Crowd Counting with Stacked Pooling for Boosting Scale Invariance

Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, Fei Wu, and Alexander Hauptmann

***Abstract*—In this work, we take insight into the 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. Specifically, the multi-kernel pooling comprises of pooling kernels with multiple receptive fields to capture the responses at multi-scale local ranges. The stacked pooling is an equivalent form of multi-kernel pooling, while it reduces considerable computing cost. 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, convolutional neural network, pooling layer.**

I. Introduction

With the vast demands of public safety and city planning, recent years have witnessed a great development of crowd counting 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 within an end-to-end learning scheme. 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. Therefore, 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 in the perspective direction. 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 [1], [2]. In the context of crowd counting, it is a common practice to take the strategy of adopting multi-scale inputs [3] or establishing multi-branch networks [4] to enhance the scale invariance capability of convolutional neural networks (CNNs). For instance, the popular Multi-Column CNN [5] and its variants [6] 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.

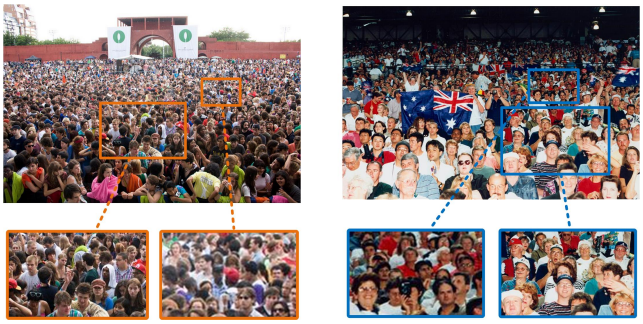

Fig. 1. In dense crowd images, regions of different scales exhibit high visual similarity if we resize them to certain sizes. This indicates the importance of scale invariance in crowd counting.

In contrast, we model the scale invariance by designing the lightweight scale-aware pooling module for simplicity and efficiency. So far, the studies of pooling [7], [8], [9] have revealed the limitations of the existing pooling operations in coping with significant scale changes [10]. Consequently, it often suffers from the perspective scale variations in crowd counting scenarios, as shown in Fig. 1. In this case, a pooling module with a larger receptive field is likely to adapt to larger scale variations, and, enabling a 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

S. Huang and Z. Zhang are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (email: siyuhuang@zju.edu.cn, zhongfei@zju.edu.cn).

X. Li (corresponding author) and F. Wu are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (email: xilizju@zju.edu.cn, wufei@cs.zju.edu.cn).

Z. Cheng is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China (email: zhiqicheng@gmail.com).

Alexander Hauptmann is with the School of Computer Science, Carnegie Mellon University, PA, USA (email: alex@cs.cmu.edu).

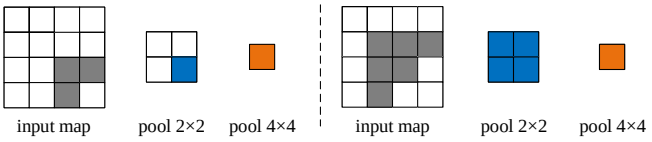

Fig. 2. An intuitive illustration of the scale invariance brought by a larger pooling kernel.

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:

- *Non-parametric:* They do not introduce any extra parameters and hyper-parameters into the model, ensuring a high efficiency in learning.
- *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 about pooling kernel sizes and their impact on the scale variance of CNNs further reveal their effectiveness.

We summarize the contributions of this paper as follows:

- We model the perspective scale invariance in dense crowd images for effective crowd counting by designing the lightweight scale-aware pooling module.
- We propose simple and flexible variants of vanilla pooling layer, i.e., multi-kernel pooling and stacked pooling, to boost the scale invariance of CNNs and improve crowd counting performances.
- We empirically demonstrate the effectiveness of our proposed pooling modules and take insight into their impact on the invariance of CNNs when facing scale variations.

II. Related Work

A. Deep crowd counting

The deep CNNs are currently the state-of-the-art approach [11], [12], [13], [14], [15], [16] for crowd density estimation and crowd counting due to their powerful visual representation abilities.

Specifically to deal with the large scale variations in people size, in the literature the focus is mainly on the improvement of convolution units in recent years [4], [17], [18]. For a typical example, the Multi-Column CNN [5] and Switching CNN [6] exploited multi-sized convolutional kernels to adapt CNNs to people of different sizes. Another popular approach is to transform the scale of the feature map to adapt feature itself to scale variation. For instance, the Hydra CNN [3] adopted a pyramid of multi-scale image patches as input such that each branch of CNN learns the feature representation for a particular scale of the pyramid. [19] proposed a scale-consistency regularization constraint to integrate large-scale and small-scale images.

Different from all of these approaches, this work focuses on the pooling layer, as it is generally assumed that the pooling layer enables the scale invariance of CNNs. Motivated by the significant scale variation in crowd counting, we propose the multi-kernel pooling to take place of the vanilla pooling module, aiming at more scale-invariant CNNs.

B. Variants of pooling

Various variants of pooling have been proposed in the computer vision community [20], [21]. For instance, the well-known L2 pooling [22], [23] is proposed towards the complex invariances of CNNs beyond translational invariance. Hybrid pooling methods [24], [25] combine different types of pooling together into the a network. Stochastic pooling [26], [27] randomly picks the activation in each pooling region obeying a multinomial distribution.

Among variants of pooling, the one most close to this work is the spatial pyramid pooling (SPP) [28], [29]. SPP employs multiple pooling filters followed by concatenation, down-sampling the 2-D feature maps into a fixed-length vector, where the number of pooling filters is fixed and the size of each filter is adapted to the image size. Our multi-kernel pooling is similar to SPP in the manner of fusion of multi-scale pooling regions, but, differing from it mainly in two aspects: 1) SPP is proposed for the use of an alternative to the image cropping and warping operation. Differently, multi-kernel pooling and stacked pooling are proposed towards a boost of scale invariance of CNNs; 2) SPP is often adopted at the top of convolutional layers for the generation of a fixed-length vector for subsequent fully-connected layers. Our proposed pooling layers are more general and can replace the vanilla pooling layers in any CNNs, especially the fully convolutional networks (FCNs) which are the state-of-the-art backbone framework for crowd segmentation, density estimation, and counting.

III. 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. Please note that we take the max pooling as an example in this paper. In practice, our proposed pooling modules are applicable to the other versions of poolings.

A. Vanilla Pooling

The vanilla max pooling $\mathcal{P}_k$ with a kernel size of k can be formulated as

$$\mathcal{P}_k(z) \overset{\text{def}}{=} \max_{\dot{z}\in\kappa(z,k)} X(\dot{z}) \tag{1}$$

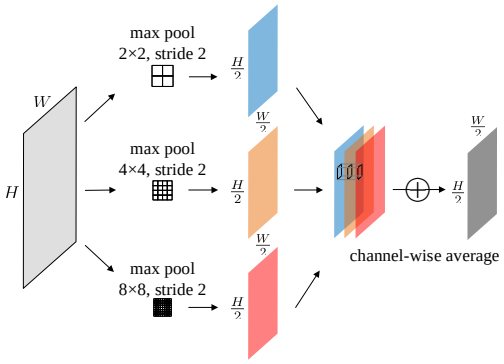

Fig. 3. **Multi-kernel pooling** with a set of kernels $\{2, 4, 8\}$ and a stride of 2. The three pooling kernels are applied on the input feature map and then concatenated with element-wise mean.

where z denotes a pixel position on a feature map $X \in \mathbb{R}^{W \times H}$, and $\kappa(z, k)$ denotes the square neighbourhood of z with a side length of k.

By applying pooling $\mathcal{P}_k$ on feature map X in a manner of sliding window $(*)$, we obtain the feature map after vanilla pooling layer

$$Y_{\text{vanilla}} = X * \mathcal{P}_k \tag{2}$$

B. 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.

The multi-kernel pooling enables a kernel set K comprising of different pooling kernel sizes, such as $K = \{k_1, k_2, ..., k_n\}$. Similar to Eq. 2, we apply the i-th pooling kernel $\mathcal{P}_{k_i}$ on feature map X

$$Y_i = X * \mathcal{P}_{k_i} \tag{3}$$

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. 3, the feature maps are concatenated as

$$Y_{\text{multi-kernel}} = \frac{1}{n} \sum_{i=1}^{n} Y_i \tag{4}$$

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

$$\downarrow_s Y_{\text{multi-kernel}} = \frac{1}{n} \sum_{k \in K} X * \mathcal{P}_k^{(s)} \tag{5}$$

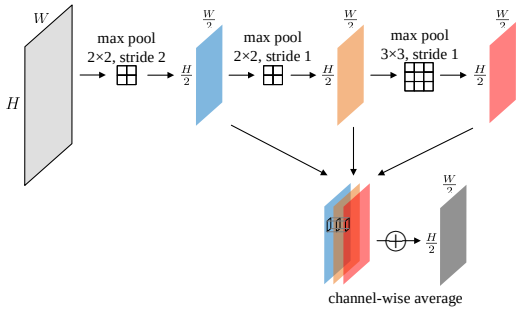

Fig. 4. **Stacked pooling** with a set of kernels $\{2, 2, 3\}$. It is an equivalent form of multi-kernel pooling shown in Fig. 3 with less computing cost.

TABLE I
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

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

C. 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

$$\downarrow_{s_i'} Y_i^{'} = Y_{i-1}^{'} * \mathcal{P}_{k_i'}^{(s_i')} \tag{6}$$

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. 6, the output of stacked pooling concatenates the intermediate feature maps as

$$\downarrow_s Y_{\text{stacked}} = \frac{1}{n} \sum_{i=1}^{n} \downarrow_{s_i'} Y_i^{'} \tag{7}$$

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 I 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.

Fig. 5. **Density maps.** The first row shows the crowd images. From left to right, the images belong to density groups 1 to 5 in ShanghaiTech-A dataset. The second and the third row are density maps output by Deep-Net with vanilla pooling layers and with stacked pooling layers, respectively. Between density maps of the two models, there are obvious differences on their sharpness, robustness to noises (image #2, #3, #5), and robustness to scale variations (image #4, #5).

IV. Experiments

A. Datasets and Metrics

In this work, we conduct empirical studies[1] on two popular benchmark crowd counting datasets: ShanghaiTech [5] and WorldExpo'10 [31], as both datasets are very challenging due to diverse scene types and varying density levels. More details of data preparation can be found in the supplementary material.

We use mean absolute error (MAE) and mean squared error (MSE) to evaluate the performance of different crowd counting methods:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{\text{gt}}|, \quad \text{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(C_i - C_i^{\text{gt}}\right)^2} \tag{8}$$

where C_i is the estimated people count and C_i^{gt} is the ground truth count of the i-th image. N is the number of test images. The MAE metric indicates the accuracy of crowd estimation algorithm, while the MSE metric indicates the robustness of estimation.

B. Network Architectures

We evaluate our proposed pooling modules[2] 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", "M", and "L", coming from the three columns of Multi-Column CNN [5] 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. More details of the backbone architectures can be found in the supplementary material.

C. Qualitative Results

We qualitatively compare vanilla and stacked pooling by visualizing the density maps. Fig. 5 shows the density maps generated by Deep-Net with vanilla pooling layers and stacked pooling layers, respectively. 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:

- *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).
- *Robustness to noises.* For instance, there is an evident error on bottom of density map #3 of vanilla pooling due to the dense textures of the woman's clothes. On top of density map #5 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.
- *Robustness to scale variations.* Within image #4 and #5, there are severe scale variations among different image parts. The corresponding density maps of stacked pooling show more distinct responses on the bottom of images compared with those of vanilla pooling, indicating that the network with stacked pooling enables a better robustness to scale variations. It is in line with the motivation of using stacked pooling.

D. Study on Pooling Kernels

We first empirically study the configuration of pooling kernel set K as shown in Fig. 6. The experiments are conducted

[1]The implementation is available at https://github.com/siyuhuang/crowdcount-stackpool

[2]Unless otherwise specified, we do experiments on stacked pooling as it is numerically equivalent to multi-kernel pooling with a better efficiency.

TABLE II
COMPARISON OF VANILLA POOLING AND STACKED POOLING ON SHANGHAITECH DATASET

	Base-S		Base-M		Base-L		Wide		Deep	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
ShanghaiTech-A										
vanilla	142.42	225.21	121.71	192.74	**114.66**	**176.05**	122.22	198.23	97.63	153.26
stacked	**127.57**	**197.75**	**116.06**	**182.61**	118.94	186.19	**113.71**	**181.52**	**93.98**	**150.59**
ShanghaiTech-B										
vanilla	27.64	49.22	29.45	51.88	**22.67**	**39.51**	28.21	51.70	21.17	39.20
stacked	**22.27**	**41.45**	**26.03**	**46.11**	23.47	44.46	**26.42**	**47.69**	**18.73**	**31.86**

TABLE III
MAE PERFORMANCES ON FIVE TEST SCENES OF WORLDEXPO'10 DATASET

	Scene #1	Scene #2	Scene #3	Scene #4	Scene #5	Average
Wide + vanilla	5.01	**18.96**	**14.76**	21.36	14.57	14.95
Wide + 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
Deep + stacked	**3.26**	**12.39**	**13.97**	31.41	**3.50**	**12.92**

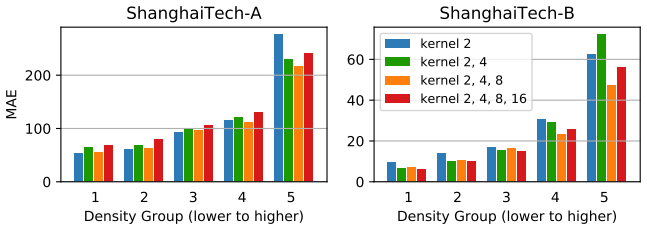

Fig. 6. **Experiments on kernel sizes of poolings.** The MAE, vs. the density groups from lower density to higher density.

by testing Base-M Net on ShanghaiTech dataset. 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 the vanilla pooling performs worse than our multi-kernel pooling on the high 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 configuration of stacked pooling in the following experiments.

E. ShanghaiTech Dataset

We quantitatively compare vanilla pooling and stacked pooling by adopting them in five different CNN architectures. Table II shows the empirical results on ShanghaiTech-A and B, respectively. Stacked pooling obviously outperforms vanilla pooling by showing a superior performance in most settings of datasets, network architectures, and metrics. With regard to datasets, part-A and part-A of ShanghaiTech dataset vary largely with crowd densities, scenes, and camera perspectives. With regard to network architectures, the five evaluated networks cover the commonly used CNN architectures, from small to large, and from shallow to deep. With regard to evaluation metrics, MAE reveals the estimation accuracy of the model, and MSE reveals the robustness of the model. The evidences of improvements in these settings indicate that our stacked pooling module is an effective variant of vanilla pooling module for crowd counting task.

The performances of Deep-Net with different pooling modules are what we care the most, because a deep network is generally effective and most often used in practical crowd counting applications. Table II shows that the Deep-Net is empirically better than Wide-Net and Base-Nets on ShanghaiTech dataset. In this work, we down-sample the feature maps in Deep-Net by three max pooling layers. 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, but at the same time, preserving more information during the down-sampling process, thus benefiting the information flow in deep layers.

F. WorldExpo'10 Dataset

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

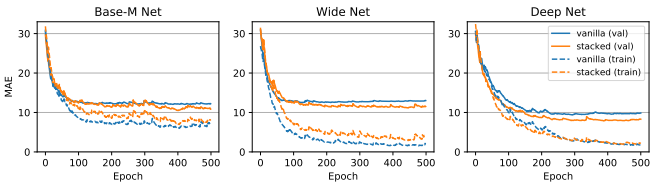

Fig. 7. **Learning curves.** The MAE on training and validation sets, vs. the number of training epochs.

G. Learning Curves

We investigate the training procedure of different pooling modules by studying their learning curves. Fig. 7 shows the training and validation MAEs of the trained models at every epoch, where the learning curves of Base-M Net, Wide-Net, and Deep-Net are shown from left to right, respectively. For better viewing, we smooth the learning curves by applying an exponential moving average (EMA) with a smoothing factor $\alpha = 0.1$.

On the training set, the stacked pooling based models show higher MAEs than the vanilla pooling based models, where the learning curves of Base-M Net and Wide-Net distinctly show this result. In machine learning, model performance on training set generally denotes the fitting degree of a model and the training set. The vanilla pooling shows a better performance on training set and worse performance on testing set, indicating that it has a better fitting capability with a worse generalization capability, such that it may be easier to get overfitting. The MAEs of Deep-Net with the two pooling modules are close to each other after training to convergence, mainly because the Deep-Net model is deeper and larger with more parameters, enabling a better fitting capability. In conjunction with Deep-Net, the stacked pooling also shows a good generalization performance, demonstrating its practicability in real world crowd counting scenarios.

V. 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.

References

[1] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with em routing," in *ICLR*, 2018.
[2] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *CVPR*, 2015, pp. 991–999.
[3] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *ECCV*, 2016, pp. 615–629.
[4] L. Zeng, X. Xu, B. Cai, S. Qiu, and T. Zhang, "Multi-scale convolutional neural networks for crowd counting," in *ICIP*, 2017, pp. 465–469.
[5] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *CVPR*, 2016, pp. 589–597.
[6] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," in *CVPR*, vol. 1, no. 3, 2017, p. 6.
[7] F. J. Huang, Y.-L. Boureau, Y. LeCun *et al.*, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *CVPR*, 2007, pp. 1–8.
[8] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 111–118.
[9] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *ICANN*, 2010, pp. 92–101.
[10] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *ECCV*, 2014, pp. 392–407.
[11] V. A. Sindagi and V. M. Patel, "Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in *AVSS*, 2017, pp. 1–6.
[12] ——, "Generating high-quality crowd density maps using contextual pyramid cnns," in *ICCV*, 2017, pp. 1879–1888.
[13] J. Liu, C. Gao, D. Meng, and A. G. Hauptmann, "Decidenet: Counting varying density crowds through attention guided detection and density estimation," in *CVPR*, 2018, pp. 5197–5206.
[14] D. Babu Sam, N. N. Sajjan, R. Venkatesh Babu, and M. Srinivasan, "Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn," in *CVPR*, 2018, pp. 3618–3626.
[15] H. Li, X. He, H. Wu, S. A. Kasmani, R. Wang, X. Luo, and L. Lin, "Structured inhomogeneous density map learning for crowd counting," *arXiv preprint arXiv:1801.06642*, 2018.
[16] S. Huang, X. Li, Z. Zhang, F. Wu, S. Gao, R. Ji, and J. Han, "Body structure aware deep crowd counting," *IEEE TIP*, vol. 27, no. 3, pp. 1049–1059, 2018.
[17] L. Zhang and M. Shi, "Crowd counting via scale-adaptive convolutional neural network," in *WACV*, 2018.
[18] Y. Li, X. Zhang, and D. Chen, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *CVPR*, 2018, pp. 1091–1100.
[19] Z. Shen, Y. Xu, B. Ni, M. Wang, J. Hu, and X. Yang, "Crowd counting via adversarial cross-scale consistency pursuit," in *CVPR*, 2018, pp. 5245–5254.
[20] Y.-L. Boureau, N. Le Roux, F. Bach, J. Ponce, and Y. LeCun, "Ask the locals: multi-way local pooling for image recognition," in *ICCV*, 2011, pp. 2651–2658.
[21] D. Yoo, S. Park, J.-Y. Lee, and I. So Kweon, "Multi-scale pyramid pooling for deep convolutional representation," in *CVPR Workshop*, 2015, pp. 71–80.
[22] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural image statistics: a probabilistic approach to early computational vision*. Springer, 2009.
[23] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, "Tiled convolutional neural networks," in *NIPS*, 2010, pp. 1279–1287.
[24] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, "Deep multi-patch aggregation network for image style, aesthetics, and quality estimation," in *ICCV*, 2015, pp. 990–998.
[25] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *AISTATS*, 2016, pp. 464–472.
[26] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *ICLR*, 2013.
[27] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. S. Feris, "S3pool: Pooling with stochastic spatial sampling," in *CVPR*, 2017, pp. 4003–4011.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014, pp. 346–361.
[29] ——, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE TPAMI*, vol. 37, no. 9, pp. 1904–1916, 2015.
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
[31] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015, pp. 833–841.

Crowd Counting with Stacked Pooling for Boosting Scale Invariance
Supplementary Material

Siyu Huang, Xi Li, Zhi-Qi Cheng, Zhongfei Zhang, Fei Wu, and Alexander Hauptmann

I. Experimental Details

A. Datasets

In this work, we conduct experiments on two datasets: ShanghaiTech [1] and WorldExpo'10 [2].

- The ShanghaiTech dataset [1] consists of 1198 images with 330,165 annotated heads. It contains two parts: Part A and Part B. Part A consists of 482 images which are randomly chosen from the Internet, having relatively larger crowd densities. Part B consists 716 images taken from the streets of metropolitan areas in Shanghai, having relatively smaller crowd densities. Part A and Part B are separately evaluated in our experiments, denoted as ShanghaiTech-A and ShanghaiTech-B.
- The WorldExpo'10 dataset [2] consists of 1132 annotated video sequences captured by 108 surveillance cameras. It contains a total of 199,923 annotated pedestrians in 3980 images.

B. Data Preparation

In each dataset, we randomly split the original training set into a training set and a validation set by a ratio of 9:1. We randomly crop 9 patches on each training image, where all the patches are half the size of the original image. The ground truth density map is generated by summing a 2D Gaussian kernel with a fixed $\sigma = 4$ centered at every person's position [3], [2]. We use a simple method in order to ensure that the improvements achieved are due to the proposed method and are not dependent on the sophisticated methods for calculating the ground truth density maps.

C. Network Architectures

The configurations of backbone network architectures used in this work are shown in Table I. The convolutional layer parameters are denoted as "⟨kernel size⟩*⟨kernel size⟩, ⟨channels⟩". "pooling" denotes a vanilla/stacked max-pooling layer. The ReLU function and the same padding operation are added after every convolutional layer. "S", "M", "L" represent small, medium, and large convolutional kernel size versions of base network respectively.

D. Learning Details

In this work, the CNNs are implemented based on PyTorch framework [4]. For a fair comparison, we adopt identical learning settings for vanilla pooling and stacked pooling. The Base-Net, Wide-Net, and Deep-Net are trained by an Adam optimizer [5]. The batch size is set as 1 on ShanghaiTech dataset and set as 32 on WorldExpo'10 dataset to ensure a comprehensive evaluation with respect to batch size. The training process runs for 500 epochs on the training set. We evaluate the checkpoints on the validation set at an interval of 2 epochs. The model with the best MAE is selected as the best model used for testing.

TABLE I
Network architecture configurations (shown in columns).

Base			**Wide**	**Deep**
S	M	L		
input image				
5*5, 24	7*7, 20	9*9, 16	7*7, 128	5*5, 64 5*5, 64
pooling				
3*3, 48	5*5, 40	7*7, 32	5*5, 256	5*5, 128 5*5, 128
pooling				
3*3, 24 3*3, 12	5*5, 20 5*5, 10	7*7, 16 7*7, 8	5*5, 128 5*5, 64	3*3, 256 3*3, 256 pooling
1*1, 1	1*1, 1	1*1, 1	1*1, 1	3*3, 128 3*3, 64 3*3, 32 3*3, 16 1*1, 1

II. Study on Scale Invariance

In the paper, we discuss the scale invariance of CNN models, and believe that the pooling layer is one of its most important supporters. Here, we further take some insight into the scale invariance driven by pooling modules. Specifically, we evaluate the variation ratio of feature maps after a pooling layer v.s. the scale variation of an input image. The variation ratio γ is formulated as

$$\gamma = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \frac{\sum |\hat{X}_{wh} - X_{wh}|}{\sum |X_{wh}|} \tag{1}$$

X is a feature map within the feature maps $\mathcal{X}$ of a CNN model given an input image. We resize the input image according to a certain scaling factor β and again calculate the corresponding feature map followed by resizing the feature map to the same size of X. $|\mathcal{X}|$ is the number of feature map channels. The

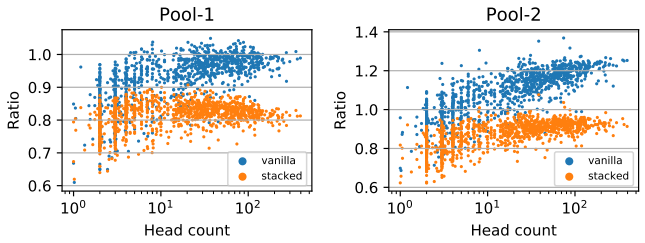

Fig. 1. **The scale invariance of poolings.** The variation ratio of feature maps, vs. the number of head counts.

variation ratio γ is used to evaluate the scale invariance of a CNN model, where a CNN model with a stronger scale-invariant representation has smaller γ when facing the same input image of different scales.

We conduct this experiment by applying Base-M Net to ShanghaiTech-B dataset, where the network is previously trained on the training set and evaluated on the testing set. Fig. 1 shows the variation ratio γ of the feature maps after two respective pooling layers, given the images in the testing set. An up-sample scaling factor $\beta = 2$ is adopted in this experiment. Large data points ($\gamma > 2$) are ignored as outliers.

In Fig. 1, it is distinct that the stacked pooling has a smaller variation ratio γ than the vanilla pooling w.r.t. both pooling layers. It indicates that given the same image of different scales, the stacked pooling layer is able to provide more scale-invariant feature maps for the subsequent convolutional layers, i.e., the feature maps are more consistent with the original feature maps. Such scale-invariant representation improves the generalization capability of a CNN model, especially for crowd counting datasets which have high intra-image and inter-image visual similarities.

It is noticeable that in Fig. 1 the variation ratios γ of the two pooling modules are closer on low-density images while exhibiting greater differences on high-density images. The stacked pooling has much smaller γ than vanilla pooling on high-density images. It indicates that the stacked pooling works particularly well at high-density crowd counting cases. Fig. 5 in the paper also presents this result, where the kernel set $K = \{2, 4, 8\}$ performs much better than a single kernel $K = \{2\}$ on high-density images.

REFERENCES

[1] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *CVPR*, 2016, pp. 589–597.
[2] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *CVPR*, 2015, pp. 833–841.
[3] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *NIPS*, 2010, pp. 1324–1332.
[4] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS Workshop*, 2017.
[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.