STACKED POOLING FOR BOOSTING SCALE INVARIANCE OF CROWD COUNTING

Siyu Huang[1] *Xi Li*[2] *Zhi-Qi Cheng*[3] *Zhongfei Zhang*[4] *Alexander Hauptmann*[3]

[1]Big Data Lab, Baidu Research, China
[2]College of Computer Science and Technology, Zhejiang University, China
[3]School of Computer Science, Carnegie Mellon University, USA
[4]Department of Computer Science, State University of New York at Binghamton, USA

ABSTRACT

In this work, we take insight into the dense crowd counting problem by exploring the phenomenon of cross-scale visual similarity caused by perspective distortions. It is a quite common phenomenon in crowd scenarios, suggesting the crowd counting model to enable a good performance of scale invariance. Existing deep crowd counting approaches mainly focus on the multi-scale techniques over convolutional layers to capture scale-adaptive features, resulting in high computing costs. In this paper, we propose simple but effective pooling variants, i.e., multi-kernel pooling and stacked pooling, to take place of the vanilla pooling layers in convolutional neural networks (CNNs) for boosting the scale invariance. Our proposed pooling modules do not introduce extra parameters and can be easily implemented in practice. Empirical studies on two benchmark crowd counting datasets show that the proposed pooling modules beat the vanilla pooling layer in most experimental cases.

Index Terms— Crowd counting, scale invariance, pooling

1. INTRODUCTION

With the vast demands of public safety and city planning, recent years have witnessed a great development of crowd counting [1, 2, 3, 4, 5, 6, 7, 8] in visual intelligence. The goal of crowd counting is to automatically and precisely estimate the number of pedestrians in crowded scenes. Typically, crowd counting is cast as a crowd density map regression problem [9] within an end-to-end learning scheme [10, 11, 12]. In practice, a key insight into this problem is that effective density map regression requires capturing the scale-invariant crowd feature information from perspective distortions. In this paper, we focus on how to build a simple yet effective deep learning module for boosting the performance of perspective scale invariance.

As shown in Fig. 1, the crowd image patches from different perspective scales exhibit the mutually similar visual properties after resizing. This common phenomenon in crowd counting delivers a fact of the cross-scale visual similarity

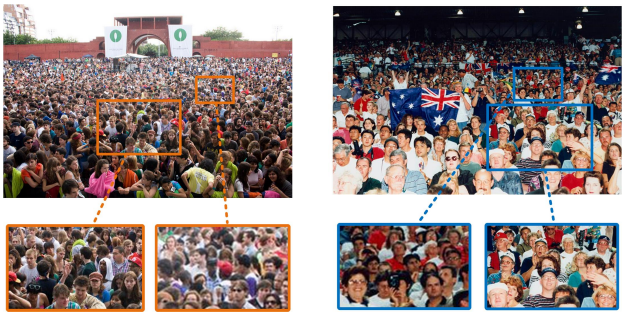

Fig. 1: In dense crowd images, regions of different scales exhibit high visual similarity if we resize them to certain sizes. This indicates the importance of scale invariance in crowd counting.

in the perspective direction [13, 14]. Hence, an ideal vision model is supposed to pursue the goal of *scale invariance* for pedestrian number estimation, no matter how a crowd image is resized to other scales, rather than the scale equivariance for general vision models [15, 16]. In the context of crowd counting, it is a common practice to take the strategy of adopting multi-scale inputs [17] or multi-branch networks [18, 18, 19, 20, 21] to enhance the scale invariance capability of convolutional neural networks (CNNs). For instance, the popular Multi-Column CNN [22] and its variants [23] adapt multi-sized convolution units to visual concepts (e.g., heads and pedestrians) of different scales. In principle, the above-mentioned approaches mainly rely on the multi-scale techniques over the convolutional layers, resulting in a higher computational burden.

In contrast, this paper designs lightweight scale-aware pooling module towards the goal of scale-invariant crowd counting. So far a series of literature [24, 25, 26] has revealed the limitations of existing pooling operations in coping with significant scale changes [27]. Consequently, the vanilla pooling often suffers from the perspective scale variations in crowd counting scenarios, as shown in Fig. 1. In this case, pooling modules with a larger receptive field [28, 29] are likely to adapt to larger scale variations, and, enabling a

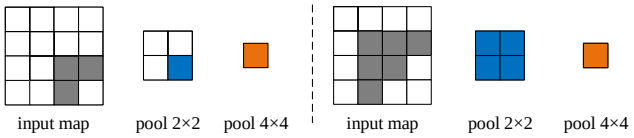

Fig. 2: An intuitive illustration of the scale invariance brought by a larger pooling kernel.

stronger scale invariance. Fig. 2 provides an intuitive illustration of how a larger pooling range enables an invariance with the input going through scale variations. The feature map after 2×2 max-pooling varies while the feature map after 4×4 max-pooling presents an invariance.

In this paper, we propose simple yet effective pooling variants, i.e., multi-kernel pooling and stacked pooling, to boost the scale invariance of CNNs. Specifically, the multi-kernel pooling comprises of pooling kernels with multiple receptive fields to capture the responses at multi-scale local ranges, and then, concatenating the feature maps together to its successive layer. Technically, the larger pooling kernels can provide a wider range of scale invariance for CNNs, while the fine-grained information is also preserved by smaller pooling kernels. The stacked pooling is an equivalent form of multi-kernel pooling by stacking smaller pooling kernels. It further reduces the computing cost of multi-kernel pooling.

In practice, our proposed pooling modules have the following advantages: 1) *Non-parametric:* They do not introduce any extra parameters and hyper-parameters into the model, ensuring a high efficiency in learning; 2) *Simple and flexible:* They are succinct and very easy to implement. They can take place of the vanilla pooling layer at any time if needed. Empirically, the multi-kernel pooling and stacked pooling show favorable performance in comparison with the vanilla pooling. They beat the vanilla pooling layer in most experimental cases on two benchmark crowd counting datasets. In addition, studies on pooling kernel sizes further reveal their effectiveness.

2. OUR APPROACH

The deep CNN based crowd counting models estimate the pedestrian count in a crowd image by jointly learning the crowd density map and count. In this work, we improve the scale invariance of CNNs by introducing very simple yet effective pooling modules, including multi-kernel pooling and stacked pooling, to take place of the vanilla pooling layers in CNNs. In practice, our proposed pooling modules can be applied to various versions of poolings.

2.1. Multi-Kernel Pooling

In the practice of deep CNNs, a small pooling kernel, e.g., $k = 2$, is commonly used mainly because a larger pooling kernel may excessively discard information of the original feature map. However, a larger pooling kernel is able to provide a wider range of scale invariance for CNNs as illustrated in Fig. 2. Specifically in crowd counting, image regions of different scales generally present a high visual similarity. Thus, in this work we exploit a set of poolings with different kernel sizes, i.e., multi-kernel pooling, to boost the scale invariance of a deep crowd counting model.

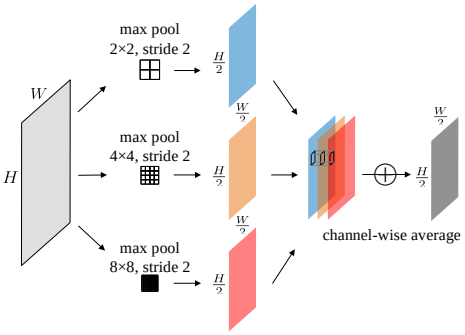

Fig. 3: **Multi-kernel pooling** with a set of kernels $\{2, 4, 8\}$ and a stride of 2. The three pooling kernels are applied on the input feature map and then concatenated with element-wise mean.

The multi-kernel pooling enables a kernel set K comprising of different pooling kernel sizes, such as $K = \{k_1, k_2, ..., k_n\}$. We apply the i-th pooling kernel $\mathcal{P}_{k_i}$ on feature map X

$$Y_i = X * \mathcal{P}_{k_i} \tag{1}$$

There are many ways to concatenate the output feature maps. In this work we use *element-wise mean* because: 1) It keeps the shape of original feature map; 2) It has been demonstrated to be effective in various deep architectures; 3) It does not introduce extra learnable parameters. Following Eq. 1, the feature maps are concatenated as

$$Y_{\text{multi-kernel}} = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{2}$$

In CNNs, we often use a pooling $\mathcal{P}_k^{(s)}$ with a sliding window stride $s \geq 2$ and proper paddings to down-sample a feature map $X \in \mathbb{R}^{W \times H}$ into $\downarrow_s Y \in \mathbb{R}^{\frac{W}{s} \times \frac{H}{s}}$. The multi-kernel pooling with a down-sampling rate s is written as

$$\downarrow_s Y_{\text{multi-kernel}} = \frac{1}{n}\sum_{k \in K} X * \mathcal{P}_k^{(s)} \tag{3}$$

In theory, the multi-sized pooling kernels incorporate responses of multiple local areas into the output feature map, thus providing a wider range of scale invariance for CNNs. In addition, the fine-grained information is also preserved by those poolings with smaller kernels. Fig. 3 illustrates an example of the multi-kernel pooling, where the kernel set

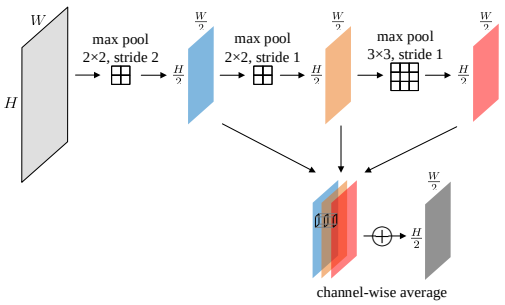

Fig. 4: **Stacked pooling** with a set of kernels $\{2, 2, 3\}$. It is an equivalent form of multi-kernel pooling shown in Fig. 3 with less computing cost.

Table 1: Time cost of pooling methods (ms). 'pool layer' is a single pooling layer. 'network' is the VGG-13 network.

		vanilla	stacked	multi-kernel
pool layer	forward	0.11	0.37	0.84
network	forward	6.1	6.6	7.7
	backward	13.6	14.1	15.7

$K = \{2, 4, 8\}$ and the stride $s = 2$. In empirical studies, this configuration also shows the best performance in most cases.

2.2. Stacked Pooling

To reduce the computing cost of multi-kernel pooling, we propose to use its equivalent form, named stacked pooling. The stacked pooling is a stack of pooling layers, where the intermediate feature maps are consecutively computed as

$$\downarrow_{s_i'} Y_i^{'} = Y_{i-1}^{'} * \mathcal{P}_{k_i'}^{(s_i')} \tag{4}$$

Specifically, $Y_0^{'} = X$ is the input feature map. Kernel size $k_i^{'}$ corresponds to k_i with a certain transformation. Stride $s_{i=1}^{'} = s$ and $s_{i>1}^{'} = 1$. Following Eq. 4, the output of stacked pooling concatenates the intermediate feature maps as

$$\downarrow_s Y_{\text{stacked}} = \frac{1}{n} \sum_{i=1}^{n} \downarrow_{s_i'} Y_i^{'} \tag{5}$$

Fig. 4 shows a diagram of stacked pooling which is exactly equivalent to the example of multi-kernel pooling shown in Fig. 3. The stacked pooling is much more efficient than multi-kernel pooling because its pooling operations are computed on down-sampled feature maps, except for its first pooling kernel. Table 1 summarizes the time cost of different pooling methods w.r.t a 256×256 input feature map. We see that the stacked pooling shows a much better computing efficiency than multi-kernel pooling. On a VGG-13 network [30], the forward and backward time of stacked pooling is close to that of vanilla pooling, thus, ensuring its practicability.

3. EXPERIMENTS

3.1. Experimental Setup

In this work, we conduct empirical studies[1] based on PyTorch framework [31] on two popular benchmark crowd counting datasets: ShanghaiTech [22] and WorldExpo'10 [32]. Both two datasets are very challenging due to diverse scene types and varying density levels. Follow the convention of existing literatures, we use mean absolute error (MAE) and mean squared error (MSE) to evaluate the performance of different crowd counting methods. The MAE metric indicates the accuracy of crowd estimation algorithm, while the MSE metric indicates the robustness of estimation.

We evaluate our proposed pooling module on different backbone CNNs. We exploit three types of network architectures, i.e., Base-Net, Wide-Net, and Deep-Net. The Base-Net is relatively small and it has three variants, namely "S" and "M", coming from the columns of Multi-Column CNN [22] and having different convolutional kernel sizes. The Wide-Net widens the Base-M Net by using more channels of feature maps. The Deep-Net follows the well-known VGG-13 network [30] with slight modifications. We use CNNs of diverse depths, widths, and convolutional kernel sizes for a comprehensive evaluation of our method.

3.2. Quantitative Comparison

ShanghaiTech Dataset Table 3 quantitatively compares methods including the baselines in previous crowd counting work and our proposed methods. Stacked pooling obviously outperforms the baselines and vanilla pooling by showing a superior performance in most settings of network architectures and metrics. The evaluated backbone networks cover the commonly used CNN architectures, from small to large, and from shallow to deep. The Deep-Net is empirically better than Wide-Net and Base-Nets on ShanghaiTech dataset. Experimental results show that the Deep-Net is 3.7% and 11.5% better under MAE by adopting stacked pooling than vanilla pooling. In theory, the stacked pooling does not introduce extra model parameters and preserves more information during the down-sampling process, thus benefiting the information flow in deep layers.

WorldExpo'10 Dataset Table 2 quantitatively compares the pooling modules on WorldExpo'10 dataset. MAE results on five different test scenes are shown respectively. We evaluate the Wide-Net and the Deep-Net for they are more often used in practice. In this experiment, the MAEs across different scenes are quite different due to diverse crowd densities of the scenes. The Deep-Net still performs better than the Wide-Net w.r.t. the average MAE. The stacked pooling performs better than the vanilla pooling w.r.t the average MAE and most of the testing scenes, indicating that the stacked pooling is as a

[1] Project codes are available at https://github.com/siyuhuang/crowdcount-stackpool

Table 2: Crowd counting performances on ShanghaiTech dataset. The first group includes baseline methods in previous work. The second group compares vanilla pooling and stacked pooling on various backbone network architectures.

Method		ShanghaiTech-A		ShanghaiTech-B	
		MAE	MSE	MAE	MSE
MCNN [22]		110.2	173.2	26.4	41.3
CMTL [1]		101.3	152.4	20.0	**31.1**
TDF-CNN [33]		97.5	**145.1**	20.7	32.8
BSAD [6]		-	-	20.2	35.6
DecideNet [3]		-	-	21.5	32.0
Base-S	+vanilla	142.42	225.21	27.64	49.22
	+stacked	127.57	197.75	22.27	41.45
Base-M	+vanilla	121.71	192.74	29.45	51.88
	+stacked	116.06	182.61	26.03	46.11
Wide	+vanilla	122.22	198.23	28.21	51.70
	+stacked	113.71	181.52	26.42	47.69
Deep	+vanilla	97.63	153.26	21.17	39.20
	+stacked	**93.98**	150.59	**18.73**	**31.86**

Table 3: MAE performances on five test scenes of World-Expo'10 dataset.

Method		#1	#2	#3	#4	#5	Ave
Wide	+vanilla	5.01	18.96	14.76	21.36	14.57	14.95
	+stacked	4.72	22.62	19.85	14.21	8.43	**13.98**
Deep	+vanilla	4.08	18.74	20.68	23.28	6.84	14.74
	+stacked	3.26	12.39	13.97	31.41	3.50	**12.92**

whole better than the vanilla pooling for crowd images with diverse densities and various scenes.

3.3. Qualitative Results

We qualitatively compare vanilla and stacked pooling by visualizing the density maps, as shown in Fig. 5. Although the only difference between the models is their pooling layers, we can see that there are obvious differences between density maps of the two models on the following aspects: (1) *Sharpness.* One main difference lies in the sharpness of the density maps. The density maps of stacked pooling are much sharper and clearer, indicating a better fitting to ground-truth density maps which are often sharp (Gaussian kernel $\sigma = 4$ in our experiments). 2) *Robustness to noises.* For instance, there is an evident error on bottom of density map #1 of vanilla pooling due to the dense textures of the woman's clothes. On top of density map #2 of vanilla pooling, some chairs are mistakenly recognized as crowds. The density maps of stacked pooling avoid these mistakes and present a better robustness to different types of noises. (3) *Robustness to scale variations.* Within image #2, there are severe scale variations among different image parts. The corresponding density maps of stacked pooling show more distinct responses, indicating a better robustness to scale variations. It is in line with the motivation of the propose of stacked pooling.

3.4. Empirical Study on Pooling Kernel Sizes

We further conduct empirical study on the configuration of pooling kernel set K based on Base-M Net on ShanghaiTech-A,B dataset. As shown in Fig. 6, four different kernel sets, including the vanilla pooling kernel $\{2\}$ and the multi-kernel pooling kernel sets $\{2, 4\}$, $\{2, 4, 8\}$, $\{2, 4, 8, 16\}$, are evaluated. We group the test images according to ground-truth pedestrian numbers and show the MAE of density groups from lower density to higher density. Fig. 6 shows that vanilla pooling performs worse than multi-kernel pooling on higher density group of ShanghaiTech-A dataset and also worse on the entire ShanghaiTech-B dataset. Among the multi-kernel pooling kernel sets, set $\{2, 4, 8\}$ performs the best with robustness on all density levels. Therefore, we employ kernel set $K = \{2, 2, 3\}$ as the default experimental configuration of stacked pooling module in this paper.

Fig. 5: Visualization of density maps. The three columns show crowd images, density maps from vanilla pooling, and density maps from stacked pooling, respectively.

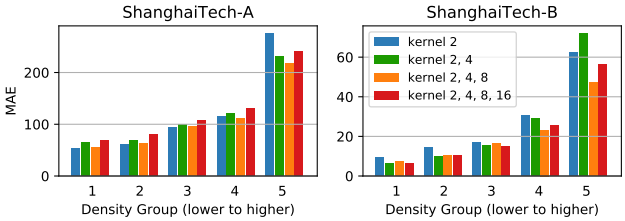

Fig. 6: Empirical studies on kernel sets of multi-kernel pooling. The MAE, vs. the density groups from lower density to higher density.

4. CONCLUSION

In this work, we have proposed simple, flexible, but effective variants of vanilla pooling module, i.e., multi-kernel pooling and stacked pooling, to boost the scale invariance of CNNs and improve crowd counting performances. The proposed pooling modules exploit a larger receptive field to enable a stronger invariance for the significant scale variations in crowd images. In experiments, the proposed pooling modules are efficient and easy to implement, showing better performance than the vanilla pooling layer in most experimental cases on benchmark crowd counting datasets.

5. REFERENCES

[1] Vishwanath A Sindagi and Vishal M Patel, "Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in *AVSS*, 2017, pp. 1–6.

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

[3] Jiang Liu, Chenqiang Gao, Deyu Meng, and Alexander G Hauptmann, "Decidenet: Counting varying density crowds through attention guided detection and density estimation," in *CVPR*, 2018, pp. 5197–5206.

[4] 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 *CVPR*, 2018, pp. 3618–3626.

[5] Hanhui Li, Xiangjian He, Hefeng Wu, Saeed Amirgholipour Kasmani, Ruomei Wang, Xiaonan Luo, and Liang Lin, "Structured inhomogeneous density map learning for crowd counting," *arXiv preprint arXiv:1801.06642*, 2018.

[6] Siyu Huang, Xi Li, Zhongfei Zhang, Fei Wu, Shenghua Gao, Rongrong Ji, and Junwei Han, "Body structure aware deep crowd counting," *IEEE TIP*, vol. 27, no. 3, pp. 1049–1059, 2018.

[7] Vishwanath A Sindagi and Vishal M Patel, "Ha-ccn: Hierarchical attention-based crowd counting network," *IEEE TIP*, vol. 29, pp. 323–335, 2019.

[8] Yuting Liu, Miaojing Shi, Qijun Zhao, and Xiaofang Wang, "Point in, box out: Beyond counting persons in crowds," in *ICCV*, 2019, pp. 6469–6478.

[9] Victor Lempitsky and Andrew Zisserman, "Learning to count objects in images," in *NIPS*, 2010, pp. 1324–1332.

[10] Zhi-Qi Cheng, Jun-Xiu Li, Qi Dai, Xiao Wu, and Alexander Hauptmann, "Learning spatial awareness to improve crowd counting," in *ICCV*, 2019.

[11] Jia Wan, Wenhan Luo, Baoyuan Wu, Antoni B Chan, and Wei Liu, "Residual regression with semantic prior for crowd counting," in *CVPR*, 2019, pp. 4036–4045.

[12] Xiaolong Jiang, Zehao Xiao, Baochang Zhang, Xiantong Zhen, Xianbin Cao, David Doermann, and Ling Shao, "Crowd counting and density estimation by trellis encoder-decoder networks," in *CVPR*, 2019, pp. 6133–6142.

[13] Miaojing Shi, Zhaohui Yang, Chao Xu, and Qijun Chen, "Revisiting perspective information for efficient crowd counting," in *CVPR*, 2019, pp. 7279–7288.

[14] Zhaoyi Yan, Yuchen Yuan, Wangmeng Zuo, Xiao Tan, Yezhen Wang, Shilei Wen, and Errui Ding, "Perspective-guided convolution networks for crowd counting," 2019.

[15] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst, "Matrix capsules with em routing," in *ICLR*, 2018.

[16] Karel Lenc and Andrea Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *CVPR*, 2015, pp. 991–999.

[17] Daniel Onoro-Rubio and Roberto J López-Sastre, "Towards perspective-free object counting with deep learning," in *ECCV*, 2016, pp. 615–629.

[18] Lingke Zeng, Xiangmin Xu, Bolun Cai, Suo Qiu, and Tong Zhang, "Multi-scale convolutional neural networks for crowd counting," in *ICIP*, 2017, pp. 465–469.

[19] Lu Zhang and Miaojing Shi, "Crowd counting via scale-adaptive convolutional neural network," in *WACV*, 2018.

[20] Yuhong Li, Xiaofan Zhang, and Deming Chen, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *CVPR*, 2018, pp. 1091–1100.

[21] Diptodip Deb and Jonathan Ventura, "An aggregated multicolumn dilated convolution network for perspective-free counting," in *CVPR Workshops*, 2018, pp. 195–204.

[22] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma, "Single-image crowd counting via multi-column convolutional neural network," in *CVPR*, 2016, pp. 589–597.

[23] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu, "Switching convolutional neural network for crowd counting," in *CVPR*, 2017, vol. 1, p. 6.

[24] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al., "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *CVPR*, 2007, pp. 1–8.

[25] Y-Lan Boureau, Jean Ponce, and Yann LeCun, "A theoretical analysis of feature pooling in visual recognition," in *ICML*, 2010, pp. 111–118.

[26] Dominik Scherer, Andreas Müller, and Sven Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *ICANN*, pp. 92–101. 2010.

[27] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014, pp. 392–407.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014, pp. 346–361.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE TPAMI*, vol. 37, no. 9, pp. 1904–1916, 2015.

[30] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS Workshop*, 2017.

[32] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015, pp. 833–841.

[33] Deepak Babu Sam and R Venkatesh Babu, "Top-down feedback for crowd counting convolutional neural network," in *AAAI*, 2018.