

# Crowd Counting on Images with Scale Variation and Isolated Clusters

Haoyue Bai Song Wen S.-H. Gary Chan

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology, Hong Kong, China

{hbaiaa, swenaa, gchan}@cse.ust.hk

## Abstract

*Crowd counting is to estimate the number of objects (e.g., people or vehicles) in an image of unconstrained congested scenes. Designing a general crowd counting algorithm applicable to a wide range of crowd images is challenging, mainly due to the possibly large variation in object scales and the presence of many isolated small clusters. Previous approaches based on convolution operations with multi-branch architecture are effective for only some narrow bands of scales, and have not captured the long-range contextual relationship due to isolated clustering. To address that, we propose SACANet, a novel scale-adaptive long-range context-aware network for crowd counting.*

*SACANet consists of three major modules: the pyramid contextual module which extracts long-range contextual information and enlarges the receptive field, a scale-adaptive self-attention multi-branch module to attain high scale sensitivity and detection accuracy of isolated clusters, and a hierarchical fusion module to fuse multi-level self-attention features. With group normalization, SACANet achieves better optimality in the training process. We have conducted extensive experiments using the VisDrone2019 People dataset, the VisDrone2019 Vehicle dataset, and some other challenging benchmarks. As compared with the state-of-the-art methods, SACANet is shown to be effective, especially for extremely crowded conditions with diverse scales and scattered clusters, and achieves much lower MAE as compared with baselines.*

## 1. Introduction

Crowd counting is to estimate the number of objects (such as people or vehicles) in unconstrained congested environments, where the image is often taken by a surveillance camera or unmanned aerial vehicle (UAV). Crowd counting has attracted widespread attention due to its application in public safety, congestion monitoring and traffic management [30], [11].

A promising approach for crowd counting is to use den-

sity map regression-based Convolutional Neural Networks (CNNs), which estimate the number of objects per unit pixel instead of detecting, recognizing and counting objects in the whole image. Despite recent advances, precise crowd counting remains challenging. This is mainly due to the following two factors:

- *Large variation in object scale:* Due to the perspective and distance of the camera with respective to an object, two objects of the same physical size would appear as different size (or scale) in an image, the so-called “scale variation.” In a crowded scene, objects may have large scale variation. Some works have studied this scale variation issue and achieved encouraging results using multi-branch architecture with different filter sizes [36, 26]. However, they are often based on fixed kernel size, which is only sensitive within a few narrow bands of receptive fields in the full spectrum, and hence cannot generally support diverse scenarios.
- *Isolated small clusters of objects:* An image often has isolated object clusters, i.e., some targets are located far from the crowded mass. These isolated clusters may be many but small in size, e.g., one or two objects in many scattered clusters. Convolutional operations with local receptive fields often cannot capture properly long-range contextual information, and hence have difficulty in modeling such isolated small clusters. This leads to unsatisfactory results, especially for many such small isolated clusters.

Crowd scenes captured by drones or surveillance cameras exhibit a wide range of scale variations and object clustering. We illustrate in Table 1 three typical images with vehicles from the VisDrone2019 dataset [38]. We show the original images in the first row; their corresponding density maps in the second; and the scale (object size) distribution of the objects in pixels, in terms of the average of its length and height, in the third row. To understand the small clustering effect, we take the distances of the  $N$  closest neighbors from an object and calculate their averages, for some small

Table 1. Illustration of scale variation and small isolated clusters based on three typical images from the VisDrone2019 Vehicle dataset.

	Image A	Image B	Image C																																																																		
Original image																																																																					
Density map																																																																					
Scale distribution	<table border="1"><caption>Data for Scale Distribution (Image A)</caption><thead><tr><th>Object size (x20 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.2</td></tr><tr><td>1</td><td>0.4</td></tr><tr><td>2</td><td>0.15</td></tr><tr><td>3</td><td>0.1</td></tr><tr><td>4</td><td>0.05</td></tr><tr><td>5</td><td>0.02</td></tr><tr><td>6</td><td>0.02</td></tr><tr><td>7</td><td>0.01</td></tr><tr><td>8</td><td>0.01</td></tr><tr><td>9</td><td>0.01</td></tr></tbody></table>	Object size (x20 pixels)	Fraction	0	0.2	1	0.4	2	0.15	3	0.1	4	0.05	5	0.02	6	0.02	7	0.01	8	0.01	9	0.01	<table border="1"><caption>Data for Scale Distribution (Image B)</caption><thead><tr><th>Object size (x20 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.05</td></tr><tr><td>1</td><td>0.9</td></tr><tr><td>2</td><td>0.1</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.02</td></tr><tr><td>5</td><td>0.01</td></tr><tr><td>6</td><td>0.01</td></tr><tr><td>7</td><td>0.01</td></tr><tr><td>8</td><td>0.01</td></tr><tr><td>9</td><td>0.01</td></tr></tbody></table>	Object size (x20 pixels)	Fraction	0	0.05	1	0.9	2	0.1	3	0.05	4	0.02	5	0.01	6	0.01	7	0.01	8	0.01	9	0.01	<table border="1"><caption>Data for Scale Distribution (Image C)</caption><thead><tr><th>Object size (x20 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.05</td></tr><tr><td>1</td><td>0.45</td></tr><tr><td>2</td><td>0.25</td></tr><tr><td>3</td><td>0.15</td></tr><tr><td>4</td><td>0.05</td></tr><tr><td>5</td><td>0.02</td></tr><tr><td>6</td><td>0.02</td></tr><tr><td>7</td><td>0.01</td></tr><tr><td>8</td><td>0.01</td></tr><tr><td>9</td><td>0.01</td></tr></tbody></table>	Object size (x20 pixels)	Fraction	0	0.05	1	0.45	2	0.25	3	0.15	4	0.05	5	0.02	6	0.02	7	0.01	8	0.01	9	0.01
Object size (x20 pixels)	Fraction																																																																				
0	0.2																																																																				
1	0.4																																																																				
2	0.15																																																																				
3	0.1																																																																				
4	0.05																																																																				
5	0.02																																																																				
6	0.02																																																																				
7	0.01																																																																				
8	0.01																																																																				
9	0.01																																																																				
Object size (x20 pixels)	Fraction																																																																				
0	0.05																																																																				
1	0.9																																																																				
2	0.1																																																																				
3	0.05																																																																				
4	0.02																																																																				
5	0.01																																																																				
6	0.01																																																																				
7	0.01																																																																				
8	0.01																																																																				
9	0.01																																																																				
Object size (x20 pixels)	Fraction																																																																				
0	0.05																																																																				
1	0.45																																																																				
2	0.25																																																																				
3	0.15																																																																				
4	0.05																																																																				
5	0.02																																																																				
6	0.02																																																																				
7	0.01																																																																				
8	0.01																																																																				
9	0.01																																																																				
Distance distribution	<table border="1"><caption>Data for Distance Distribution (Image A)</caption><thead><tr><th>Distance (x100 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.7</td></tr><tr><td>1</td><td>0.15</td></tr><tr><td>2</td><td>0.05</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.01</td></tr><tr><td>5</td><td>0.01</td></tr><tr><td>6</td><td>0.01</td></tr><tr><td>7</td><td>0.01</td></tr></tbody></table>	Distance (x100 pixels)	Fraction	0	0.7	1	0.15	2	0.05	3	0.02	4	0.01	5	0.01	6	0.01	7	0.01	<table border="1"><caption>Data for Distance Distribution (Image B)</caption><thead><tr><th>Distance (x100 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.9</td></tr><tr><td>1</td><td>0.1</td></tr><tr><td>2</td><td>0.05</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.01</td></tr><tr><td>5</td><td>0.01</td></tr><tr><td>6</td><td>0.01</td></tr><tr><td>7</td><td>0.01</td></tr></tbody></table>	Distance (x100 pixels)	Fraction	0	0.9	1	0.1	2	0.05	3	0.02	4	0.01	5	0.01	6	0.01	7	0.01	<table border="1"><caption>Data for Distance Distribution (Image C)</caption><thead><tr><th>Distance (x100 pixels)</th><th>Fraction</th></tr></thead><tbody><tr><td>0</td><td>0.7</td></tr><tr><td>1</td><td>0.2</td></tr><tr><td>2</td><td>0.1</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.02</td></tr><tr><td>5</td><td>0.01</td></tr><tr><td>6</td><td>0.01</td></tr><tr><td>7</td><td>0.01</td></tr></tbody></table>	Distance (x100 pixels)	Fraction	0	0.7	1	0.2	2	0.1	3	0.05	4	0.02	5	0.01	6	0.01	7	0.01												
Distance (x100 pixels)	Fraction																																																																				
0	0.7																																																																				
1	0.15																																																																				
2	0.05																																																																				
3	0.02																																																																				
4	0.01																																																																				
5	0.01																																																																				
6	0.01																																																																				
7	0.01																																																																				
Distance (x100 pixels)	Fraction																																																																				
0	0.9																																																																				
1	0.1																																																																				
2	0.05																																																																				
3	0.02																																																																				
4	0.01																																																																				
5	0.01																																																																				
6	0.01																																																																				
7	0.01																																																																				
Distance (x100 pixels)	Fraction																																																																				
0	0.7																																																																				
1	0.2																																																																				
2	0.1																																																																				
3	0.05																																																																				
4	0.02																																																																				
5	0.01																																																																				
6	0.01																																																																				
7	0.01																																																																				

$N$  (say, 1-5). A high average means that the object is in a small isolated cluster of size no more than  $Nk$  from the mass, and vice versa. In the fourth row of Table 1, we hence show the distribution of the average distance of the nearest  $N$  neighbors of an object, for  $N = 2$ .

We measure the degree of scale variation by the coefficient of variation (CV), defined as the ratio between the standard deviation and the mean of the object scale [1]. The larger the CV is, the more challenging the scale variation problem is. Furthermore, we measure the level of small clustering by the Dunn Validity Index (DVI) using the distance distribution [21] after applying K-means algorithms [7] ( $K = 2$  in our case) on it. The larger the DVI, the more small-size scattered clusters there are.

We show in Table 1 the CV for scale variation and DVI for distance distribution for the three images. The scale distribution of Image A is wide, while its distance plot is rather

continuous. Therefore, it is an image mainly characterized by scale variation without severe isolated clustering. This is validated with its relatively high CV on the scale (0.7592) and low DVI value (0.0205), and is clearly shown on its density map. On the other hand, the scale plot of Image B shows a rather narrow distribution, while its distance plot contains some outliers (separate bars). It is an image mainly with small isolated clusters, as validated with a low CV of scale (0.2054) and high DVI value (1.5871). Image C is intermediate between the two images: the size plot has a wide range and the distance plot has outliers (separate bars). It has both scale variation and isolated clustering, with its CV of scale and DVI value between Images A and B. This is clearly shown by its original image and density map.

To tackle diverse scale variation and capture long-range contextual information in isolated small clusters, we propose SACANet, a scale-adaptive long-range context-aware

**network** for accurate crowd estimation and high-quality density map generation. SACANet fully extracts the long-range contextual features in an image via a pyramid encoder by adaptively enlarging its receptive field. Using a novel scale-adaptive self-attention module with multi-branch architecture, it attains high scale sensitivity and much better accuracy in detecting isolated small clusters (based on the extracted long-range contextual information). Finally, it employs an efficient hierarchical fusion module to combine the multi-level scale and contextual features to generate a highly accurate density map. SACANet also uses group normalization to achieve higher optimality and better convergence by reducing small batch size influence.

We have conducted extensive experiments on both drone-based and surveillance camera-based crowd datasets for unconstrained crowd counting. We use the images from the four datasets: VisDrone2019 People & Vehicle, ShanghaiTech A & ShanghaiTech B. As compared with the state-of-the-art approaches, SACANet achieves superior performance on all of the four challenging benchmarks.

The rest of the paper is organized as follows. In Section 2, we review the related work. Then we present the details of SACANet in Section 3, in terms of its pyramid contextual module, scale-adaptive self-attention module, hierarchical fusion, group normalization, and objective function. We discuss the experimental setting and illustrate the results in Section 4, and conclude in Section 5.

## 2. Related Work

In this section, we discuss the related work of crowd counting methods in three main directions: traditional crowd counting algorithms (Section 2.1), deep learning-based approaches (Section 2.2), and crowd counting for drone-based scenes (Section 2.3).

### 2.1. Traditional Approaches

Early approaches for crowd counting are often based on detection models with hand-crafted features, i.e., they leverage pedestrian or body-part detectors to detect individual objects and count the number in the whole image [25], [17]. However, the performance of these detection-based methods degrades seriously in highly crowded scenes. Some researchers have attempted to use regression-based approaches with low-level features like HOG and SIFT to calculate the global number [4]. Even though relying on low-level features, these approaches achieve better results for the global count estimation. To incorporate spatial information, researchers have proposed the density map regression-based approaches, that is, measuring the number of people per unit pixel of an area in a crowd scene. As has been discussed in [14], the work is the first one to provide a density map regression-based crowd counting approach with linear mapping algorithms. And then, another work improves it

with random forest regression to learn non-linear mapping and achieves much better performance [23].

### 2.2. Deep Learning-based Approaches

Recently, researchers have adopted deep learning-based methods instead of relying on hand-crafted features to generate high-quality density maps and achieve accurate crowd counting [3], [27]. These approaches can be applied to count different kinds of objects (i.e., vehicles and cells) instead of people [15], [9].

Researchers propose multi-column convolutional neural networks with different kernel sizes for each column to address the scale variation problem [36]. Switching-CNN attaches a patch-based switching block to the multi-column structure, and better handles the particular range of scale for each column [26]. HydraCNN utilizes a pyramid of image patches with multiple scales for crowd estimation [22]. However, the counting networks with inappropriate receptive field size will give an unbalanced focus to multi-scale targets. In addition, these methods only rely on static receptive field size, which cannot be extended to tackle wide-scale variation in a crowd scene. Besides, current approaches cannot fully leverage the long-range contextual information and have difficulty in modeling isolated small clusters.

### 2.3. Drone-based Scenarios

Drone sensors and surveillance cameras both can capture crowd scenes, but crowd analysis for drone-based scenarios is more flexible for smart city applications [19]. However, previous works focus more on surveillance camera-based crowd counting. To the best of our knowledge, drone-based crowd counting has not yet been fully explored, and it also lacks publicly large-scale diversified datasets. Besides, compared with images taken by surveillance cameras, the isolated small clusters problem is more severe for drone-based crowd images [24]. In this paper, we modified the VisDrone2019 challenging dataset [37] into two large-scale diversified drone-based crowd counting datasets, and these newly split datasets can promote the field. Besides, we tackle the two main challenging scale variation and the isolated small clusters problem for unconstrained crowd counting within an end-to-end framework.

## 3. SACANet for Crowds with Scale Variation and Isolated Clusters

In this section, we present the details of SACANet, a novel scale-adaptive long-range context-aware network for crowd counting. SACANet consists of three components: the pyramid contextual module (Section 3.1), the scale-adaptive self-attention module (Section 3.2), and the hierarchical fusion module (Section 3.3). Besides, we describe

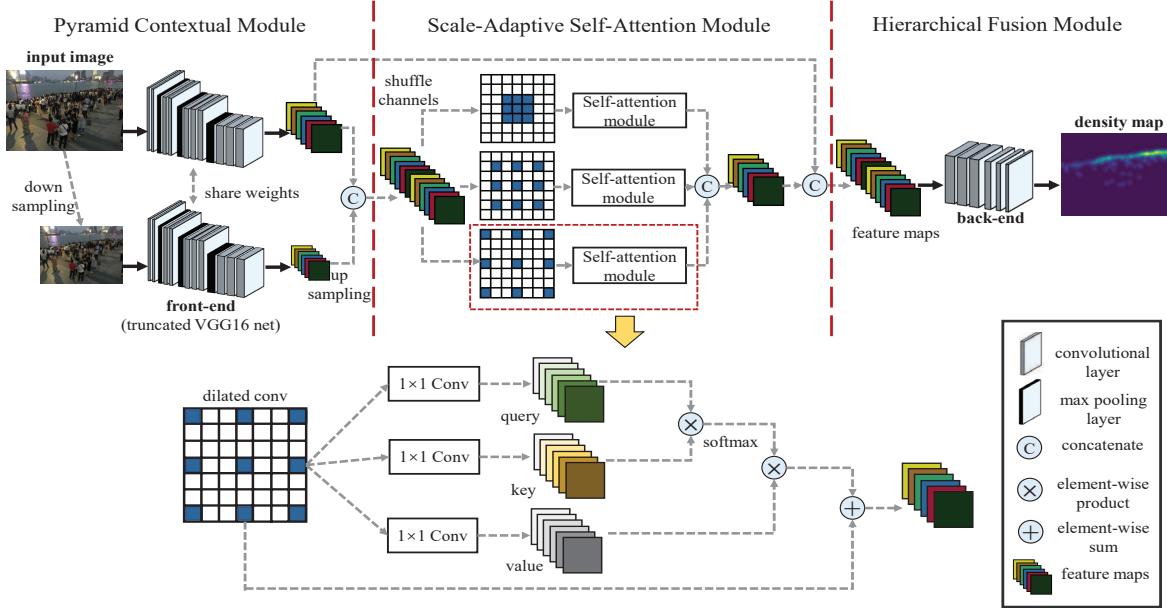


Figure 1. The architecture of SACANet for crowd counting. It contains three main components: the pyramid contextual module, the scale-adaptive self-attention module, and the hierarchical fusion module.

how to utilize group normalization to facilitate convergence and achieve better optimality in Section 3.4. The details of the objective function is discussed in Section 3.5.

In Figure 1, we show the architecture of SACANet. The upper part presents the three components of SACANet, and we describe the scale-adaptive self-attention module in more details in the bottom part (under the yellow arrow).

### 3.1. Pyramid Contextual Module

The pyramid contextual module aims to extract long-range contextual information and enlarge the receptive field. For fair comparison with previous works [15] [5] [20], our approach incorporates truncated VGG-16 [28] with excellent transferability as the backbone. We extract the first ten layers of VGG-16 with only three pooling layers to balance the large valid receptive field and density map resolution. Besides, adopting multi-branch only for the higher layers can benefit memory efficiency during training.

The pyramid contextual module is composed of a feature extractor and contextual fusion. We downsample the input image to 1/4 of the original resolution and then feed these two kinds of inputs to the VGG-16 front-end with shared weights. We shuffle the channels of the generated feature maps before feeding them into the second scale-adaptive self-attention module. This component can enlarge the receptive field and capture long-range contextual information. The encoder part is also designed to facilitate the downstream scale-adaptive self-attention processing.

### 3.2. Scale-Adaptive Self-Attention Module

The scale-adaptive self-attention module is to accommodate wide scales and detect isolated clusters via the long-

range contextual information. This part consists of three branches with the same filter size ( $3 \times 3$ ), but different dilated ratios [35] ( $1 \times 1, 2 \times 2, 3 \times 3$ ). We also add a separate convolution layer with a filter size  $1 \times 1$  at the beginning of each branch in order to reduce the numbers of channels to 1/4 of its input. This can help to reduce memory requirements without sacrificing performance [31]. Besides, the scale-adaptive self-attention module is one of the key elements of each branch after the dilated convolutional layer.

The simplest way to use multi-branch features is to concatenate them [36]. However, the features with different receptive fields would be quite large and may contain redundant information. Inspired by [32], we utilize a scale-adaptive self-attention module to capture long-range dependencies, which computes a weighted sum of values and assigns weights to measure the importance of this branch. Crowd analysis suffers from scale variations. A single multi-branch structure only possesses the same weight for each branch, and inappropriate filter size has a bad effect on the estimation. Our scale-adaptive self-attention module is able to decide for itself which to focus.

The scale-adaptive self-attention module first transfers input  $x$  to query  $Q_x$ , key  $K_x$  and value  $V_x$ :

$$Q_x = f(x), K_x = g(x), V_x = h(x). \quad (1)$$

The output weighted density map  $Y$  is computed by two kinds of matrix multiplications:

$$Y = \text{softmax}(Q_x K_x^T) V_x. \quad (2)$$

Therefore, our scale-adaptive self-attention module can automatically choose the most suitable branches and enlarge the receptive field with limited extra parameters.

### 3.3. Hierarchical Fusion Module

The hierarchical fusion module integrates multi-level self-attention features to achieve accurate crowd counting estimation. It can take advantage of autofocusing branches with different receptive fields and generate high-quality density maps. We need long-range contextual information from the deeper layers with a large receptive field and semantic information. At the same time, we also require short-range contextual information for each unit. Therefore, our method can accurately model multi-level contextual information and recognize isolated small clusters.

Inspired by ResNet [8] for recognition and RefineNet [16] for semantic segmentation, we gradually refine all the details of the generated density maps from the previous layers. Besides, there is a fully convolutional network backbone in the end to recover the spatial information. Finally, an output convolutional layer is used to predict the density map value for each pixel. Note that ReLU operations are added after each convolutional layer [6], and the entire network can be efficiently trained end-to-end.

### 3.4. Group Normalization for Better Convergence

When directly training the entire network, we find that the model cannot converge well due to a gradient vanishing problem. We tried batch normalization [10] but the result is not ideal because the error increases when the batch size becomes smaller. Our task cannot use a large batch size due to large image resolution and limited computation memory. Inspired by [34], we utilize group normalization instead of batch normalization in our approach for better convergence.

GN separates the channels into different groups and calculates the mean  $\mu$  and standard deviation  $\sigma$  within each group. This has no relation to batch size and enables our network to converge. The formulation for the mean and standard deviation are as follows:  $\mu_i = \frac{1}{m} \sum_{k \in S_i} x_k$ ,  $\sigma_i = \sqrt{\frac{1}{m} \sum_{k \in S_i} (x_k - \mu_i)^2 + \epsilon}$ , where  $\epsilon$  is a small constant, and  $m$  is the size of the set.  $S_i$  is the set of pixels where the mean and standard deviation are computed. For group normalization, set  $S_i$  can be defined as,  $S_i = k | k_N = i_N$ ,  $\lfloor \frac{k_G}{C/G} \rfloor = \lfloor \frac{i_G}{C/G} \rfloor$ , where  $G$  is the number of groups, which is a hyper-parameter we need to decide.  $C/G$  is the number of channels for each group, and  $\lfloor \cdot \rfloor$  is the floor operation. In our experiment, we set the channels per group at 16 if the total number of channels is larger than 16, or we let  $G$  be the same as the number of channels.

### 3.5. Objective Function

Most of the recent works use Euclidean loss to optimize their models for crowd counting [36], we also use it to optimize the aforementioned network. The Euclidean loss is a pixel-wise estimation error, which is defined as:

$$L_E = \frac{1}{N} \|F(X; \alpha) - Y\|_2^2, \quad (3)$$

where  $\alpha$  indicates the model parameters,  $N$  means the number of pixels,  $X$  denotes the input image and  $Y$  is its ground truth and  $F(X; \alpha)$  is the generated density map. We can predict the crowd counting result by summarizing over the estimated crowd density map.

## 4. Experiments and Illustrative Results

In this section, we describe the evaluation metrics and comparison schemes in Section 4.1. The description of the four datasets and ground truth generation method is presented in Section 4.2. Section 4.3 shows our training details. Qualitative and quantitative analysis of both people and vehicle datasets are detailed in Section 4.4. Besides, we conduct ablation study in Section 4.5 and compare SACANet on unconstrained scenarios in Section 4.6.

### 4.1. Evaluation Metrics and Comparison Schemes

We use the coefficient of variation (CV) to measure the degree scale variation [1]. For the level of isolated small clusters, we leverage the Dunn Validity Index (DVI) to calculate the distance distribution after applying K-means algorithms on it [21]. The two evaluation metrics are defined as follows:  $CV = \sigma/\mu$ , where  $\sigma$  is the standard deviation, and  $\mu$  is the mean,

$$DVI = \frac{\min_{0 < m \neq n < K} \left\{ \min_{\forall x_i \in \Omega_m, \forall x_j \in \Omega_n} \{ \|x_i - x_j\| \} \right\}}{\max_{0 < m \neq n \leq K} \max_{\forall x_i, x_j \in \Omega_m} \{ \|x_i - x_j\| \}}. \quad (4)$$

DVI calculates the shortest inter-cluster distance divided by the maximum inner-cluster distance. In our experiments, we apply the  $K$ -means algorithm [7] ( $K = 2$  in our experiment) to preprocess the average  $N = 2$  nearest distance data, and then calculate the DVI value. The larger the DVI value, the larger the inter-cluster distance and the shorter the inner-cluster distance, and the more isolated small-size clusters there are.

For overall crowd counting results, two metrics are used for evaluation [33], Mean Absolute Error (MAE) and Mean Squared Error (MSE), which are defined as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^N |C_i - \hat{C}_i|, \quad MSE = \sqrt{\frac{1}{N} \sum_{i=1}^N |C_i - \hat{C}_i|^2},$$

where  $N$  is the total number of test images,  $C_i$  means the ground truth count of the  $i$ -th image, and  $\hat{C}_i$  represents the estimated count.

We compare our approach with three schemes on the VisDrone2019 datasets: VGG-16 [28], MCNN [36] and CSRNet [15]. VGG-16 is a strong backbone, and we directly modify it into a crowd counting network. Multi-Column Convolutional Neural Network (MCNN) is a well-known crowd counting approach with multi-branch architecture. We implement it in our framework with three branches, which is the same as ours for a fair comparison. CSRNet

Table 2. Statistics of different datasets in our experiment.

Dataset	Average Resolution	Images	Max	Min	Total
VisDrone2019 People [37]	$969 \times 1482$	3347	289	10	108,464
VisDrone2019 Vehicle [37]	$991 \times 1511$	5303	349	10	198,984
ShanghaiTech A [36]	$589 \times 868$	482	313	33	241,677
ShanghaiTech B [36]	$768 \times 1024$	716	578	9	88,488

is one of the state-of-the-art methods for congested scenes understanding, which leverages dilated convolution to enlarge the receptive field. We implement it with the same experiment settings as ours.

## 4.2. Datasets and Ground Truth Generation

We evaluate our method on four challenging crowd counting datasets: VisDrone2019 People & Vehicle, and ShanghaiTech A & B. The statistics of the four datasets are presented in Table 2.

VisDrone2019 People [38]. We modify the original VisDrone2019 object detection dataset [37] with bounding boxes of targets to crowd counting annotations. The original VisDrone2019 dataset contains 11 categories. Category 0 (pedestrian) and category 1 (people) are combined into one dataset for people crowd counting, and the new annotation location is the head point of the original bounding box. We rearrange the data split and filter out the cases whose number of annotated target objects is less than 10. Finally, this dataset consists of 2392 training samples, 329 validation samples, and 626 test samples.

VisDrone2019 Vehicle [38]. Following the similar modification and rearranging step for the VisDrone2019 People dataset, we combine category 4 (car), category 5 (van), category 6 (truck) and category 9 (bus) into one dataset for vehicle crowd counting. We get 3953 training samples, 364 validation samples, and 986 test samples. The new vehicle annotation location is the center point of the original bounding box. These two kinds of new annotation operations are defined as follows:

$$\begin{aligned} \text{People}[cols, rows] &= [\text{bb}_{\text{left}} + \frac{\text{bb}_{\text{width}}}{2}, \text{bb}_{\text{top}}], \\ \text{Vehicle}[cols, rows] &= [\text{bb}_{\text{left}} + \frac{\text{bb}_{\text{width}}}{2}, \text{bb}_{\text{top}} + \frac{\text{bb}_{\text{height}}}{2}]. \end{aligned} \quad (6)$$

ShanghaiTech A [36]. The ShanghaiTech A dataset includes 482 crowd images with a total number of 241,677 persons. The counts for each image vary from 33 to 3139. This dataset is randomly crawled from the Internet and has unfixed resolutions.

ShanghaiTech B [36]. The ShanghaiTech B dataset has 716 images with fixed resolutions, which were taken from busy shopping streets by fixed cameras. The counts for each image vary from 12 to 578.

We generate the ground truth by way of blurring each head or center of vehicle annotation with a Gaussian kernel (which is normalized to 1) and taking the spatial distribution of all images into consideration from each dataset. For the

sparse crowd dataset VisDrone2019 People, VisDrone2019 Vehicle, and ShanghaiTech B, we use fixed kernel  $\sigma = 15$  as the generating method. For ShanghaiTech A crowded datasets, we adopt the geometry-adaptive kernels method to generate the ground truth [36].

## 4.3. Training Details

In our experiment, we use the VGG-16 backbone with pretrained weights from the ImageNet classification challenge dataset [13]. For the other layers, we initialize with random weights from Gaussian distributions with zero mean and a standard deviation of 1. We use the Adam optimizer [12] with an initial learning rate of 1e-4. For the ShanghaiTech B dataset, we randomly flip some of the samples to augment the original training data. For datasets with much higher resolutions, we resize the input image to no more than  $768 \times 1024$  but maintain the same aspect ratio.

In the test step, we directly test the full images for the ShanghaiTech B dataset with fixed resolutions. For the other datasets with unfixed resolutions, we resize the images before feeding them into our network. The estimated total count for each image is given by summing the whole image. We implement SACANet based on PyTorch.

## 4.4. Experiments on the VisDrone2019 Datasets

We modify the original VisDrone2019 challenge dataset into two crowd counting datasets (VisDrone2019 Vehicle & People) and conduct extensive experiments on the two new datasets. Details are presented in Table 2. We implement a strong VGG-16 network and the state-of-the-art approaches MCNN [36], and CSRNet [15] and compare with SACANet on the same dataset. SACANet achieves much better performance on both the VisDrone2019 People and the VisDrone2019 Vehicle datasets, and the quantitative results are presented in Table 3.

Figure 2 shows the qualitative results on both the VisDrone2019 Vehicle and the VisDrone2019 People datasets. The first three rows are the results for vehicle crowd counting, and the last three rows are results for people crowd counting. For each part, the first row shows the original image of the VisDrone2019 dataset, the second row presents the ground truth, and the third row is our generated density map. Besides, the total number of ground truth and our estimated counting results are shown below each part.

Furthermore, we split the VisDrone2019 Vehicle & People datasets based on the different levels of scale variation and object separation problem distribution. In Figure 3, we

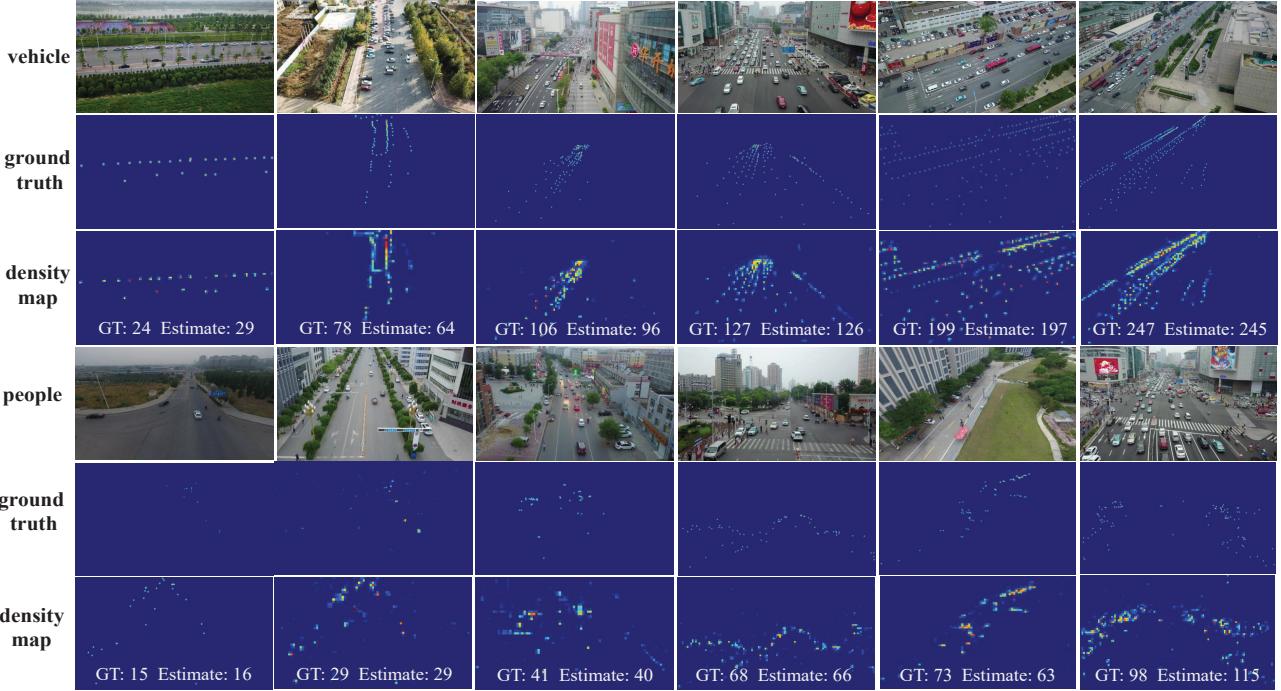


Figure 2. Qualitative results of SACANet on the VisDrone2019 Vehicle & People dataset.

Table 3. Quantitative results on the VisDrone2019 Datasets.

Method	Dataset People		Dataset Vehicle	
	MAE	MSE	MAE	MSE
VGG-16 [28]	22.0	44.8	21.4	29.3
MCNN [36]	16.4	39.1	14.9	21.6
CSRNet [15]	12.1	36.7	10.9	16.6
SACANet(ours)	<b>10.5</b>	<b>35.1</b>	<b>8.6</b>	<b>12.9</b>

plot perception vs. the evaluation metrics (i.e., Coefficient of Variation for object size distribution and Dunn Validity Index for distance distribution). Figure 3 (a) and (b) presents the scale distribution measured by CV. (c) and (d) shows the distance distribution measured by DVI.

For the scale variation problem, we divided the test dataset into five parts based on the value of CV. Thus, we get a five-level scale variation problem setting with different CV ranges:  $[0.0, 0.2)$ ,  $[0.2, 0.4)$ ,  $[0.4, 0.6)$ ,  $[0.6, 0.8)$ ,  $[0.8, \infty)$ . The higher the scale split set level, the more difficult the scale variation problem. Besides, for the isolated small clusters problem, the test dataset is decomposed into four subsets based on the DVI value, which indicates four levels of the isolated clusters problem:  $[0.0, 1.0)$ ,  $[1.0, 2.0)$ ,  $[2.0, 3.0)$ ,  $[3.0, \infty)$ . The higher the distance split sets the level, the more small-size isolated clusters there are.

We compared SACANet with MCNN [36] on the split subsets. In Figure 4 (a) and (c), we plot the value of MAE and MSE vs. scale (CV) ranges. For (b) and (d), we plot the value MAE and MSE vs. distance (DVI) ranges. The results of SACANet are always better than MCNN on most of the

Table 4. Ablation study on the VisDrone2019 dataset.

Method	Dataset People		Dataset Vehicle	
	MAE	MSE	MAE	MSE
baseline	14.5	40.8	12.5	19.1
baseline+context	13.8	39.2	11.0	16.4
baseline+context+SASA	11.7	36.1	9.1	13.6
SACANet(ours)	<b>10.5</b>	<b>35.1</b>	<b>8.6</b>	<b>12.9</b>

subsets. Our method achieves much lower MAE on the VisDrone2019 Vehicle than on the people dataset as the vehicle dataset is relatively more crowded, which indicates that SACANet is more effective for extremely crowded scenes.

#### 4.5. Ablation Study

In this section, we perform ablation studies on the VisDrone2019 People & Vehicle datasets and analyze the results, which shows the effectiveness of our approach.

**Effectiveness of the baseline:** Our baseline network consists of three parts: a truncated VGG-16, multi-branch subnet, and the backend. For a given input image, we feed it to our baseline network and get its generated density map. And then, we sum all the pixel values to get the total count.

**Effectiveness of the pyramid context module:** After the operations mentioned above, we use our context-aware front-end instead of one column VGG-16 backbone to train our model, and we find that the error decreased.

**Effectiveness of the Scale-Adaptive Self-Attention (SASA):** We enrich the baseline with our novel scale-adaptive attention scheme to each branch. The results show

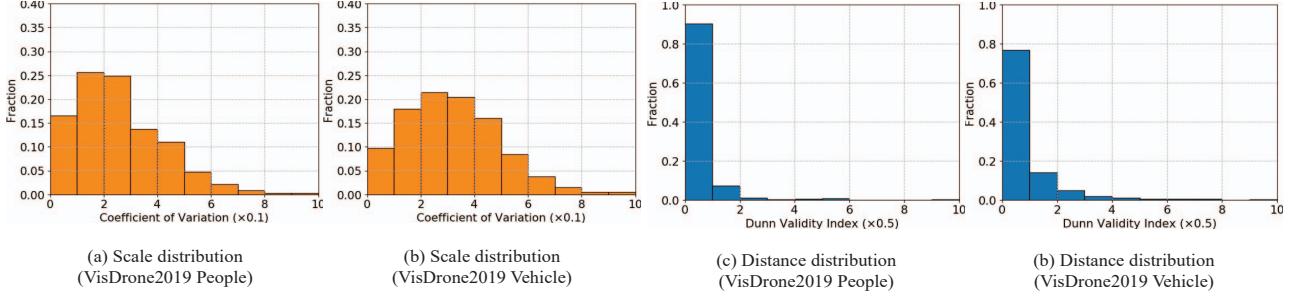


Figure 3. Distribution of scale variation and object separation problem on the VisDrone2019 Vehicle & People datasets.

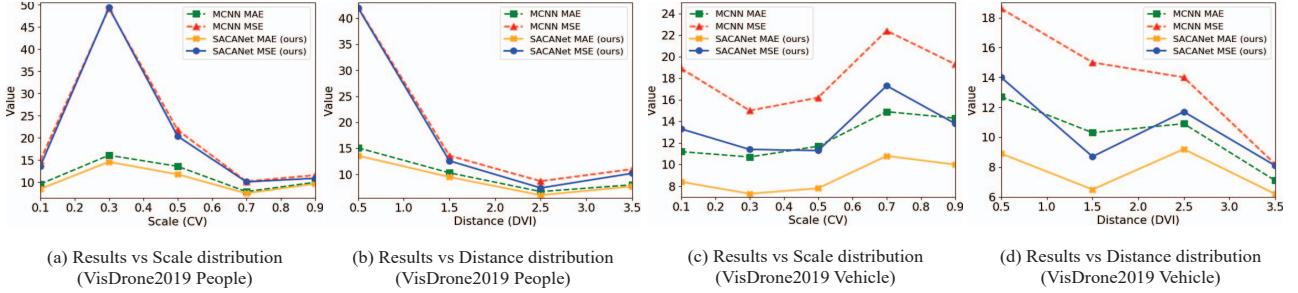


Figure 4. Comparison via different distributions on the VisDrone2019 Vehicle & People datasets.

Table 5. Results of our approaches on two challenging people counting datasets. ShanghaiTech A & B.

Method	Dataset A		Dataset B	
	MAE	MSE	MAE	MSE
MCNN [36]	110.2	173.2	26.4	41.3
Switching-CNN [26]	90.4	135.0	21.6	33.4
CP-CNN [29]	73.6	106.4	20.1	30.1
ACSCP [27]	75.7	102.7	17.2	27.4
IG-CNN [2]	72.5	118.2	13.6	21.1
CSRNet [15]	68.2	115.0	10.6	16.0
DRSAN [18]	69.3	96.4	11.1	18.2
SANet [3]	67.0	104.5	8.4	13.6
Baseline	68.3	107.5	11.9	18.7
SACANet(ours)	<b>64.4</b>	<b>95.9</b>	<b>7.8</b>	<b>13.5</b>

a significant improvement in terms of MAE and MSE.

Effectiveness of the hierarchical fusion: We add the hierarchical fusion module and retrain the model. This module also brings improvement to the performance, and we get our final results in Table 4.

#### 4.6. Evaluation on Unconstrained Scenarios

We compare SACANet with other approaches in the literature on two challenging people crowd counting datasets with unconstrained scenarios. The results are proposed in Table 5. Our methods always show a better performance than the baseline, which demonstrates the importance of the pyramid contextual module and scale-adaptive

self-attention mechanism. Besides, we see that our approach surpasses the state-of-the-art methods CSRNet [15] and SANet [3] for both the ShanghaiTech A and the ShanghaiTech B datasets. Compared with SANet [3], our SACANet reduces the MAE by over 7% on the ShanghaiTech A dataset and reduces the MAE by about 4% on the ShanghaiTech B dataset. These results further demonstrate the effectiveness of our SACANet.

## 5. Conclusion

In this work, we tackle two main challenges in crowd counting: large scale variation and isolated small clusters. We propose SACANet, a novel Scale-Adaptive long-range Context-Aware network for accurate crowd counting in unconstrained crowded scenes. A pyramid contextual module can fully encode the contextual information. We present a scale-adaptive self-attention scheme to automatically choose the most appropriate branches and naturally enlarge the receptive field. By utilizing the hierarchical fusion module, our method can fuse multi-level contextual information in crowded scenes.

Extensive experiments show that our approach achieves compelling results on the VisDrone2019 Vehicle & People datasets and two other challenging people crowd counting benchmarks. As compared with prior arts, SACANet achieves much better performance in terms of MAE and MSE, especially when the image exhibits large variation in object scales and many isolated small clusters.

## References

- [1] Hervé Abdi. Coefficient of variation. *Encyclopedia of research design*, 1:169–171, 2010.
- [2] Deepak Babu Sam, Neeraj N Sajjan, R Venkatesh Babu, and Mukundhan Srinivasan. Divide and grow: capturing huge diversity in crowd images with incrementally growing cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3618–3626, 2018.
- [3] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.
- [4] Antoni B Chan and Nuno Vasconcelos. Counting people with low-level features and bayesian regression. *IEEE Transactions on Image Processing*, 21(4):2160–2177, 2012.
- [5] Xinya Chen, Yanrui Bin, Nong Sang, and Changxin Gao. Scale pyramid network for crowd counting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1941–1950. IEEE, 2019.
- [6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.
- [7] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [9] Shenghua He, Kyaw Thu Minn, Lilianna Solnica-Krezel, Hua Li, and Mark Anastasio. Automatic microscopic cell counting by use of unsupervised adversarial domain adaptation and supervised density regression. In *Medical Imaging 2019: Digital Pathology*, volume 10956, page 1095604. International Society for Optics and Photonics, 2019.
- [10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [11] Di Kang, Zheng Ma, and Antoni B Chan. Beyond counting: Comparisons of density maps for crowd analysis tasks—counting, detection, and tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5):1408–1422, 2018.
- [12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [14] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in neural information processing systems*, pages 1324–1332, 2010.
- [15] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018.
- [16] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.
- [17] Zhe Lin and Larry S Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):604–618, 2010.
- [18] Lingbo Liu, Hongjun Wang, Guanbin Li, Wanli Ouyang, and Liang Lin. Crowd counting using deep recurrent spatial-aware network. *arXiv preprint arXiv:1807.00601*, 2018.
- [19] Weizhe Liu, Krzysztof Maciej Lis, Mathieu Salzmann, and Pascal Fua. Geometric and physical constraints for drone-based head plane crowd density estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2019.
- [20] Yuting Liu, Miaojing Shi, Qijun Zhao, and Xiaofang Wang. Point in, box out: Beyond counting persons in crowds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6469–6478, 2019.
- [21] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence*, 24(12):1650–1654, 2002.
- [22] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *European Conference on Computer Vision*, pages 615–629. Springer, 2016.
- [23] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3253–3261, 2015.
- [24] Chunping Qiu, Lichao Mou, Michael Schmitt, and Xiao Xiang Zhu. Local climate zone-based urban land cover classification from multi-seasonal sentinel-2 images with a recurrent residual network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 154:151–162, 2019.
- [25] Vincent Rabaud and Serge Belongie. Counting crowded moving objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, volume 1, pages 705–711. IEEE, 2006.
- [26] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4031–4039. IEEE, 2017.
- [27] Zan Shen, Yi Xu, Bingbing Ni, Minsi Wang, Jianguo Hu, and Xiaokang Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254, 2018.
- [28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

- [29] Vishwanath A Sindagi and Vishal M Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.
- [30] Vishwanath A Sindagi and Vishal M Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters*, 107:3–16, 2018.
- [31] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [33] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.
- [34] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.
- [35] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [36] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.
- [37] Pengfei Zhu, Longyin Wen, Xiao Bian, Ling Haibin, and Qinghua Hu. Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*, 2018.
- [38] Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Ling, and Qinghua Hu. Vision meets drones: A challenge. *CoRR*, abs/1804.07437, 2018.