Stacked Pooling: Improving Crowd Counting by Boosting Scale Invariance

Siyu Huang[1], **Xi Li**[1], **Zhi-Qi Cheng**[2], **Zhongfei Zhang**[1], **Alexander Hauptmann**[3]
[1]Zhejiang University, [2]Southwest Jiaotong University, [3]Carnegie Mellon University
{siyuhuang,xilizju,zhongfei}@zju.edu.cn, zhiqicheng@gmail.com, alex@cs.cmu.edu

Abstract

In this work, we explore the cross-scale similarity in crowd counting scenario, in which the regions of different scales often exhibit high visual similarity. This feature is universal both within an image and across different images, indicating the importance of scale invariance of a crowd counting model. Motivated by this, in this paper we propose simple but effective variants of pooling module, i.e., multi-kernel pooling and stacked pooling, to boost the scale invariance of convolutional neural networks (CNNs), benefiting much the crowd density estimation and counting. 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 into model and can easily take place of the vanilla pooling layer in implementation. In empirical study on two benchmark crowd counting datasets, the stacked pooling beats the vanilla pooling layer in most cases.

Introduction

Crowd counting has been widely studied for decades of years because of a great many practical demands such as public safety and city planning. While, crowd counting still remains challenging and researchers seek to address it by focusing on aspects of severe occlusions, perspective distortions, and diverse crowd distributions.

In this work, we explore an important feature in crowd counting scenario, i.e., *cross-scale visual similarity*. Fig. 1 shows two examples of the cross-scale visual similarity within crowd images. In each image, it is not difficult to find two regions which are visually similar when they are resized to the same scale. The cross-scale visual similarity is quite universal in crowd counting, not only within an individual image, but also among different images of various scenes. In contrast, this feature is not typical for the natural images such as Cifar-10 (Krizhevsky and Hinton 2009) and ImageNet (Deng et al. 2009). Therefore, the vision model designed for crowd counting especially requires the capability of scale invariance.

This work was done when Siyu Huang and Zhi-Qi Cheng visited Carnegie Mellon University.

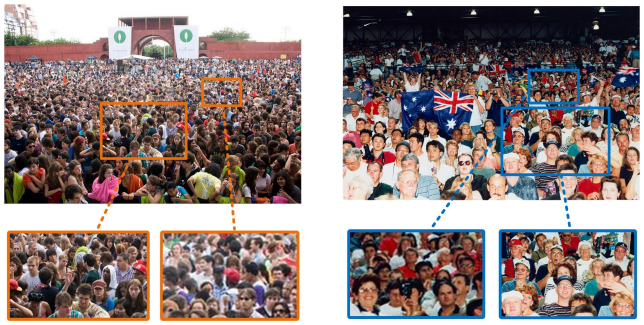

Figure 1: **In crowd images, regions of different scales exhibit high visual similarity if we resize them to certain sizes.** This feature is common for regions within an image and also for regions among different images. It indicates the importance of scale invariance in crowd counting.

A common solution for augmenting scale invariance of convolutional neural networks (CNNs) would be to make CNNs larger (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2015; Huang et al. 2017) and deeper (Szegedy et al. 2015; He et al. 2016) by introducing more learnable parameters to improve their representation performances. Another solution is to manually build branches in CNNs for visual concepts of different scales (Xu et al. 2014; Cai et al. 2016). Specifically in crowd counting, researchers explore various variants (Zeng et al. 2017; Sam, Surya, and Babu 2017) of multi-sized convolutions (Zhang et al. 2016) to deal with the scale variation in people size.

Different from these approaches, we focus on the pooling module to boost the scale invariance of CNNs. As studied by existing literature (Huang et al. 2007; Boureau, Ponce, and LeCun 2010; Scherer, Müller, and Behnke 2010), the scale invariance of CNNs is in general brought by the pooling layer. However, it is evident that the conventional pooling can only deal with slight scale change (Gong et al. 2014), thus cannot well cope with the significant scale variation in crowd counting scenarios, e.g., the examples shown in Fig.1. Motivated by this, we propose to employ a larger pooling range to adapt the network to such a severe scale variation. Fig. 2 illustrates how a larger pooling range enables an in-

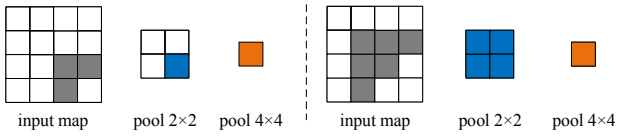

Figure 2: An intuitive illustration of the scale invariance brought by a larger pooling kernel.

variance when the input goes through a scale variation. The feature map after 2×2 max-pooling changes. While, the feature map after 4×4 max-pooling remains unchanged, presenting a scale invariance.

In this paper, we propose simple but effective variants of pooling module, 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, 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:

- *Non-parametric:* They do not introduce extra parameters and hyper-parameters into the model, ensuring the model efficiency and preventing the overfitting 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 when need be.

In empirical study, the stacked pooling shows favorable performance in comparison with the vanilla pooling. It beats the vanilla pooling in most experiments on two benchmark crowd counting datasets. In addition, insight studies about pooling kernel sizes and their impact on the scale variance of CNNs further reveal the effectiveness of stacked pooling.

We summarize the contributions of this paper as follow:

- We explore the cross-scale visual similarity in crowd images to promote the study on the scale invariance of crowd counting models.
- We propose multi-kernel pooling and stacked pooling which are simple and flexible variants of vanilla pooling for boosting the scale invariance of CNNs and improving crowd counting performances.
- We empirically demonstrate the effectiveness of proposed pooling modules and take insight into their impact on the invariance of CNNs when facing scale variation.

Related Work

Deep crowd counting

The deep CNNs are currently the state-of-the-art approach (Sindagi and Patel 2017a; Sindagi and Patel 2017b; Liu et al. 2018; Babu Sam et al. 2018; Li et al. 2018; Huang et al. 2018)for crowd density estimation and crowd counting due to their powerful visual representation ability.

Specifically to deal with the large scale variation in people size, researchers mainly focused on the improvement of convolution units in recent years (Zeng et al. 2017; Zhang and Shi 2018; Li, Zhang, and Chen 2018). For a typical example, the Multi-Column CNN (Zhang et al. 2016) and Switching CNN (Sam, Surya, and Babu 2017) 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 (Onoro-Rubio and López-Sastre 2016) 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. Shen et al. 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.

Variants of pooling

Various variants of pooling have been proposed by the computer vision community (Boureau et al. 2011; Yoo et al. 2015). For instance, the well-known L2 pooling (Hyvärinen, Hurri, and Hoyer 2009; Ngiam et al. 2010) is proposed towards the complex invariances of CNNs beyond translational invariance. Hybrid pooling methods (Lu et al. 2015; Lee, Gallagher, and Tu 2016) combine different types of pooling together into the a network. Stochastic pooling (Zeiler and Fergus 2013; Zhai et al. 2017) 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) (He et al. 2014; He et al. 2015). 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, while, 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 substitute 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.

Our Approach

We introduce very simple yet effective variants of vanilla pooling, i.e., multi-kernel pooling and stacked pooling, to

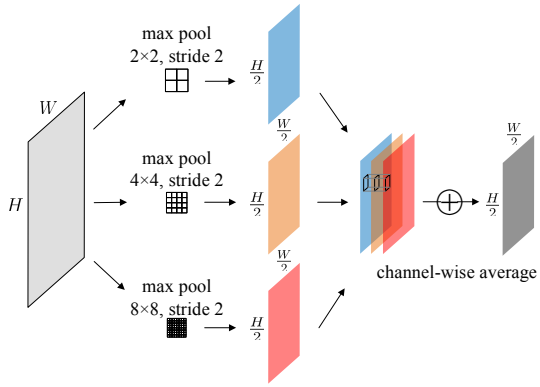

Figure 3: **Multi-kernel pooling** with a set of kernels $\{2, 4, 8\}$ and a stride of 2. The three pooling kernels are applied on the input feature map and then concatenated with element-wise mean.

improve the scale invariance of convolutional neural networks (CNNs). Please note that we take the max pooling as an example in this paper. In practice, our proposed pooling modules are totally compatible with other versions of poolings.

Vanilla Pooling

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

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

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 get the feature map after vanilla pooling layer

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

Multi-Kernel Pooling

In the practice of deep CNNs, a small pooling kernel, etc., $k = 2$, 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 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 $K = \{k_1, k_2, ..., k_n\}$. As same as 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 machine learning tasks; 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

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

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 empirical studies, this configuration also shows the best performance in most cases.

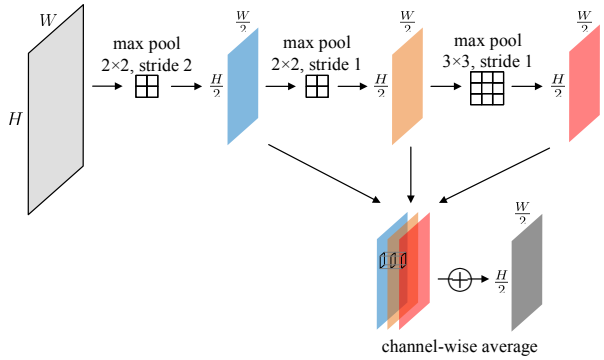

Figure 4: **Stacked pooling** with a set of kernels $\{2, 2, 3\}$. It is an equivalent form of multi-kernel pooling shown in Fig. 3 with less computing cost.

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

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 its first pooling kernel. Table 1 gives the time cost of different pooling methods w.r.t a 256×256 input feature map. We can see that the stacked pooling shows a much better computing efficiency than multi-kernel pooling. On a VGG-13 network (Simonyan and Zisserman 2015), the forward and backward time of stacked pooling is close to that of vanilla pooling, thus, ensuring its practicability.

Experimental Setup

Datasets

In this work, we do empirical study on two popular crowd counting datasets: ShanghaiTech (Zhang et al. 2016) and WorldExpo'10 (Zhang et al. 2015), as both the two datasets are very challenging due to diverse scene types and varying density levels.

- The ShanghaiTech dataset (Zhang et al. 2016) 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 (Zhang et al. 2015) consists of 1132 annotated video sequences captured by 108 surveillance cameras. It contains a total of 199,923 annotated pedestrians in 3980 images.

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 (Lempitsky and Zisserman 2010; Zhang et al. 2015).

Metrics

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.

Table 2: **Network architecture configurations** (shown in columns). 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.

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

Network Architectures

We evaluate our proposed stacked pooling module[1] on different backbone CNNs as shown in Table 2. 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 (Zhang et al. 2016) and having different convolutional kernel sizes. The Wide-Net widen the Base-M Net by using more channels of feature maps. The Deep-Net follows the well-known VGG-13 network (Simonyan and Zisserman 2015) with slight modifications. We use CNNs of diverse depths, widths, and convolutional kernel sizes for a comprehensive evaluation of our method.

Learning Settings

In this work, the CNNs are implemented based on PyTorch framework (Paszke et al. 2017). For a fair comparison,

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

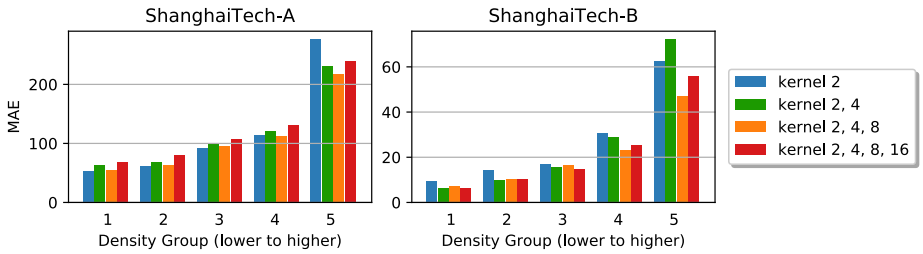

Figure 5: **Experiments on kernel sizes of poolings.** The MAE, vs. the density groups from lower density to higher density.

Table 3: **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	138.56	220.40	128.91	200.24	117.14	181.23	123.11	201.32	99.19	158.88
stacked	**109.41**	**168.49**	**119.48**	**184.73**	**115.35**	**176.43**	**113.71**	**181.52**	**93.98**	**150.59**
ShanghaiTech-B										
vanilla	31.54	55.04	25.96	47.77	23.55	42.91	31.79	57.22	20.34	37.32
stacked	**23.60**	**43.90**	**22.91**	**39.91**	**20.96**	**37.78**	**25.84**	**47.36**	**18.02**	**35.64**

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 (Kingma and Ba 2015). 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. Please refer to our code[2] for more implementation details.

Results

Study on Pooling Kernels

We first empirically study the configuration of pooling kernel set K as shown in Fig. 5. The experiments are conducted on ShanghaiTech dataset and Base-M Net. 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 the crowd densities and show the MAE of density groups from lower density to higher density.

Fig. 5 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.

[2]https://github.com/siyuhuang/crowdcount-stackpool

Vanilla Pooling vs. Stacked Pooling

ShanghaiTech dataset We quantitatively compare vanilla pooling based and stacked pooling by adopting them in five different CNN architectures described in Table 2. Kernel set $K = \{2, 2, 3\}$ is used for stacked pooling.

Table 3 shows the empirical results on ShanghaiTech-A and B, respectively. stacked pooling obviously outperforms vanilla pooling by showing a superior performance over all settings of datasets, network architectures, and metrics. In regard to datasets, A part and B part of ShanghaiTech dataset vary largely with crowd densities, scenes, and camera perspectives. In regard to network architectures, the five evaluated network cover the common-used CNN architectures, from small to large, and from shallow to deep. In regard to evaluation metrics, MAE reveals the estimation accuracy of model, and MSE reveals the robustness of model. The evidences of improvements over these settings indicate that our stacked pooling module is a universally effective variant of vanilla pooling module for crowd counting task.

The performance of pooling module on Deep-Net is what we most care, because a deep network is generally effective and is the most often used in practical crowd counting applications. Table 3 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 5.2% and 11.4% better under MAE by adopting stacked pooling instead of vanilla pooling. In theory, the stacked pooling does not introduce extra model parameters, meanwhile, preserving more information during the down-sampling process, thus benefiting the information flow in deep layers.

Table 4: **Comparison of vanilla pooling and stacked pooling on WorldExpo'10 dataset.** We show the MAE performances on five test scenes.

	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**

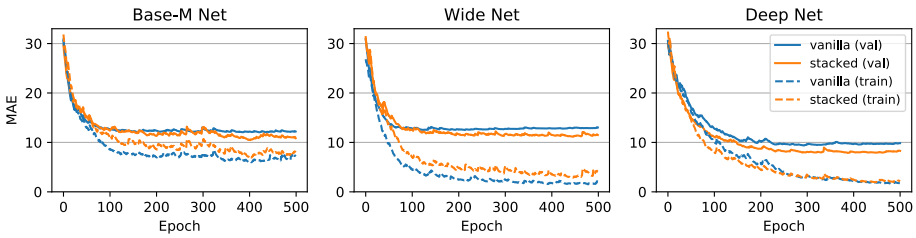

Figure 6: **Learning curves.** The MAE on training and validation sets, vs. the number of training epochs.

WorldExpo'10 dataset Table 4 compares 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 Deep-Net still performs better than the Wide-Net w.r.t. the average MAE. The MAEs across different scenes are quite different due to diverse crowd densities of the scenes.

The stacked pooling performs better than the vanilla pooling w.r.t. the average performance and most of the testing scenes. The stacked pooling performs well on low-density scenes while showing similar performance with vanilla pooling on high-density scenes, indicating that the stacked pooling is as a whole better than the vanilla pooling for crowd images with diverse densities and various scenes.

Learning curves We investigate the training procedure of different pooling modules by studying their learning curves. Fig. 6 shows the training and validation MAEs of 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 compared to 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 better performance on training set and worse performance on testing set, indicating that it has a better fitting capability, while, 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.

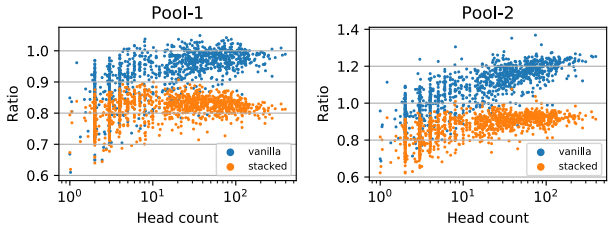

Figure 7: **The scale invariance of poolings.** The variation ratio of feature maps, vs. the number of head counts.

Study on Scale Invariance

In previous sections, we discuss the scale invariance of CNN models, and, believe the pooling layer is one of its most important supporters. In this section, 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

$$\gamma = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \frac{\sum |\hat{X}_{wh} - X_{wh}|}{\sum |X_{wh}|} \tag{9}$$

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 variation ratio γ is used to evaluate the scale invariance of a CNN model, where a CNN model with a stronger scale-invariant representation will have 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. 7 shows the variation ratio γ of 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. 7, 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 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. 7 the variation ratios γ of 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 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.

Conclusion

In this work, we have explored an important feature in crowd counting scenario, i.e., cross-scale visual similarity, to highlight the importance of scale invariance of crowd counting models. Further, we have proposed multi-kernel pooling and stacked pooling to boost the scale invariance of CNNs, where a larger pooling range enables a stronger invariance for significant scale variation in crowd images. The stacked pooling layer is efficient and easy to implement, showing better performance than vanilla pooling layer in most cases on benchmark crowd counting datasets.

References

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

[Boureau et al. 2011] Boureau, Y.-L.; Le Roux, N.; Bach, F.; Ponce, J.; and LeCun, Y. 2011. Ask the locals: multi-way local pooling for image recognition. In *ICCV*, 2651–2658.

[Boureau, Ponce, and LeCun 2010] Boureau, Y.-L.; Ponce, J.; and LeCun, Y. 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 111–118.

[Cai et al. 2016] Cai, Z.; Fan, Q.; Feris, R. S.; and Vasconcelos, N. 2016. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 354–370.

[Deng et al. 2009] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255.

[Gong et al. 2014] Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 392–407.

[He et al. 2014] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 346–361.

[He et al. 2015] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI* 37(9):1904–1916.

[He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

[Huang et al. 2007] Huang, F. J.; Boureau, Y.-L.; LeCun, Y.; et al. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 1–8.

[Huang et al. 2017] Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*, volume 1, 3.

[Huang et al. 2018] Huang, S.; Li, X.; Zhang, Z.; Wu, F.; Gao, S.; Ji, R.; and Han, J. 2018. Body structure aware deep crowd counting. *IEEE TIP* 27(3):1049–1059.

[Hyvärinen, Hurri, and Hoyer 2009] Hyvärinen, A.; Hurri, J.; and Hoyer, P. O. 2009. *Natural image statistics: a probabilistic approach to early computational vision*. Springer.

[Kingma and Ba 2015] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[Krizhevsky and Hinton 2009] Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

[Lee, Gallagher, and Tu 2016] Lee, C.-Y.; Gallagher, P. W.; and Tu, Z. 2016. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 464–472.

[Lempitsky and Zisserman 2010] Lempitsky, V., and Zisserman, A. 2010. Learning to count objects in images. In *NIPS*, 1324–1332.

[Li et al. 2018] Li, H.; He, X.; Wu, H.; Kasmani, S. A.; Wang, R.; Luo, X.; and Lin, L. 2018. Structured inhomogeneous density map learning for crowd counting. *arXiv preprint arXiv:1801.06642*.

[Li, Zhang, and Chen 2018] Li, Y.; Zhang, X.; and Chen, D. 2018. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, 1091–1100.

[Liu et al. 2018] Liu, J.; Gao, C.; Meng, D.; and Hauptmann, A. G. 2018. Decidenet: Counting varying density crowds through attention guided detection and density estimation. In *CVPR*, 5197–5206.

[Lu et al. 2015] Lu, X.; Lin, Z.; Shen, X.; Mech, R.; and Wang, J. Z. 2015. Deep multi-patch aggregation network for image style, aesthetics, and quality estimation. In *ICCV*, 990–998.

[Ngiam et al. 2010] Ngiam, J.; Chen, Z.; Chia, D.; Koh, P. W.; Le, Q. V.; and Ng, A. Y. 2010. Tiled convolutional neural networks. In *NIPS*, 1279–1287.

[Onoro-Rubio and López-Sastre 2016] Onoro-Rubio, D., and López-Sastre, R. J. 2016. Towards perspective-free object counting with deep learning. In *ECCV*, 615–629.

[Paszke et al. 2017] Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS Workshop*.

[Sam, Surya, and Babu 2017] Sam, D. B.; Surya, S.; and Babu, R. V. 2017. Switching convolutional neural network for crowd counting. In *CVPR*, volume 1, 6.

[Scherer, Müller, and Behnke 2010] Scherer, D.; Müller, A.; and Behnke, S. 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *ICANN*. 92–101.

[Shen et al. 2018] Shen, Z.; Xu, Y.; Ni, B.; Wang, M.; Hu, J.; and Yang, X. 2018. Crowd counting via adversarial cross-scale consistency pursuit. In *CVPR*, 5245–5254.

[Simonyan and Zisserman 2015] Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

[Sindagi and Patel 2017a] Sindagi, V. A., and Patel, V. M. 2017a. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *AVSS*, 1–6.

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

[Szegedy et al. 2015] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9.

[Xu et al. 2014] Xu, Y.; Xiao, T.; Zhang, J.; Yang, K.; and Zhang, Z. 2014. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*.

[Yoo et al. 2015] Yoo, D.; Park, S.; Lee, J.-Y.; and So Kweon, I. 2015. Multi-scale pyramid pooling for deep convolutional representation. In *CVPR Workshop*, 71–80.

[Zeiler and Fergus 2013] Zeiler, M. D., and Fergus, R. 2013. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*.

[Zeng et al. 2017] Zeng, L.; Xu, X.; Cai, B.; Qiu, S.; and Zhang, T. 2017. Multi-scale convolutional neural networks for crowd counting. In *ICIP*, 465–469.

[Zhai et al. 2017] Zhai, S.; Wu, H.; Kumar, A.; Cheng, Y.; Lu, Y.; Zhang, Z.; and Feris, R. S. 2017. S3pool: Pooling with stochastic spatial sampling. In *CVPR*, 4003–4011.

[Zhang and Shi 2018] Zhang, L., and Shi, M. 2018. Crowd counting via scale-adaptive convolutional neural network. In *WACV*.

[Zhang et al. 2015] Zhang, C.; Li, H.; Wang, X.; and Yang, X. 2015. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 833–841.

[Zhang et al. 2016] Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; and Ma, Y. 2016. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 589–597.