Crowd Counting in Images via DSMCNN

Yu-qian ZHANG\textsuperscript{1,*}, Wen-qian WANG\textsuperscript{1}, Tao WANG\textsuperscript{2}, Guo-hui LI\textsuperscript{1} and Jun LEI\textsuperscript{1}

\textsuperscript{1}School of Systems Engineering, National University of Defense Technology, Changsha, P.R. China
\textsuperscript{2}31104 Troops of PLA, Nanjing, P.R. China

*Corresponding author

Keywords: Crowd density map, DSMCNN model, Stacked pooling, Dilated layer.

Abstract. Counting the crowd from a single image accurately is always a challenging task. In this work, we propose a Dilated Stacked Multi-column Convolutional Neural Network architecture for crowd density estimation in still single images. The model is composed of three columns of convolutional layers with sharing layers. We use smaller kernel and the dilated layer. We stack multifarious pooling layers and optimize the loss function. The DSMCNN model is an end-to-end and easy-trained system. Meanwhile, it shows robust for images with different perspective or crowd density. We demonstrate experiments on the ShanghaiTech dataset and the mall dataset.

Introduction

Crowd scene analysis has become an important security technique with people pouring into modern cities. Some extremely congested scenes are shown in figure 1. CNN-based works \cite{1}, \cite{5} in recent years were proposed and have achieved significant improvements in crowd density estimation. But most of these methods suffer drawbacks. Deeper convolution may further shrunk the output size, hard to generate high-quality density maps. Moreover, the pooling layer dramatically reduce the spatial resolution so the spatial information of feature map is lost. Meanwhile, the traditional Euclidean loss have some defects \cite{7} like image blur and sensitivity to outliers. When the predicted value differs greatly from the target value, the gradient of the Euclidean Loss may result in gradient explosion.

In this paper, we propose a novel Convolutional Neural Network for crowd density estimation and crowd counting aimed at address these problems. Our DSMCNN model could generate a high-quality density map and count pedestrians accurately. Moreover, it is extraordinarily convenient to optimize. The summary of our foremost work are as follows:

1. We apply dilated convolution to the network. The dilated convolution can enlarge the receptive field and reduce the spatial resolution rather than increasing parameters or computation amounts. To solve the sensitivity to outliers and image blur caused by traditional Euclidean loss, we add the Manhattan loss to the final loss function.

2. We use a simple but effective variants of pooling module, stacked pooling. We stack the multi-kernel pooling layers with multiple receptive fields then concatenating the feature maps together to its successive layer. The larger pooling kernels can provide a wider range of scale invariance, while the smaller pooling kernels can also preserved fine-grained information.

3. We propose the Dilated Stacked Multi-column Convolutional Neural Network (DSMCNN) architecture. Owing to the smaller kernels, shared parameters and $1 \times 1$ kernels replacing fully connected layers, our model has less convolutional parameters. There is no need to split the model into three columns and pre-train each column independently. See details in 3.3.
Related Work

Crowd counting first via head detections has been tackled by [14] using motion cues and appearance features to train detectors. But crowd counting using head detections has limitations due to high occlusion between people in dense crowded scenes. Early works in still image crowd counting employ a combination of handcrafted features, they use HOG based detections, interest points based counting and Fourier analysis. These weak representations based on local features are outperformed by modern deep representations. [10] trains the CNNs to regress the crowd density map. They retrieve images from the training data similar to a test image using density and perspective information as the similarity metric. The trained network is then fine-tuned using the retrieved images for a specific target test scene, and finally outputs the density map. However, fine-tuning is required for each test scene and perspective maps are not always available resisting its applicability.[12] uses a multiscale CNN architecture to tackle the large scale variations in crowd scenes. They train separately for each scale, from which fully-connected layers are used to fuse the maps. However, it is sensitive to the number of levels in the image pyramid as indicated by performance across datasets. [1] use multi-column CNN and perform fusion of features from different CNN columns. Shallow CNN columns with different receptive fields capture the large variation in scale and perspective in crowd scenes. It fuses the feature maps from the CNN columns by weighted averaging via a 1 × 1 convolutional layer to predict the density map. However, it needs to split the model into three columns and train each column separately before the training process, which is cumbersome.

Proposed Approach

Dilated Convolution

Segmentation tasks have utilized dilated convolutional layers for significant improvement of accuracy [2]. Although deconvolutional layers can alleviate the loss of information, it would increase additional complexity and execution latency. Dilated convolution is a better choice to alternate the convolutional and pooling layer. This character enlarges the receptive field without increasing the parameters or the computation amounts. As shown in figure 2, the 3 × 3 convolution kernel with an expansion rate of 2, the receptive field is the same as the 5 × 5 convolution kernel, and only requires 9 parameters. Thus it allows flexible aggregation of the multi-scale contextual information while keeping the same resolution.

The input of crowd counting is an image of crowds. Traditionally, input is passed to a series of convolutional layers then downsampled by a max pooling layer with factor 2. After that the generated feature map is only 1/2 of the original input so it needs to do deconvolution (bilinear interpolation) for upsampling. But if we try dilated convolution and adapt the same kernel to a dilated kernel with a stride of 2, the output is shared the same dimension as the input without pooling and deconvolutional layers. Most importantly, the output from dilated convolution contains more detailed information.

In the architecture table of our model (Table 1), the dilated stride is represented by the fourth parameter of the convolutional layer (Conv). When the stride is set as 1, the layer is just like the normal convolutional layer. And when it turns to 2, the spacing between kernel elements is 2.
Stacked Pooling Layers

[4] proposes that the conventional pooling can only deal with slight scale change, thus is hard to cope with the significant scale variation in crowd counting scenarios. In figure 3, the feature maps after a 2×2 max-pooling are different with different input scales. Meanwhile, the feature maps after a 4×4 max-pooling do not change. It illustrates that a larger pooling range enables an invariance when the input goes through a scale variation.

Motivated by [5], we use stacked pooling to replace the single max pooling. The stacked pooling is an equivalent form of multi-kernel pooling by stacking smaller pooling kernels. As shown in Figure 4, the multi-kernel pooling comprises of pooling kernels with multiple receptive fields. It captures the responses at multiscale local ranges and concatenates the feature maps to its successive layer. The stacked pooling is a stack of pooling layers, where the feature map $F'_{i}$ are consecutively computed as:

$$F'_{i} = F'_{i-1} \ast p^{(s)}_{k_{i}}$$

(1)

Where $F'_{0} = X$ is the input feature map. $k_{i}$ is the kernel size, which corresponds to $k_{i}$ with a certain transformation. $s_{i}$ is the stride, $s_{i} = s$ and $s_{i+1} = 1$. The final output of $F_{i}$ is as follows:

$$F_{i} = \frac{1}{n} \sum_{i=1}^{n} F'_{i}$$

(2)

The stacked pooling is succinct and very easy to implement. Particularly, it is significant efficient owing to its pooling operations are computed on down-sampled feature maps, except its first pooling kernel. On the one hand, a wider range of scale invariance can be provided for CNNs by the larger pooling kernels, on the other hand, the fine-grained information is also preserved by smaller pooling kernels. The stacked pooling shows favorable performance, meanwhile it further reduces the computing cost of multi kernel pooling.

Dilated Stacked MCNN Architecture

Table 1 shows details of our crowd counting architecture. The architecture improves from MCNN [1]. There follows the advantages of our architecture:

Smaller Kernels and Deeper Network. MCNN [1] uses filters with the sizes of 5 × 5, 7 × 7 and 9 × 9. These relatively large kernels need more network parameters, so that it calls for more computational resource. VGG-16 [12] proposes that two sequential 3 × 3 kernels could achieve one 5 × 5 kernel’s local receptive field. By this way, it reduces the model parameters and computational expense. So we decompose the large kernels with several 3 × 3 filters. Similarly, three 3×3 kernels are used to replace each 7 × 7 kernel and four 3 × 3 kernels to replace each 9 × 9 kernel. In this way, there are only 3 × 3 filters in our network.

All the small kernels are followed by ReLU. It is most important that activation function can make the network deeper and bring more non-linear character. ReLU works well in convolutional neural networks as an activation function and can enhance restoration. Its activation value is positive number, the same as the value of density map.
**Parameters Sharing.** The network includes three deeper CNN columns. Obviously, each column has two $3 \times 3$ convolutional layers. As they all extract low-level features from input images, we set all three columns share these two convolutional layers, reducing the parameters of network and simplifying the network structure. As shown in Table 1, we set the $5 \times 5$ column in MCNN as C1, $7 \times 7$ column as C2, and $7 \times 7$ column as C3. C2 and C3 can share Conv_3 after Conv_1 and Conv_2 (Conv_i represents the convolutional layer i). After that, we also set C2 and C3 share Conv_4 and Conv_5. Moreover, DepthConcat_0 and DepthConcat_1 are formed to generate an Inception-like structure motivated by GoogLeNet [11]. We concat the C2 and C3 before Conv_4. Therefore DepthConcat_0 concats features from C2 and C3, DepthConcat_1 concats features from C1, C2 and C3.

**Fitting Ability Promotion.** [10] uses fully connected layers to generate the density maps of input patches. However, the inputs of our model are original images which are in larger sizes. The model will be much more heavy and slow fully connected layers directly used. As the viewpoint of some brilliant papers such as NIN [13], convolutional layers with $1 \times 1$ kernels can achieve the same property of fully connected layer. As shown as Conv_fc_0 and Conv_fc_1 in Table 1, $1 \times 1$ kernels can not only obtain the similar fitting ability of fully connected layers but also avoid the high computational expense and slow training process. We also use $1 \times 1$ kernels to merge the feature map into a density map called Conv_merge.

|       | C1   | C2   | C3   |
|-------|------|------|------|
| Input |      |      |      |
|       |      |      |      |
| MaxPool | Conv_1$\rightarrow$3x3x64x1 | Conv_2$\rightarrow$3x3x48x1 |
| Conv(*2)$\rightarrow$3x3x48x2 | MaxPool | Conv(*)$\rightarrow$3x3x28x2 |
| StackedPool | Conv(*)$\rightarrow$3x3x36x2 | MaxPool |
| Conv(*2)$\rightarrow$3x3x48x2 | StackedPool | Conv(*)$\rightarrow$3x3x28x2 |
| Conv(*2)$\rightarrow$3x3x36x2 | StackedPool | Conv(*)$\rightarrow$3x3x28x2 |
| DepthConcat_0 | Conv_4, Conv_5$\rightarrow$3x3x48x2 |
| DepthConcat_1 | Conv_6, Conv_7$\rightarrow$3x3x48x2 |
| Conv_fc_0$\rightarrow$1x1x1024 | | |
| Conv_fc_1$\rightarrow$1x1x320 | | |
| Conv_merge$\rightarrow$1x1x1 | | |
| Output | | | |

**Ground Truth Generation**

We generate the ground truth of density maps based on the geometry-adaptive kernels [1]. The geometry-adaptive kernel is defined as:

$$F(x) = \sum_{i=1}^{N} \delta(x - x_i) \times G_{\sigma_i}(x), \sigma_i = \beta d_i$$

(3)

For each targeted object $x_i$ in the ground truth $\delta$, $d_i$ is the average distance of $k$ nearest neighbors. To generate the density map, we convolve $\delta(x - x_i)$ as a Gaussian kernel with parameter $\sigma_i$ (the standard deviation), where $x$ is the position of pixel in the image. In experiment, we follow the
configuration in [1] and set $\beta=0.3, k=3$. For input with sparse crowd, we adapt the Gaussian kernel to the average head size to blur all the annotations.

**Model Optimization**

[7] proves the traditional Euclidean loss have some defects like image blur and sensitivity to outliers. So we combine the Euclidean distance ($L_2$ Loss) and Manhattan distance ($L_1$ Loss) to measure the difference between the output density map and the corresponding ground truth. $L_1$ Loss is less sensitive to outliers than $L_2$ Loss. When the predicted value differs greatly from the target value, the gradient of $L_1$ Loss is constant. Adding $L_1$ Loss can prevents gradient explosions when the predicted value is extremely different from the target value suffering from $L_2$ loss.

The final loss function is defined as follows:

$$L_\Theta(\Theta) = \frac{1}{N} \sum_{i} \| F_d(X_i; \Theta) - D_i \|^2 + \frac{1}{N} \sum_{i} \| F_d(X_i; \Theta) - D_i \|$$

(4)

Where $\Theta$ is the set of the model parameters and $N$ is the number of training samples. $F_d(X_i; \Theta)$ represents the output estimated density map of the input image $X_i$ with parameters $\Theta$, while $D_i$ is the ground truth density map.

**Experiment**

We demonstrate our approach in four worldwide datasets: the ShanghaiTech dataset and the mall dataset. Our training and evaluation are performed on the PyTorch framework. In the training phase, we use Adam solver algorithms with a mini-batch size of 1 at a fixed constant momentum value of 0.9. And the learning rate is $1e^{-5}$. 1/10 of the training data is for validation to evaluate the performance and help optimize hyper parameters. There is no splitting, pre-training and fine-tuning. So the whole process is much simple and convenient.

**Evaluation Metric**

Most of the existing works such as [1] use two metrics to estimate the accuracy for crowd counting, the mean absolute error (MAE) and the mean squared error (MSE). The definition is as follows:

$$MAE = \frac{1}{N} \sum_{i} \left| z_i - p(z_i) \right|$$

(5)

$$MSE = \sqrt{\frac{1}{N} \sum_{i} (z_i - p(z_i))^2}$$

(6)

Where $N$ is the number of test images, $z_i$ and $p(z_i)$ are separately the actual and the estimated number of crowd in the i image.

**Ablation Study**

To validate the efficacy of the DSMCNN model, we perform several ablation experiments on ShanghaiTech Part A dataset. The comparison is shown in Figure 5, where our improvements get a better evaluation results.

**The Sharing Layers.** We do experiment on taking apart the input sharing layers to three divided columns and compare the result with our proposed model. It can be seen from the figure that the result on our model is better than the model with divided layers.

**The stacked pooling layer.** We compare the training disparity between using the stacked pooling layers and max pooling layers. As we can see in the figure, the result of using stacked pooling performs better.
The Dilated Convolution. We also do the experiments paying attention to the dilation. To inspect the effect of using the dilated convolution, we utilize the dilated convolution in different layers meanwhile to find the best configurations. Firstly we try to replace all the layers to dilated convolution, which shows a slightly worse result. This may reflect that in the first serval layers, small kernels are needed to capture low-level features. Then we keep the previous sharing layers as regular convolutions, and replaced the convolutional layers of the following three columns to dilated layers. It works better. The results reflect that dilated convolution achieves better effect than the regular convolution. Furthermore, it also performs faster as it decreases the amount of parameters and computation.

The L1 Loss Function. To incept the effect of the loss function, we only using the L2 loss function to train our model. And the accuracy compared with the combination loss function is worse. Since L1 loss function could solve some of the problems for single L2 loss function mentioned in sec 3.5, both of the MAE and MSE are improved, separately by 5.7% and 8.3%.

The ShanghaiTech Dataset
The ShanghaiTech crowd counting dataset introduced by [1] has two parts: Part A and Part B. The scenes collected from the Internet in Part A are extremely dense. The number of people varys from 33 to 3139. While Part B taken from busy streets in Shanghai is less dense with crowd counts from 9 to 578. In Part A 300 images are set for training data and the remaining 182 images for testing. In Part B 400 images are for training and 316 for testing.

Result comparisons of MAE and MSE are listed in Table 2. And figure 1 shows some density predictions of our method along with the input images in Part B. Obviously our method outperforms all other models by a significant margin both in MAE and MSE on the part A. And it slightly performs worse on the part B than the Cascaded-MTL model in MSE. It indicates that our model works well in the dense crowd scenes.

| Dataset          | Part A    | Part B    |
|------------------|-----------|-----------|
| Method           | MAE | MSE | MAE | MSE |
| Zhang et al. [10] | 181.8 | 277.7 | 32.0 | 49.8 |
| MCNN [1]         | 110.2 | 173.2 | 26.4 | 41.3 |
| Cascaded-MTL [6] | 101.3 | 152.4 | 20.0 | 31.1 |
| Our model        | 99.09 | 150.76 | 20.0 | 35.3 |

The Mall Dataset
The Mall dataset [3] contains 2000 frames, collected in a shopping mall. Each frame has a fixed resolution of 320×240. We follow the pre-defined settings to use the first 800 frames as the training dataset and the rest 1200 frames as the test dataset. The validation set is selected randomly from 100 images in the training set. We compare our model with some approaches that did experiments in this dataset. The evaluation results are exhibited in Table 3. Our model shows the beyond compare results compared to other models demonstrating its excellence.
Conclusion

In this paper, we propose a Dilated Stacked Multi-column Convolutional Neural Network (DSMCNN) model to estimate crowd density distribution and crowd count accurately in images from different scenes and regions. We conduct experiments on the proposed model in common datasets. The result outperforms most of the state-of-the-art methods. It reveals the model is accurate, robust and widely used. Our model also converges and utilizes easily and can be used as deep feature extractor. It is much more accuracy and easy to use in practical application.

Table 3. Estimation errors on the mall dataset.

| Method           | MAE | MSE |
|------------------|-----|-----|
| Faster R-CNN[8]  | 5.91| 6.60|
| Count Forest[9]  | 4.40| 2.40|
| Our model        | 3.86| 4.70|

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 71673293) and (No.61806215).

References

[1] S. C. S. G. Y. Zhang, D. Zhou and Y. Ma, “Single image crowd counting via multi-column convolutional neural network,” in IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 589–597.

[2] I. K. K. M. L. C. Chen, G. Papandreou and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, PP (99):11, 2017, pp. 1–1.

[3] S. G. K. Chen, C. C. Loy and T. Xiang, “Feature mining for localised crowd counting,” in BMVC, 2012.

[4] Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. “Multi-scale orderless pooling of deep convolutional activation features” in ECCV, 392–407, 2014.

[5] Siyu Huang, Xi Li1, Zhi-Qi Cheng, Zhongfei Zhang, Alexander Hauptmann. “Stacked Pooling: Improving Crowd Counting by Boosting Scale Invariance.” in arXiv: 1808.07456,2018.

[6] 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, IEEE, 2017, pp. 1–6.

[7] A. A. J. Johnson and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in European Conference on Computer Vision, 2016, p. 694711.

[8] R. G. S. Ren, K. He and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015.

[9] O. Y. V.-Q. Pham, T. Kozakaya and R. Okada, “Count forest: Covoting uncertain number of targets using random forest for crowd density estimation,” in ICCV, 2015.

[10] X. W. X. Y. C. Zhang, H. Li, “Cross-scene crowd counting via deep convolutional neural networks.” IEEE, 2015, pp. 833–841.

[11] Y. J. P. S. R. D. A.-D. E. V. A. R. C. Szegedy, W. Liu, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[12] A. Z. K. Simonyan, “Very deep convolutional networks for largescale image recognition.”
[13] S. Y. M. Lin, Q. Chen, “Network in network,” in arXiv preprint arXiv: 1312.4400.

[14] B. Wu and R. Nevatia, “Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors,” in IEEE, 2005, p. 9097.