Deep Density-aware Count Regressor

CHEN ZHUOJUN georgechenzj@outlook.com

Abstract

*We seek to improve crowd counting on both prediction accuracy and time efficiency by not limiting on the current prevalent counting by density map estimation approach but to propose a novel deep CNN that focuses on prediction of a global count besides efficiently leveraging advantages of density maps and optionally producing them. We introduce multilayer gradient fusion for training a density-aware count regressor. By taking advantage of such method, our model outperforms the state-of-the-art methods with 27.5%, 2.7% and 14.3% lower MAE on UCF-QNRF, Shanghai Tech Part A and Part B datasets respectively. Our code and models will be publicly available at: https://github.com/****.*

1. Introduction

Crowd counting is a task to count people in image. It is mainly used in real-life for automated public monitoring such surveillance and traffic control, though there are more applicable extensions proposed such as cell counting, vehicle counting or flock counting. For its versatile potentialities, crowd counting has recently drawn much more attention from computer vision researchers and has been in a significant progress in recent years [4,5,6,7]. However, the task is riddled with many challenges due to the presence of various complexities such as non-uniform density, intra- and inter-scene variations in scale and perspective, and cluttering [8, 9]. Figure 1 shows some of those challenging scenarios.

Early methods [1,2] attempt to solve crowd counting problem by detecting each individual pedestrian in a crowd, while some methods [3] rely on hand-crafted features from multi-source. These methods often perform poorly in the face of the above-mentioned conditions. The recent development of crowd counting comes from DNN-based methods, which have achieved commendable performance. These methods [4,5,6,7] concentrate on generating the demanding density maps before integrating them to the count. They are therefore categorised into density map-based methods. Yet, density maps have yet, in effect, to shown too much importance in practice except for

Figure 1. Representative images for challenges of non-uniform density, intra- and inter-scene variations in scale and perspective, and cluttering.

opportunely provision of demonstration, but are expensive to compute and their quality is difficult to guarantee. Meanwhile, methods that regress the global count directly has remained untouched for a while.

There is evidence in [29] suggesting that direct count regression may have comparable performance to density map-based methods. But there is no reason so far for one to deny that density maps do contribute to the improvement of count prediction, raising state-of-the-art performance in many works. One advantage of density map-based methods may be that information with respect to location, scale alike is fed to the network through supervision. Consequently, they usually adopt multi-scale or multi-column architectures to fuse the features from different scales to capture this information. However, there exists two main drawbacks: first, computational cost drastically increases along with the growth of number of columns; second, useful information learned by the low-level detectors might be lost through forward propagation. Likewise, the error information contained in gradients would be attenuated through backward propagation, making it difficult for low-level detectors to learn.

To address these problems, we propose a novel Gradient Fusion based model called DeepCount (network architecture shown in figure 2) for crowd counting, making efforts to both avert expenditure of multi-column architecture and improve precision. As in figure 2, our proposed model contains a backbone network with convolutional layers deeply regressing a global count. Additionally, five auxiliary modules branch out from the backbone to learn from density maps and to feed gradients back to the backbone. Each branch has different depths and independent parameters, which could help the network to learn the features from different scales. Moreover, each branch can directly access different levels of backbone, and thus to inculcate knowledge to the backbone to make it more perceptive on the density distribution of the image, namely to be density-aware.

In inference deployment, the backbone network can be used unaccompanied to efficiently predict the global count or, if needed, with an auxiliary branch to also visualise a density map. Expensive computation is taken out, but with functionality promised.

By taking advantage of such method, extensive experimental results on four benchmark datasets demonstrate significant improvements of our method against the state-of-the-art methods on Shanghai Tech Part A, Part B and UCF-QNRF datasets and excellent performance on the Mall dataset.

The rest of the paper is structured as follow. We reviewed works for crowd counting in Section 2. Section 3 provides the detailed interpretation of our method. Section 4 reports experiment results. In section 5, we further discuss our findings. We conclude the paper in Section 6.

2. Related Works

2.1. Detection based-methods

Early crowd counting methods tended to rely on counting-by-detection. With sliding window approach, low-level hand-crafted features [1, 22, 34] such as Histograms of Oriented Gradients, silhouette-oriented features were exploited for traditional classifiers, such as Support Vector Machine and Random Forest [10, 12]. Following are CNN-based methods (e.g. Faster R-CNN [28]) which have shown credible detection precision [23]. Nonetheless, in such times when the subject of crowd counting is more on the stage of pedestrian detection, performances on highly dense crowd scenes of these methods are similarly limited.

2.2. Count Regression-based Methods

Count regression-based methods were proposed to overcome limits encountered by detection-based methods. The idea of these method is to regress a global count from the input image. There are methods using ridge regression [13, 14], log-linear regression [17] or MLP [20] on low-level hand-crafted features to estimate the count. While these methods work satisfactorily on invariant scenes of sparse density, hand crafted features can hardly represent

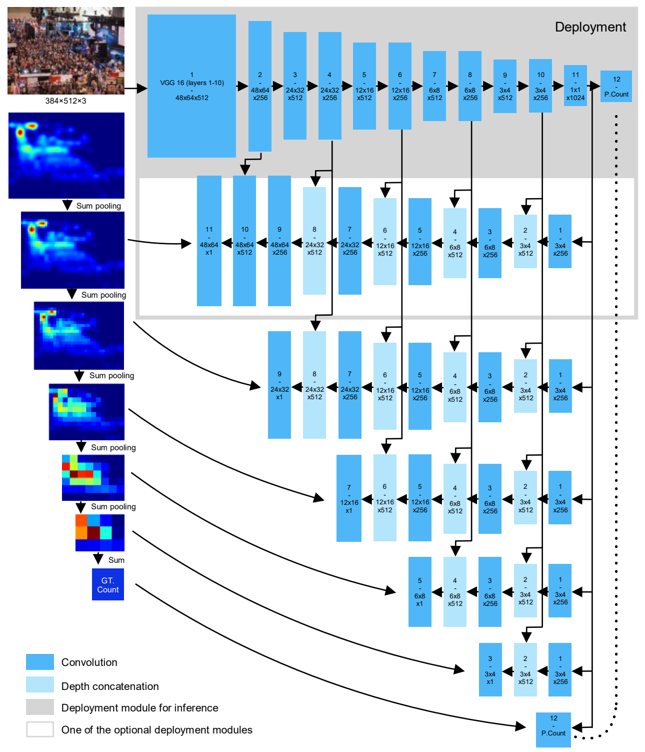
384×512×3
Deployment
VGG 16 (layers 1-10) 48x64x512
Sum pooling
Sum
GT. Count
P.Count
Convolution
Depth concatenation
Deployment module for inference
One of the optional deployment modules

Figure 2. The architecture of the proposed DeepCount Model. Module in the grey box is the backbone regressor, below which are 5 branches predicting density maps. Numbers on the blocks are referred to detailed layer configuration in Table 1.

enough variance and intricacy in complex counting scenarios. Alternatively, with the development of deep learning, features cab be black-boxed and deeply learned to target the goal. Early success of applying deep learning methods on crowd counting would be the end-to-end deep CNN regression model by Wang et al. [16]. Though, deep learning methods quickly narrow onto density map-based methods which have prevailed over the years since, and it was not until recently in [29] that Idrees et al. report experiments on count regression by advanced CNNs: Resnet101 [25] and Densenet201 [26]. Though, their focus is still on density map-based methods.

2.3. Density map-based methods

Rodriguez et al. [21] first suggest the use of density map can improve crowd counting results significantly. It is supported by Zhang et al [18] whose model produces small density map patches as well as the patch count at its last layer. Following this density map approach, Zhang et al. [4] propose a multi-column architecture (MCNN) to also address scale variance of the counting target. Inspired by such, Gao et.al [11] introduce Scale Aggregation Network (SANet) aggregates multi-scale features and fuses

Module	Backbone	Branch 1	Branch 2	Branch 3	Branch 4	Branch 5
Input	Image 384×512×3	1×1024	1×1024	1×1024	1×1024	1×1024
1	VGG 16 Layers 1-10	Conv-tr-S1 3×4×1024×256	Conv-tr-S1 3×4×1024×256	Conv-tr-S1 3×4×1024×256	Conv-tr-S1 3×4×1024×256	Conv-tr-S1 3×4×1024×256
2	Conv-S1 3×3×512×256	Depth Concatenation	Depth Concatenation	Depth Concatenation	Depth Concatenation	Depth Concatenation
3	Conv-S2 3×3×256×512	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-S1 1×1×512×1
4	Conv-S1 3×3×512×256	Depth Concatenation	Depth Concatenation	Depth Concatenation	Depth Concatenation	
5	Conv-S2 3×3×256×512	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-S1 1×1×512×1	
6	Conv-S1 3×3×512×256	Depth Concatenation	Depth Concatenation	Depth Concatenation		
7	Conv-S2 3×3×256×512	Conv-tr-S2 4×4×512×256	Conv-tr-S2 4×4×512×256	Conv-S1 1×1×512×1		
8	Conv-S1 3×3×512×256	Depth Concatenation	Depth Concatenation			
9	Conv-S2 3×3×256×512	Conv-tr-S2 4×4×512×256	Conv-S1 1×1×512×1			
10	Conv-S1 3×3×512×256	Depth Concatenation				
11	Conv-S1 3×4×256×1024	Conv-S1 1×1×512×1				
12	Fc 1024×1					
Output	P. Count	P. Density Map 48×64	P. Density Map 24×32	P. Density Map 12×16	P. Density Map 6×8	P. Density Map 3×4

Table 1. Configuration of DeepCount network. In the table, Conv and Conv-tr mean convolution and transposed convolution respectively. The pattern $H \times W \times C \times C'$ represents the dimension of convolution kernel. S denotes strides. P denotes prediction.

them in every layer. Likewise, Switching-CNN [5] has independent columns of regressors similar to multi-column network with different receptive fields, and ic-CNN [15] aims at predicting high resolution density maps with two branches. Another set of methods devote themselves to trace contextual information as well as other abstractions all in a bit to improve the predicted density maps [6, 35, 36, 37]. On the other hand, CSRNet [16] builds dilated convolution layers upon a VGG-16 [19] backbone straightforward without too many manoeuvres yet reports excellent results and thus becomes more practiced at present.

Compared to these methods, our DeepCount model fuses gradients other than features and avoids relying on the hard-to-produce density maps to make prediction but instead to leverage them on training and separate them to be alternative on inference. By so doing, our method has incorporated advantages of accuracy, flexibility, and efficiency.

3. DeepCount

3.1. Network Configuration

As shown in Figure 2, our proposed model consists of a straightforward down-sampling backbone and five branches interconnected to it. The backbone by itself has relatively low complexity. It functions as a deep CNN regressor which takes the crowd image as input and predict the count by regression.

Specifically, the backbone has a frontend which extracts features off the input image. We transplant the first ten convolution layers from VGG16 as our frontend model for faster training. The frontend produces feature maps of 8 times smaller spatial width and height relative to the input. We design the network to have input size of 384×512 to cater most practical uses, whereas arbitrary larger input image sizes are tackled by division and combination. Following are some 3×3 convolution layers to further dwindle the feature maps size until when the it matches a 3×4 convolution layer entering to convolve the feature maps to produce a 1×1024 vector. We use 3×3 convolution with strides of 2 to halve the spatial dimension of feature maps. In addition, a standard 3×3 convolution layer is put between two down-sampling layers to deepen the network and to smooth the reduction of features.

As for branches, they work in an up-sampling manner. Branches stemming from the last feature layer (1×1024) of the backbone uses 4×4 transposed convolution with strides of 2 to up-sample the feature maps. To the output of each transposed convolution layer, a layer of feature maps from backbone with the same dimension is appended. Together, they form the input to the next transposed convolution layer, until this concatenation has the spatial dimension that matches the target density map of the branch, where finally a 1×1 convolution is used to reconstruct all channels to one to produce prediction density map. At the end of the backbone, another 1×1 convolution (or fully connected layer as the two function the same) regresses a scalar value, the count prediction. We call our network DeepCount for deep counting CNN. Table 1 details our configuration.

3.2. Gradient Fusion

We regard our methodology of designing DeepCount as Gradient Fusion. Multi-column methods such as MCNN [4], CP-CNN [6] are feature fusion methods assembling different columns the features from which are fused and gradients to which are separated. In contrast, we design branches that bring in different gradients and fuse them together to train our critical module to make good prediction. Put figuratively, with all those interconnections between branches and backbone, gradients from multiple sources propagating backwards filtered by branches find their shortcuts to penetrate into the backbone network multilayeredly when supervision is applied. This is a process that enables backbone to be trained to summarise useful information and gain more knowledge about the representation to improve itself.

3.2 Creating Ground Truth Density Maps

To produce ground truth density maps for training and validation, we first apply convolution by fixed Gaussian kernel with standard deviation $\sigma = 5$ (on the contrary of geometry-adaptive kernel adopted in [4, 16]) to generate density map of the same resolution as the original image, before Sum Pooling is used to produce different sizes of density maps. Sum pooling explicitly is summing all values inside a pooling window. In this operation, let input image be 384×512, and then density maps of sizes $\{48 \times 64, 24 \times 32, 12 \times 16, 6 \times 8, 3 \times 4\}$ are produced. An element in a density map matrix can also be interpreted as count of the cell, if its value is great enough to seem sensible. By means of sum pooling, element sum of all density maps pooled from one origin stays unchanged; this

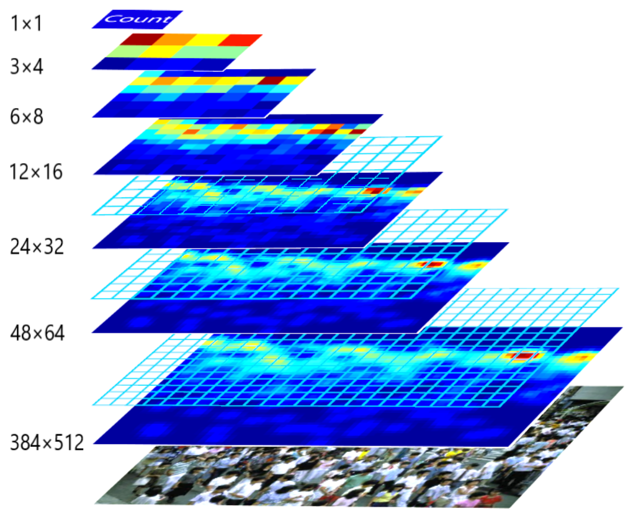

Figure 3. Illustration of sum pooling operation for density map generation.

sum is to be the ground truth count of this image. Figure 3 gives an illustration of this operation.

3.3. Objective Function

Labelling congested crowd data is indeed a painstaking task for human annotators, especially in some highly congested cases where the factual number of people is inevitably untraceable. This results many annotations in congestions themselves being estimations, which means noise in the situation. Hence, L1-norm loss is adopted so as to enhance robustness against noise as well as to convey steady updates to the network. We first define our loss function as:

$$L(\Theta) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{i,j} |y'_{nk} - f(X_n, \Theta_k)|_{ij} \qquad (1)$$

where N is the size of the training batch, K enumerates all outputs from branches and the global count regressor ($K = 6$ & $k \in \{1,2,\dots,6\}$), y'_{nk} is the ground truth density map (or global count), X_n is the input image and Θ_k denotes all parameters in model f that contribute to making the corresponding k_{th} prediction.

Given this loss function as basis, we add a multiplier β to accentuate the importance of the global count prediction on backbone (where $k = 6$). So, we notate it as a function of k:

$$B(k) = \begin{cases} \beta, & k = 6 \\ 1, & k \neq 6 \end{cases} \qquad (2)$$

Moreover, we add another hyperparameter ω to approximately adjust the loss to a reasonably small value (<10). An L2 regularisation term is also added to the loss function in an attempt to reduce overfitting. Hence, the loss function finally becomes:

$$\boldsymbol{L(\Theta)} = \left[\frac{\mathbf{1}}{\mathbf{2N}}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{i,j}|\boldsymbol{y'_{nk}} - \boldsymbol{f(X_n,\Theta_k)}|_{ij}\cdot \boldsymbol{B(k)}\right]\cdot \boldsymbol{\omega} + \frac{\boldsymbol{\lambda}}{\mathbf{2}}\|\boldsymbol{\Theta}\|_2^2 \quad (3)$$

3.4. Implementation

VGG16 model pretrained on ImageNet is used to initialise the backbone frontend. Therefore, input images are normalised in the same manner as how the VGG16 model was trained. As for initialising the remaining part of the model, we use Xavier [33] initialisation for weights and constant value 0 for biases. With the exception of the frontend from VGG16 where ReLU is the activation function, we use parametric ReLUs (leaky ReLU):

$$a(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \quad (4)$$

following every convolution layer. We choose $\lambda = 1 \times 10^{-5}$, $\omega = 1 \times 10^{-2}$ and $\beta = 16$ empirically for the loss function in equation (3) and $\alpha = 0.2$ for the activation parameter in equation (4) in light of experiments reported in [39]. We use Gradient Descent optimisation with momentum 0.9 and initial learning rate 1×10^{-4} to train our model, except for parameters in frontend where learning rate is divided by a factor of 2. Batch size N is set to 32. Model is to be trained for around 100 epochs.

As alluded to above, to cope with data images of various sizes, we divide original images to 384×512 crops to feed into our network. In testing, results from cropped images are to be merged to assemble the original image again. In order to make fair comparison with benchmark results, we do no more data augmentation than random cropping and mirroring during training.

4. Evaluation

In this section, we report evaluation results yielded by our method introduced above. We evaluate our DeepCount network on four public datasets: Shanghai Tech [4] Part A and Part B, UCF-QNRF [29] and Mall [13].

4.1. Evaluation Metrics

For evaluation, we compute mean-absolute error (MAE) and root-mean-squared error (RMSE):

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i - C_i^{GT}| \quad (5)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|C_i - C_i^{GT}\|^2} \quad (6)$$

where N is the number of testing images, C_i and C_i^{GT} meaning predicted count and ground truth count respectively.

4.2. Shanghai Tech

Shanghai Tech dataset includes Part A and Part B. Part A is the dataset for congested crowd counting. It has 241,677 annotations in 300 training images and 182 testing images with an average number 501. Part A in which most images are congested is one of the noisiest ones. On the other hand, the relatively sparse Part B is separated to into training set with 400 images and test set with 316 images taken from streets in Shanghai. Our DeepCount model achieves state-of-the-art performance on both datasets. Test results are shown in Table 2.

	Part A		Part B	
Method	MAE	RMSE	MAE	RMSE
MCNN [4]	110.2	173.2	26.4	41.3
Switching CNN [5]	90.4	135.0	21.6	33.4
DecideNet [23]	-	-	20.75	29.42
CP-CNN [6]	73.6	106.4	20.1	30.1
ic-CNN [15]	68.5	116.2	10.7	16.0
CSRNet [16]	68.2	115.0	10.6	16.0
SANet [11]	67.0	**104.5**	8.4	13.6
DeepCount (ours)	**65.2**	112.5	**7.2**	**11.3**

Table 2. Test results on Shanghai Tech Part A and Part B

4.3. UCF-QNRF

The UCF-QNRF dataset has a greater number of annotations (1,251,642) in higher quality images of a wider variety of scenes including sparse and dense ones. There are extremely dense scenes in this dataset, so much so that there is single image with maximumly 12,865 of annotations. It is the most challenging one in terms of crowd density. Our method, to a great extent, outperforms current methods (see Table 3).

Method	MAE	RMSE
Idrees et al. (2013) [3]	315	508
MCNN [4]	277	426
CMTL [24]	252	514
Switching CNN [5]	228	445
Resnet101[25]	190	277
Densenet201[26]	163	226
Idrees et al. (2018) [29]	132	191
DeepCount (ours)	**95.7**	**167.1**

Table 3. Test results on UCF-QNRF.

CSRNet (backend)			DeepCount Branch 1			DeepCount (backend)		
Layer	Output	Million FLOPs	Layer	Output	Million FLOPs	Layer	Output	Million FLOPs
Conv-s1 3×3×512×512	48×64×512	7248	Conv-tr-s1 3×4×1024×256	3×4×256	37	Conv-s1 3×3×512×256	48×64×256	3624
Conv-s1 3×3×512×512	48×64×512	7248	Conv-tr-s2 4×4×512×256	6×8×256	101	Conv-s2 3×3×256×512	24×32×512	906
Conv-s1 3×3×512×512	48×64×512	7248	Conv-tr-s2 4×4×512×256	12×16×256	403	Conv-s1 3×3×512×256	24×32×256	906
Conv-s1 3×3×512×256	48×64×256	3624	Conv-tr-s2 4×4×512×256	24×32×256	1611	Conv-s2 3×3×256×512	12×16×512	226
Conv-s1 3×3×256×128	48×64×128	906	Conv-tr-s2 4×4×512×256	48×64×256	6442	Conv-s1 3×3×512×256	12×16×256	226
Conv-s1 3×3×128×64	48×64×64	226	Conv-s1-p0 1×1×512×1	48×64×1	2	Conv-s2 3×3×256×512	6×8×512	57
Conv-s1 1×1×64×1	48×64×1	0.2				Conv-s1 3×3×512×256	6×8×256	57
						Conv-s2 3×3×256×512	3×4×512	14
						Conv-s1 3×3×512×256	3×4×256	14
						Conv-s1-p0 3×4×256×1024	1×1×1024	3
						Conv-s1-p0 (Fc) 1×1×1024×1	1	0.001
Total		26500			**8596**			**6034**

Table 5. Comparing between backends from CSRNet and backbone of our DeepCount model.

4.4. Mall

Unlike the three datasets aforementioned. Images from Mall dataset are surveillance frames from a static viewpoint at a same venue. Crowd in this dataset is sparse. So, Mall is less challenging than others. Although previous methods have shown very promising results on this dataset, we still evaluate our model on it to demonstrate its performance on invariant scene and as well to make comparison with some detection-based methods. (see Table 4).

Method	MAE	RMSE
R-FCN [27]	6.02	5.46
Faster R-CNN [28]	5.91	6.60
Count Forest [30]	4.40	2.40
MoCNN [31]	2.75	13.40
Weighted VLAD [32]	2.41	9.12
DecideNet [23]	**1.52**	**1.90**
DeepCount (ours)	1.66	2.13

Table 4. Test results on Mall

5. Discussion

5.1. Capacity and Velocity

Arguably, the more parameters a neural network has, the greater its potential is to have higher capacity to model the target function of underlying relationship for the random variable. Although many cases suggest otherwise, we do see the positive correlation between extra parameters and increments of performance [4, 19, 25, 38]. Be that as it may, we still tend to avoid expensive computation a larger network would bear in practice. Trading off between capacity and velocity has been a dilemma for long. Hereby, we explicate how the idea of our DeepCount network is able to pursue both capacity and velocity at the same time. CSRNet [16], in whose paper Li et al. did persuasively argue about the effect of number of parameters and design efficiency, is the counterpart of our model for comparative demonstration.

CSRNet and our backbone network both have a straightforward design and use the same VGG16 frontend, so the difference between the two lies in the backends, where CSRNet predicts the density map and our model predicts the count. Assuming they receive the same 384×512×3 input, we detail their layer configuration with corresponding output and computation cost of each layer in Table 5. In addition, we add branch 1 which predicts the density map as same as the one predicted by CSRNet to the table. We compute computation cost in terms of floating-point operations (FLOPs) happen throughout a forward pass in the backends. FLOPs of one convolution layer are computed as:

$$FLOPs = H \cdot W \cdot C \cdot K_1 \cdot K_2 \cdot C' \qquad (7)$$

where it depends multiplicatively upon output feature map size $H \times W$, convolution kernel size $K_1 \times K_2$, the number of output channels C, and the number of input channels C'.

We also measure number of parameters as well as frame-per-second (FPS) for both networks (see Table 6). Run time

Method	SHT Part B MAE	Million Parameters	FPS	Speedup
CSRNet	10.6	16.3	33	1×
DeepCount	7.2	21.4 (inference)	45	1.4×

Table 6. Comparing number of parameters and inference speed between CSRNet and the backbone of our DeepCount model.

	Branch 1	Branch 2	Branch 3	Branch 4	Branch 5	Backbone
Output	48×64	24×32	12×16	6×8	3×4	1×1
Shanghai Tech Part A	79.2	73.4	69.7	66.7	65.8	65.2
Shanghai Tech Part B	9.7	8.9	8.0	7.4	7.2	7.2
UCF-QNRF	193.3	185.0	155.3	112.3	96.9	95.7
Million FLOPs	6034	2152	541	138	38	-

Table 7. Comparing between outputs on different branches.

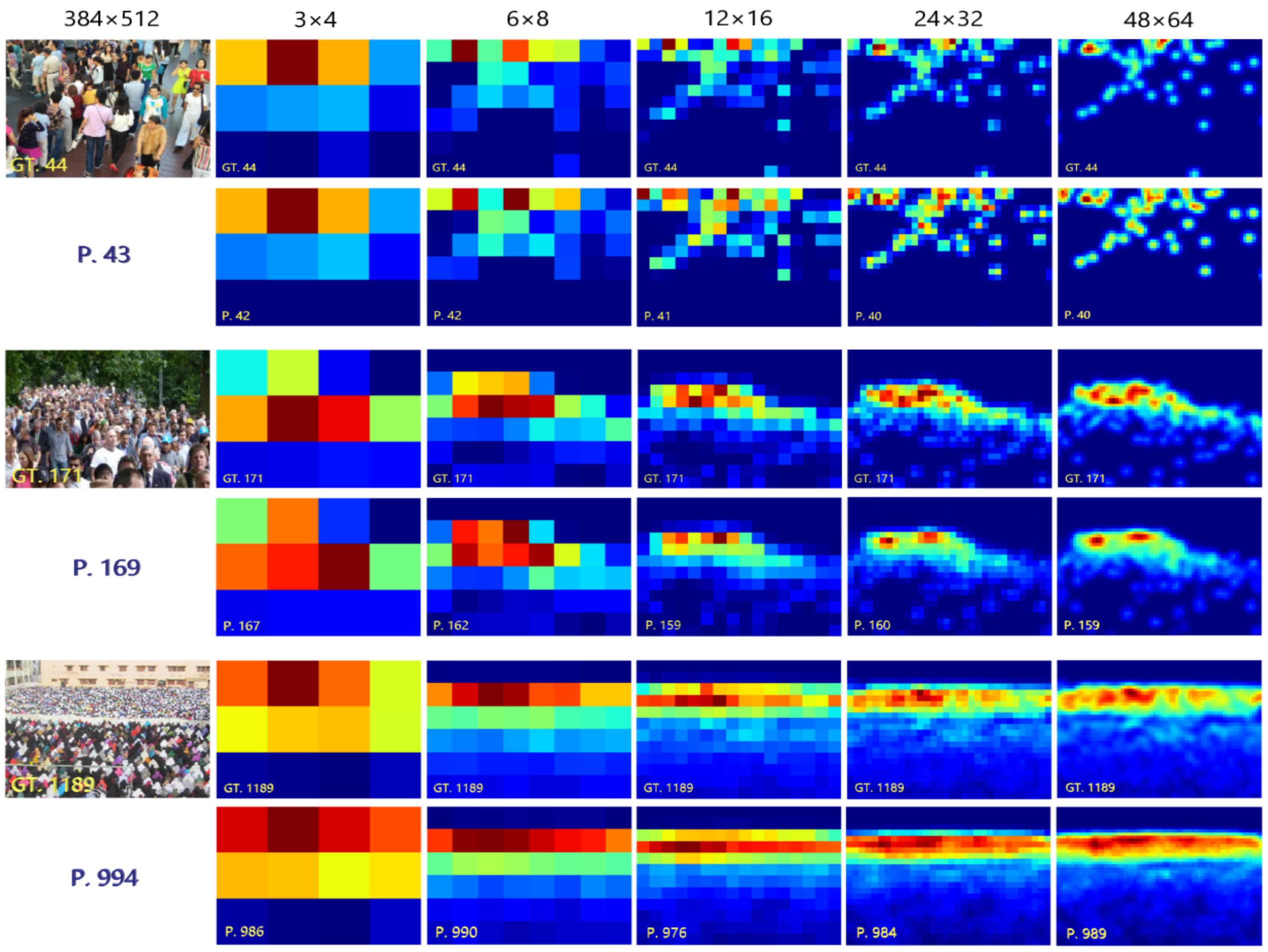

Figure 4. Examples of ground truth and output density maps of test data from Shanghai Tech Part B (top), Part A (middle), UCF-QNRF (bottom). Density maps in the row starts with an image are the ground truths. The ones below are the predictions. The number in blue below the image is the predicted count by backbone alone.

evaluation was performed on one NVIDIA Tesla P40 GPU. As mentioned above, the backbone of our DeepCount model can be a standalone network detached from the rest in deployment, and thereby becomes a count regressor without computing the computationally expensive density maps, and noticeably, with overwhelming performance compared to other global count regression approaches.

As shown in Table 5 and Table 6, having a deeper architecture and greater preponderance of parameters though, our DeepCount backbone does count inferences with much less FLOPs and in higher velocity, and perhaps more importantly, with higher accuracy. These quantitative results indicate our proposed DeepCount model has the ability of accommodating more variations while making faster and better prediction which implies its potential of outstanding capacity and efficiency.

5.2. Comparison on Branches

Each branch produces a density map at its end. Those density maps are of different resolutions. As we use the count regressed by backbone, we can as well integrate output density maps from different branches to make prediction like other density map-based methods. In the following, we compare predictions made in terms of MAE between branches. Besides, we compute FLOPs for each branch to analyse their computation cost. Results are shown in Table 7.

As shown, the fact that branches that are able to produce larger density maps give inferior predictions on the count compared to those are not implies density maps are harder to optimise for the algorithm. Immediate reasons for this may be that larger density maps are sparser and usually awash with noise which comes from annotations of large-scale heads. Our method avoids predicting the count relying merely on density maps, but exploits useful information from them to rather optimise the global count regressor. This allows more accurate results to be achieved.

In spite of extra computation, there has situations in which density maps, which give information about the distribution, become a requirement. Our DeepCount model can make direct count inference with backbone in its full speed while optionally producing density maps of optional resolutions. User can choose smaller density maps to reduce computational expensiveness or larger ones to give more illuminating impression about the crowd distribution. Figure 4 shows our predicted density maps compared to their ground truths.

5.3. Importance of Gradients

Gradients are considered crucial to the achievement of our model, as we argued. Hence, we detail more experiments to further cast light on the importance of them. Since ReLU has derivatives:

$$a(x)'_{ReLU} = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \quad (8)$$

, where gradients in half of the activation space are set to zero, it could hinder propagation of gradients and cause a large part of the network underused. Instead, Parametric ReLU has derivatives:

$$a(x)'_{PReLU} = \begin{cases} 1, & x > 0 \\ \alpha, & x < 0 \end{cases} \quad (8)$$

with non-zero gradients in all quadrants offering a more generous gradient flow. As shown in Table 8, when gradients are sparse, the capability of the network drops.

Activation	SHT Part B MAE
ReLU	8.8
PReLU	7.2

Table 8. Comparing results between using ReLU (sparse gradients) and PReLU (full gradients).

Also, the idea of being density-aware is to leverage gradients sourced from multi-resolution density maps. In ablation experiments (see Table 9 for the results), as we detach branches one by one from the largest to the smallest, the trend of performance degradation becomes more and more apparent. Interestingly though, the worst case can still be ranked as one of the-state-of-the-arts.

Ablation	SHT Part B
No ablation	7.2
Branch 1 detached	7.4
Branches 1-2 detached	7.5
Branches 1-3 detached	7.7
Branches 1-4 detached	8.3
Branches 1-5 detached	9.1

Table 9. Ablation on branches.

These two facts suggest the abundance of gradients has advantageous influence on our network. Nevertheless, we are now safer to conclude that parameters in branches are indeed instrumental in training of the backbone. Giant as it may be, the network of branches is not a concern in deployment for inferences. Unless training efficiency is in a serious consideration, having a rationally greater number of parameters in this auxiliary module should be deemed innocuous as long as performance does not remain stagnant.

6. Conclusion

In this paper, we discussed advantages and limitations of current crowd counting methods, in light of which we proposed a novel DeepCount network to be both accurate on count prediction and flexible on density map generation. State-of-the-art performance on public datasets evidenced the effectiveness of our method. We reckon it is worth mentioning that our method also far prevailed over current state-of-the-art on a 14k-images dataset from BaiduStar2018 competition of crowd counting which could be a more convincing evidence than the abovementioned. Our code and models will be publicly available at https://github.com/********.

References

[1] Ge W, Collins R T. Marked point processes for crowd counting[C]//2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009: 2913-2920.

[2] Idrees H, Soomro K, Shah M. Detecting humans in dense crowds using locally-consistent scale prior and global occlusion reasoning[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(10): 1986-1998.

[3] Idrees H, Saleemi I, Seibert C, et al. Multi-source multi-scale counting in extremely dense crowd images[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 2547-2554.

[4] Zhang Y, Zhou D, Chen S, et al. Single-image crowd counting via multi-column convolutional neural network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 589-597.

[5] Sam D B, Surya S, Babu R V. Switching convolutional neural network for crowd counting[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017: 4031-4039.

[6] Sindagi V A, Patel V M. Generating high-quality crowd density maps using contextual pyramid cnns[C]//Proceedings

of the IEEE International Conference on Computer Vision. 2017: 1861-1870.
[7] Zhang C, Li H, Wang X, et al. Cross-scene crowd counting via deep convolutional neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 833-841.
[8] Sindagi V A, Patel V M. A survey of recent advances in cnn-based single image crowd counting and density estimation[J]. Pattern Recognition Letters, 2018, 107: 3-16.
[9] Zitouni M S, Bhaskar H, Dias J, et al. Advances and trends in visual crowd analysis: A systematic survey and evaluation of crowd modelling techniques[J]. Neurocomputing, 2016, 186: 139-159.
[10] Viola P, Jones M J, Snow D. Detecting pedestrians using patterns of motion and appearance[J]. International Journal of Computer Vision, 2005, 63(2): 153-161.
[11] Cao X, Wang Z, Zhao Y, et al. Scale aggregation network for accurate and efficient crowd counting[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 734-750.
[12] Gall J, Yao A, Razavi N, et al. Hough forests for object detection, tracking, and action recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2011, 33(11): 2188-2202.
[13] Chen K, Loy C C, Gong S, et al. Feature mining for localised crowd counting[C]//BMVC. 2012, 1(2): 3.
[14] Chan A B, Liang Z S J, Vasconcelos N. Privacy preserving crowd monitoring: Counting people without people models or tracking[C]//2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008: 1-7.
[15] Ranjan V, Le H, Hoai M. Iterative crowd counting[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 270-285.
[16] Li Y, Zhang X, Chen D. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 1091-1100.
[17] Ma Z, Chan A B. Crossing the line: Crowd counting by integer programming with local features[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013: 2539-2546.
[18] Zhang C, Li H, Wang X, et al. Cross-scene crowd counting via deep convolutional neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 833-841.
[19] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[20] Kong D, Gray D, Tao H. A viewpoint invariant approach for crowd counting[C]//18th International Conference on Pattern Recognition (ICPR'06). IEEE, 2006, 3: 1187-1190.
[21] Rodriguez M, Laptev I, Sivic J, et al. Density-aware person detection and tracking in crowds[C]//2011 International Conference on Computer Vision. IEEE, 2011: 2423-2430.
[22] Li M, Zhang Z, Huang K, et al. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection[C]//2008 19th International Conference on Pattern Recognition. IEEE, 2008: 1-4.
[23] Liu J, Gao C, Meng D, et al. Decidenet: Counting varying density crowds through attention guided detection and density estimation[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 5197-5206.
[24] Sindagi V A, Patel V M. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting[C]//2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2017: 1-6.
[25] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
[26] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.
[27] Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[C]//Advances in neural information processing systems. 2016: 379-387.
[28] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.
[29] Idrees H, Tayyab M, Athrey K, et al. Composition loss for counting, density map estimation and localization in dense crowds[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 532-546.
[30] Pham V Q, Kozakaya T, Yamaguchi O, et al. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 3253-3261.
[31] Kumagai S, Hotta K, Kurita T. Mixture of counting cnns: Adaptive integration of cnns specialized to specific appearance for crowd counting[J]. arXiv preprint arXiv:1703.09393, 2017.
[32] Sheng B, Shen C, Lin G, et al. Crowd counting via weighted VLAD on a dense attribute feature map[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2018, 28(8): 1788-1797.
[33] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks[C]//Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010: 249-256.
[34] Zhao T, Nevatia R, Wu B. Segmentation and tracking of multiple humans in crowded environments[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 30(7): 1198-1211.
[35] Liu, L., Wang, H., Li, G., Ouyang, W. and Lin, L., 2018. Crowd counting using deep recurrent spatial-aware network. arXiv preprint arXiv:1807.00601.
[36] Sam, D.B. and Babu, R.V., 2018, April. Top-down feedback for crowd counting convolutional neural network. In Thirty-Second AAAI Conference on Artificial Intelligence.
[37] Shi, Z., Zhang, L., Liu, Y., Cao, X., Ye, Y., Cheng, M.M. and Zheng, G., 2018. Crowd counting with deep negative correlation learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5382-5390).

[38] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[39] Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.