong Han. Shallow feature based dense attention network for crowd counting. In *AAAI*, volume 34, pages 11765–11772, 2020. 2, 5
[45] Min-hwan Oh, Peder Olsen, and Karthikeyan Natesan Ramamurthy. Crowd counting with decomposed uncertainty. In *AAAI*, volume 34, pages 11799–11806, 2020. 3
[46] Greg Olmschenk, Jin Chen, Hao Tang, and Zhigang Zhu. Dense crowd counting convolutional neural networks with minimal data using semi-supervised dual-goal generative adversarial networks. In *CVPR Workshop*, 2019. 3
[47] Mikel Rodriguez, Ivan Laptev, Josef Sivic, and Jean-Yves Audibert. Density-aware person detection and tracking in crowds. In *ICCV*, pages 2423–2430, 2011. 2
[48] Sravya Vardhani Shivapuja, Mansi Pradeep Khamkar, Divij Bajaj, Ganesh Ramakrishnan, and Ravi Kiran Sarvadevabhatla. Wisdom of (binned) crowds: A bayesian stratification paradigm for crowd counting. In *ACM MM*, pages 3574–3582, 2021. 3
[49] Vishwanath Sindagi, Rajeev Yasarla, and Vishal MM Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *IEEE TPAMI*, 2020. 5, 7
[50] Vishwanath A Sindagi and Vishal M Patel. Generating high-quality crowd density maps using contextual pyramid cnns. In *ICCV*, pages 1861–1870, 2017. 2, 3
[51] Vishwanath A Sindagi and Vishal M Patel. Ha-ccn: Hierarchical attention-based crowd counting network. *IEEE TIP*, 29:323–335, 2019. 2
[52] Vishwanath A Sindagi and Vishal M Patel. Inverse attention guided deep crowd counting network. In *AVSS*, pages 1–8, 2019. 2
[53] Vishwanath A Sindagi, Rajeev Yasarla, Deepak Sam Babu, R Venkatesh Babu, and Vishal M Patel. Learning to count in the crowd from limited labeled data. In *ECCV*, 2020. 3
[54] Qingyu Song, Changan Wang, Zhengkai Jiang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yang Wu. Rethinking counting and localization in crowds: A purely point-based framework. In *ICCV*, pages 3365–3374, 2021. 3
[55] Qingyu Song, Changan Wang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Jian Wu, and Jiayi Ma. To choose or to fuse? scale selection for crowd counting. In *AAAI*, volume 35, pages 2576–2583, 2021. 3
[56] Yu-Chuan Su and Kristen Grauman. Kernel transformer networks for compact spherical convolution. In *CVPR*, June 2019. 4
[57] Domen Tabernik, Matej Kristan, and Aleš Leonardis. Spatially-adaptive filter units for compact and efficient deep neural networks. *IJCV*, 128(8):2049–2067, 2020. 2
[58] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *ICCV*, pages 1130–1139, 2019. 7
[59] Jia Wan and Antoni Chan. Modeling noisy annotations for crowd counting. *NeurIPS*, 33, 2020. 1, 2, 3, 4, 7
[60] Jia Wan, Ziquan Liu, and Antoni B Chan. A generalized loss function for crowd counting and localization. In *CVPR*, pages 1974–1983, 2021. 3, 7
[61] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. *arXiv*, 2020. 3, 5, 7
[62] Chuan Wang, Hua Zhang, Liang Yang, Si Liu, and Xiaochun Cao. Deep people counting in extremely dense crowds. In *ACM MM*, pages 1299–1302, 2015. 1, 2
[63] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *CVPR*, pages 8198–8207, 2019. 3
[64] Qi Wang, Tao Han, Junyu Gao, and Yuan Yuan. Neuron linear transformation: Modeling the domain shift for crowd counting. *IEEE TNNLS*, 2021. 3
[65] Qiming Wu, Zhikang Zou, Pan Zhou, Xiaoqing Ye, Binghui Wang, and Ang Li. Towards adversarial patch analysis and certified defense against crowd counting. *arXiv*, 2021. 3
[66] Haipeng Xiong, Zhiguo Cao, Hao Lu, Simon Madec, Liang Liu, and Chunhua Shen. Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. *Plant Methods*, 15(1):1–14, 2019. 7

[67] Haipeng Xiong, Hao Lu, Chengxin Liu, Liang Liu, Zhiguo Cao, and Chunhua Shen. From open set to closed set: Counting objects by spatial divide-and-conquer. In *ICCV*, pages 8362–8371, 2019. 7

[68] Chenfeng Xu, Dingkang Liang, Yongchao Xu, Song Bai, Wei Zhan, Xiang Bai, and Masayoshi Tomizuka. Autoscale: learning to scale for crowd counting. *arXiv*, pages arXiv–1912, 2019. 3

[69] Yanyu Xu, Ziming Zhong, Dongze Lian, Jing Li, Zhengxin Li, Xinxing Xu, and Shenghua Gao. Crowd counting with partial annotations in an image. In *ICCV*, pages 15570–15579, 2021. 3, 7

[70] Zhaoyi Yan, Ruimao Zhang, Hongzhi Zhang, Qingfu Zhang, and Wangmeng Zuo. Crowd counting via perspective-guided fractional-dilation convolution. *IEEE TMM*, 2021. 2

[71] Yifan Yang, Guorong Li, Dawei Du, Qingming Huang, and Nicu Sebe. Embedding perspective analysis into multi-column convolutional neural network for crowd counting. *IEEE TIP*, 30:1395–1407, 2020. 1, 2

[72] Anran Zhang, Jiayi Shen, Zehao Xiao, Fan Zhu, Xiantong Zhen, Xianbin Cao, and Ling Shao. Relational attention network for crowd counting. In *ICCV*, pages 6788–6797, 2019. 2

[73] Anran Zhang, Lei Yue, Jiayi Shen, Fan Zhu, Xiantong Zhen, Xianbin Cao, and Ling Shao. Attentional neural fields for crowd counting. In *ICCV*, pages 5714–5723, 2019. 2

[74] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, pages 833–841, 2015. 2

[75] Qi Zhang, Wei Lin, and Antoni B Chan. Cross-view cross-scene multi-view crowd counting. In *CVPR*, pages 557–567, 2021. 3

[76] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, pages 589–597, 2016. 1, 5, 7

[77] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, pages 589–597, 2016. 2, 5, 6, 7, 8

[78] Kun Zhao, Bin Liu, Luchuan Song, Weihai Li, and Nenghai Yu. Cascaded residual density network for crowd counting. In *ICIP*, pages 2199–2203, 2019. 2

[79] Zhen Zhao, Miaojing Shi, Xiaoxiao Zhao, and Li Li. Active crowd counting with limited supervision. In *ECCV*. Springer, 2020. 3

[80] Joey Tianyi Zhou, Le Zhang, Du Jiawei, Xi Peng, Zhiwen Fang, Zhe Xiao, and Hongyuan Zhu. Locality-aware crowd counting. *IEEE TPAMI*, 2021. 3

[81] Yuan Zhou, Jianxing Yang, Hongru Li, Tao Cao, and Sun-Yuan Kung. Adversarial learning for multiscale crowd counting under complex scenes. *IEEE Trans Cybern*, 2020. 3

[82] Zhikang Zou, Xiaoye Qu, Pan Zhou, Shuangjie Xu, Xiaoqing Ye, Wenhao Wu, and Jin Ye. Coarse to fine: Domain adaptive crowd counting via adversarial scoring network. In *ACM MM*, pages 2185–2194, 2021. 3