This ECCV 2018 paper, provided here by the Computer Vision Foundation, is the author-created version.
The content of this paper is identical to the content of the officially published ECCV 2018
LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/eccv

Iterative Crowd Counting

Viresh Ranjan, Hieu Le, and Minh Hoai

Department of Computer Science,
Stony Brook University
{vranjan,hle,minhhoai}@cs.stonybrook.edu

Abstract. In this work, we tackle the problem of crowd counting in images. We present a Convolutional Neural Network (CNN) based density estimation approach to solve this problem. Predicting a high resolution density map in one go is a challenging task. Hence, we present a two branch CNN architecture for generating high resolution density maps, where the first branch generates a low resolution density map, and the second branch incorporates the low resolution prediction and feature maps from the first branch to generate a high resolution density map. We also propose a multi-stage extension of our approach where each stage in the pipeline utilizes the predictions from all the previous stages. Empirical comparison with the previous state-of-the-art crowd counting methods shows that our method achieves the lowest mean absolute error on three challenging crowd counting benchmarks: Shanghaitech, WorldExpo'10, and UCF datasets.

Keywords: crowd counting, density estimation, multi-stage CNN

1 Introduction

Gathering of large crowds is commonplace nowadays, and estimating the size of a crowd is an important problem for different purposes ranging from journalism to public safety. Without turnstiles to provide a precise count, the media and crowd safety specialists must estimate the size of the crowd based on images and videos of the crowd. Manual visual estimation, however, is difficult and laborious for humans. Humans are good at subitizing, i.e., predicting fast and accurate counts for small number of items, but the accuracy with which humans count deteriorates as the number of items increase [7]. Furthermore, the addition of each new item beyond a few adds an extra processing time of around 250 to 300 milliseconds [17]. As a result, any crowd monitoring system that relies on humans for counting people in crowded scenes will be slow and unreliable. There is a need for an automatic computer vision algorithm that can accurately count the number of people in crowded scenes based on images and videos of the crowds.

There exist a number of computer vision algorithms for crowd counting, and the current state-of-the-art methods are based on *density estimation* rather than *detection-then-counting*. Density-estimation methods use Convolutional Neural

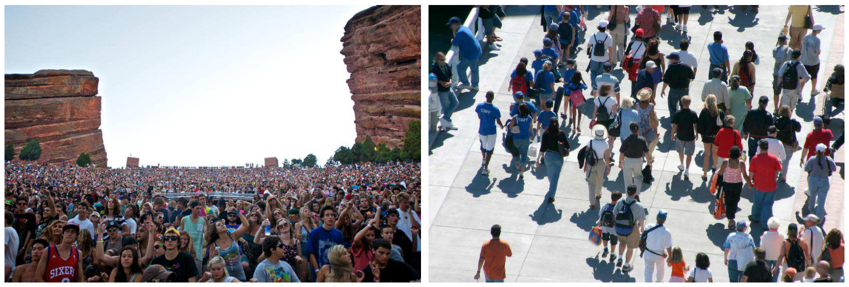

Fig. 1. Crowd counting can be posed as a CNN-based density estimation problem, but this problem can be challenging for a single CNN due to the huge variation of density values across pixels of different images. This figure shows two images from the Shanghaitech dataset that have very different crowd densities. As can be seen, crowd count could vary from a few to a few thousand.

Networks (CNNs) [9, 8] to output a map of density values, one for each pixel of the input image. The final count estimate can be obtained by summing over the predicted density map. Unlike the detection-then-counting approach (e.g., [5]), the output of the density estimation approach at each pixel is not necessarily binary. Density estimation has been proved to be more robust than the detection-then-counting approach because the former does not have to commit to binarized decisions at an early stage.

Estimating the crowd density per pixel is a challenging task due to the large variation of the crowd density values. As shown in Figure 1, some images contain hundreds of people, while others have only a few. It is difficult for a single CNN to handle the entire spectrum of crowd densities. Earlier works [20, 15] have tackled this challenge by using a multi-column or a switching CNN architecture. These CNN architectures consist of three parallel CNN branches with different receptive field sizes. In such architectures, a branch with smaller receptive fields could handle the high density images well, while a branch with larger receptive fields could handle the low density images. More recently, a five-branch CNN architecture was proposed [16] where three of the branches resembled the previous multi-column CNN [20], while the remaining two branches acted as global and local context estimators. These context estimator branches were trained beforehand on the related task of classifying the image into different density categories. Some of the key takeaways from these previous approaches are: (1) using a multi-column CNN model with varying kernel sizes improves the performance of crowd density estimation; and (2) augmenting the feature set with the ones learned from a task related to density estimation, such as count range classification, improves the performance of the density estimation task.

In this work, we propose iterative counting Convolutional Neural Networks (ic-CNN), a CNN-based iterative approach for crowd counting. Unlike previous approaches, where three [20, 15] or more [16] columns are needed to achieve good performance, our ic-CNN approach has a simpler architecture comprising of two

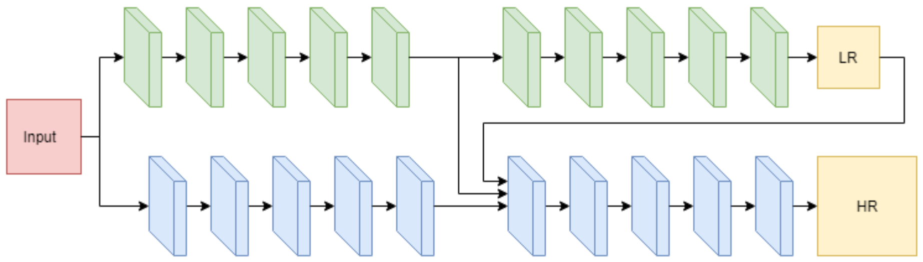

Fig. 2. Figure shows the ic-CNN architecture which consists of two columns/branches. On the top is the Low Resolution CNN branch (LR-CNN) and at the bottom is the High Resolution CNN branch (HR-CNN). LR-CNN predicts a density map at a lower resolution (LR). It passes the predicted density map and the convolutional feature maps to HR-CNN. HR-CNN fuses its feature maps with the feature maps and predicted density map from LR-CNN, and predicts a high resolution density map (HR) at the size of the original image. LR and HR are low and high resolution prediction maps respectively.

columns/branches. The first branch predicts a density map at a lower resolution of $\frac{1}{4}$ the size of the original image, and passes the predicted map and a set of convolutional features to the second branch. The second branch predicts a high resolution density map at the size of the original image. Density maps contain information about the spatial distribution of crowd in an image. Hence, the first stage map serves as an important feature for the high resolution density map prediction task. We also propose a multi-stage extension of ic-CNN where we combine multiple ic-CNNs sequentially to further improve the quality of the predicted density map. Each ic-CNN in the multi-stage pipeline provides both the low and high resolution density predictions to all subsequent stages. Figure 2 illustrates the schematic architecture for ic-CNN. ic-CNN has two branches: Low Resolution CNN (LR-CNN) and High Resolution CNN (HR-CNN). LR-CNN predicts the density map at a low resolution while HR-CNN predicts the density map at the original image resolution. The key highlights of our work are:

1. We propose ic-CNN, a two-stage CNN framework for crowd density estimation and counting.
2. ic-CNN achieves state of the art results on multiple crowd counting datasets. On Shanghaitech Part B dataset, ic-CNN yields 48.3% improvement in terms of mean absolute error over the previously published results [16].
3. We also propose a multi-stage extension of ic-CNN, which can combine predictions from multiple ic-CNN models.

2 Related Work

Crowd counting is an important research problem and a number of approaches have been proposed by the computer vision community. Earlier work tackled crowd counting as an object detection problem [11, 12]. Lin *et al.* [12] extracted Haar features for head like contours and used an SVM classifier to classify these features as the contour of a head or not. Li *et al.* [11] proposed a detection based approach where the input image was first segmented into foreground-background regions and a HOG feature based head-shoulder detector was used to detect each person in the crowd. These detection based methods often fail to accurately count people in extremely dense scenes. To handle images of dense crowds, some methods [2, 3] proposed to use a regression approach to avoid the harder detection problem. They instead extracted local patch level features and learned a regression function to directly estimate the total count for an input image patch. These regression approaches, however, do not fully utilize the available annotation associated with training data; they ignore the spatial density and distribution of people in training images. Several researchers [10, 14] proposed to use a density estimation approach to take advantage of the provided crowd density annotation maps of training images. Lempitsky & Zisserman [10] learned a linear mapping between the crowd images and the corresponding ground truth density maps. Pham *et al.* [14] learned a more robust mapping by using a random decision forest to estimate the crowd density map. These density-based methods solve some of the challenges faced by the earlier detection and regression based approaches, by avoiding the harder detection problem and also utilizing the spatial annotation and correlation. All aforementioned methods predated the deep-learning era, and they used hand crafted features for crowd counting.

More recent methods [18, 4, 20, 15, 13, 16] used CNNs to tackle crowd counting. Wang *et al.* [18] posed crowd counting as a regression problem, and used a CNN model to map the input crowd image to its corresponding count. Instead of predicting the overall count, Fu *et al.* [4] classified an image into five broad crowd density categories and used a cascade of two CNNs in a boosting like strategy where the second CNN was trained on the images misclassified by the first CNN. These methods also overlooked the benefits provided by the crowd density annotation maps.

The methods that are most related to our work are [20, 15, 16]. Zhang *et al.* [20] proposed a CNN-based method to predict crowd density maps. To handle the large variation in crowd densities and sizes across different images, Zhang *et al.* [20] proposed a multi-column CNN architecture (MCNN) with filters and receptive fields of various sizes. The CNN column with smaller receptive field and filter sizes were responsible for the denser crowd images, while the CNN columns with larger receptive fields and filter sizes were meant for the less dense crowd images. The features from the three columns were concatenated and processed by a $1{\times}1$ convolution layer to predict the final density map. To handle the variations in density and size within an image, the authors divided each image into non-overlapping patches, and trained the MCNN architecture on these patches. Given that the number of training samples in annotated crowd counting datasets is much smaller in comparison to the datasets pertaining to image classification and segmentation tasks, training a CNN from scratch on full images might lead to overfitting. Hence, patch-based training of MCNN was essential in preventing overfitting and also improving the overall performance by serving as a data augmentation strategy. One issue with MCNN was that it fused the features form three CNN columns for predicting the density map. For a given patch, it is expected that the counting performance can be made more accurate by choosing the right CNN column that specializes in analyzing images of similar density values. Sam *et al.* [15] built on this idea and decoupled the three columns into separate CNNs, each focused on a subset of the training patches. To decide which CNN to assign a patch to, the authors trained a CNN-based switch classifier. However, since the ground truth label needed to train the switch classifier was unavailable, the authors resorted to a multi-stage training strategy: 1) training the three density predicting CNNs on the entire set of training patches, 2) training the switch classifier using the count from the previous stage to decide the switch labels, and 3) retraining the three CNNs using the patches assigned by the switch classifier. In a more recent work, Sindagi *et al.* [16] further modified the MCNN architecture by adding two more branches for estimating global and local context maps. The global/local context prediction branches were trained beforehand for the related task of classifying an image/patch into five different count categories. The classification scores were used to create a feature map of the same size as the image/patch, which served as the global/local context map. These context maps were fused with the convolutional feature maps obtained using a three branch multi-column CNN, and the resulting features were further processed by convolutional layers and a 1×1 convolution layer to obtain the final density map.

3 Proposed Approach

In this section, we describe the architecture of ic-CNN, its multi-stage extension and the training strategy. ic-CNN is discussed in Section 3.1. The multi stage extension of ic-CNN is discussed in Section 3.2, and the training details are discussed in Sec 3.3.

3.1 Iterative Counting CNN

Let $\mathcal{D} = \{(X_1, Y_1, Z_1), \ldots, (X_n, Y_n, Z_n)\}$ be the training set of n (image, high resolution density map, low resolution density map) triplets, where X_i is the i^{th} image, Y_i is the corresponding crowd density map at the same resolution as the image X_i, and Z_i is a low resolution version of the crowd density map. Y_i and Z_i have the same overall count. Let f_l and f_h be the mapping functions which transform the image into the low resolution and high resolution density maps, respectively. Let the parameters of the low resolution branch (LR-CNN) and high resolution branch (HR-CNN) be θ_l and θ_h respectively. Note that f_l depends on only θ_l, while f_h depends on both θ_l and θ_h. Given an input image X_i, the low resolution density map $\hat{Z}_i$ can be obtained by a doing a forward pass through the LR-CNN branch:

$$\hat{Z}_i = f_l(X_i; \theta_l). \tag{1}$$

The inputs to the high resolution branch HR-CNN are: the image X_i, the features computed by the low resolution branch LR-CNN, and the low resolution prediction $\hat{Z}_i$. HR-CNN predicts a high resolution density map of the same size as the original image:

$$\hat{Y}_i = f_h(X_i, \hat{Z}_i; \theta_l, \theta_h). \tag{2}$$

The low resolution prediction $\hat{Z}_i$ contains information about the spatial distribution of the crowd in the image X_i. It serves as an important feature map for the high resolution prediction task. We can learn the parameters θ_l and θ_h by minimizing the loss function $\mathcal{L}(\theta_l, \theta_h)$:

$$\mathcal{L}(\theta_l, \theta_h) = \frac{1}{n}\sum_{i=1}^{n}(\lambda_l L(f_l(X_i; \theta_l), Z_i) + \lambda_h L(f_h(X_i, \hat{Z}_i; \theta_l, \theta_h), Y_i)), \tag{3}$$

where $L(\cdot, \cdot)$ denotes the loss function, and a reasonable choice is to use the squared error between the estimated and ground truth values. λ_l and λ_h are scalar hyperparameters which can be used to give more importance to one of the loss terms. Using Equations (1) and (2), the right hand side can be further simplified as:

$$\mathcal{L}(\theta_l, \theta_h) = \frac{1}{n}\sum_{i=1}^{n}(\lambda_l L(\hat{Z}_i, Z_i) + \lambda_h L(\hat{Y}_i, Y_i)). \tag{4}$$

At test time, given an image X_i, we first obtain the low resolution output $\hat{Z}_i$ by doing a forward pass through LR-CNN and then pass the convolutional features and the low resolution map $\hat{Z}_i$ to HR-CNN, which will predict the high resolution map $\hat{Y}_i$. We use the high resolution output predicted by HR-CNN as the final output of ic-CNN. The overall crowd count is obtained by summing over all the pixels in the density map $\hat{Y}_i$.
Below we provide the architecture details for the LR-CNN and HR-CNN branches.
LR-CNN. The LR-CNN branch takes as input an image, and predicts a density map at $\frac{1}{4}$ the size of the original image. LR-CNN has the following architecture: Conv3-64, Conv3-64, MaxPool, Conv3-128, Conv3-128, MaxPool, Conv3-256, Conv3-256, Conv3-256, Conv7-196, Conv5-96, Conv3-32, Conv1-1. Here, ConvX-Y implies a convolution layer having Y filters with $X \times X$ kernel size. MaxPool is the max pooling layer. We use a ReLU nonlinearity after each convolutional layer.

HR-CNN. The HR-CNN branch predicts the high resolution density map at the same size as the input image. HR-CNN has the following architecture: Conv7-16, MaxPool, Conv5-24, MaxPool, Conv3-48, Conv3-48, Conv3-24, Conv7-196, Conv5-96, Upsampling-2, Conv3-32, Upsampling-2, Conv1-1. Here, Upsampling-2 is a bilinear interpolation layer which upsamples the input to twice its size.

3.2 Multi-stage Crowd Counting

A multi-stage ic-CNN is a network that combines multiple building blocks of ic-CNN described in the previous section. Each ic-CNN block inputs the low and high resolution prediction maps from all the previous blocks. Given an input image X_i, the low resolution branch of the k^{th} block, represented by the function f_l^k, outputs the low resolution prediction:

$$\hat{Z}_i^k = f_l^k(X_i, \hat{Z}_i^{1:k-1}, \hat{Y}_i^{1:k-1}, \theta_l^k), \tag{5}$$

where θ_l^k represents the parameters of LR-CNN, $\hat{Z}_i^{1:k-1}$ and $\hat{Y}_i^{1:k-1}$ represent the set of low and high level predictions from the first $k-1$ blocks for the input X_i. The high resolution branch of the k^{th} block, represented by the function f_h^k, takes as input the image X_i, the feature maps computed by the low resolution branch f_l^k, the low resolution prediction $\hat{Z}_i^k$, and the entire set of low and high resolution prediction maps from the first $k-1$ blocks. Hence, the output of the k^{th} HR-CNN can be computed using:

$$\hat{Y}_i^k = f_h^k(X_i, \hat{Z}_i^{1:k}, \hat{Y}_i^{1:k-1}, \theta_l^k, \theta_h^k). \tag{6}$$

Note that f_l^k and f_h^k do not depend on the parameters for the first $k-1$ blocks, and $\hat{Z}_i^{1:k-1}$ and $\hat{Y}_i^{1:k-1}$ are treated as fixed inputs (i.e., the parameters of the corresponding network blocks are frozen). We can learn the parameters θ_l^k and θ_h^k by minimizing the loss function $\mathcal{L}(\theta_l^k, \theta_h^k)$:

$$\begin{aligned}\mathcal{L}(\theta_l^k, \theta_h^k) = & \frac{\lambda_l}{n}\sum_{i=1}^{n} L(f_l^k(X_i, \hat{Z}_i^{1:k-1}, \hat{Y}_i^{1:k-1}, \theta_l^k), Z_i) \\ & + \frac{\lambda_h}{n}\sum_{i=1}^{n} L(f_h^k(X_i, \hat{Z}_i^{1:k}, \hat{Y}_i^{1:k-1}, \theta_l^k, \theta_h^k), Y_i).\end{aligned} \tag{7}$$

3.3 Training Details

An ic-CNN is trained by minimizing the loss function $\mathcal{L}(\theta_l, \theta_h)$ from Equation (3). We use the Stochastic Gradient Descent algorithm with the following hyper parameters (unless specified otherwise): learning rate 10^{-4}, momentum 0.9, batch size 1. We give more importance to the high resolution loss term in Equation (3) and set λ_l and λ_h to 10^{-2} and 10^2, respectively.

We train a multi-stage ic-CNN in multiple stages. In the k^{th} stage, we train the k^{th} ic-CNN block by minimizing the loss function given in Equation (7), using the Stochastic Gradient Descent algorithm with the same hyper parameters as above. Once the training for the k^{th} stage has converged, we freeze the parameters for the k^{th} stage and proceed to the next stage.

The training data consists of crowd images and corresponding ground truth annotation files. A ground truth annotation for an image specifies the location of each person in the image with a single dot on the person. We convert this annotation into a binary map consisting of 0's at all locations, except for the annotated points which are assigned the value of 1. We convolve this binary map with a Gaussian filter of standard deviation 5. We use the resulting density map for training the networks.

Table 1. Count errors of different methods on the Shanghaitech dataset. This dataset has two parts: A and B. We compare ic-CNN with the previous state-of-the-art approaches, using two metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). ic-CNN (one stage) is the single stage ic-CNN with two branches HR-CNN and LR-CNN. ic-CNN (two stages) is the two-stage variant of ic-CNN. Both ic-CNN networks outperform the previous approaches in 3 out of 4 cases. On the Shanghaitech Part B dataset, using the one-stage ic-CNN, which has a simpler architecture than CP-CNN [16], we improve on the previously reported state of the art results by 48.3% using the MAE metric and 46.8% using the RMSE metric

	Part A		Part B	
	MAE	RMSE	MAE	RMSE
Crowd CNN [19]	181.8	277.7	32.0	49.8
MCNN [20]	110.2	173.2	26.4	41.3
Switching CNN [15]	90.4	135.0	21.6	33.4
CP-CNN [16]	73.6	**106.4**	20.1	30.1
ic-CNN (one stage)	69.8	117.3	**10.4**	16.7
ic-CNN (two stages)	**68.5**	116.2	**10.7**	**16.0**

4 Experiments

We conduct experiments on three challenging datasets: Shanghaitech [20], WorldExpo'10 [19], and UCF Crowd Counting Dataset [6].

4.1 Evaluation Metrics

Following previous works for crowd counting, we use the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the performance of our proposed method. If the predicted count for image i is $\hat{C}_i$ and the ground truth count is C_i, the MAE and RMSE can be computed as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|C_i - \hat{C}_i|, \quad RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(C_i - \hat{C}_i)^2} \tag{8}$$

where n is the number of test images.

4.2 Experiments on the Shanghaitech Dataset

The Shanghaitech dataset [20] consists of 1198 annotated crowd images. The dataset is divided into two parts, Part-A containing 482 images and Part-B containing 716 images. Part-A is split into train and test subsets consisting of 300 and 182 images, respectively. Part-B is split into train and test subsets consisting of 400 and 316 images. Each person in a crowd image is annotated with one point close to the center of the head. In total, the dataset consists of 330,165 annotated

Table 2. MAE and RMSE on Shanghaitech Part-A dataset as we vary the resolution being used for the low resolution branch LR-CNN of ic-CNN. The resolution of HR-CNN is fixed at 1, the size of the input image.

LR-Resolution	HR-Resolution	MAE	RMSE
1/8	1	74.9	131.6
1/4	1	**69.8**	**117.3**
1/2	1	73.3	124.4
1	1	74.4	128.3

Table 3. Effect of varying hyper parameter λ_h: Mean absolute error on Shanghaitech Part A dataset. λ_l is kept fixed at 10^{-2}.

λ_h	LR-CNN	HR-CNN
10^{-4}	73.7	78.8
10^{-2}	73.0	73.6
1	75.1	73.3
10^{2}	79.9	69.8
10^{4}	432.6	74.4

people. Images from Part-A were collected from the Internet, while images from Part-B were collected on the busy streets of Shanghai. To avoid the risk of over-fitting to the small number of training images, we trained ic-CNNs on random crops of size $\frac{H}{3} \times \frac{W}{3}$, where H and W are the height and width of a training image. In Table 1, we compare ic-CNNs with the previous state-of-the-art approaches. ic-CNNs outperform the previous approaches in three out of four cases by a large margin. On Part-B of the Shanghaitech dataset, using the one-stage ic-CNN which has a simpler architecture than the five-branch CP-CNN [16], we improve on the previously reported state of the art results by 48.3% for MAE metric and 46.8% for the RMSE metric. On Part A of the Shanghaitech dataset, we achieve a 5.1 absolute improvement in MAE over CP-CNN. Furthermore, for Part A data, the two-stage ic-CNN results in an improvement of 1.3 MAE over the one-stage ic-CNN. We also trained a three-stage ic-CNN on Part A data, which resulted in MAE = 69.4 and RMSE = 116.0. Since adding the 3^{rd} stage did not yield a significant performance gain, we did not experiment with more than three stages.

In Table 2, we analyze the effects of varying the resolution of the intermediate prediction on the overall performance. Using any resolution other than $\frac{1}{4}$ leads to a drop in the performance.

In Table 3, we analyze the effects of varying the hyperparameter λ_h on performance of ic-CNN. We use Shanghaitech Part-A dataset for this experiment. We show the MAE of the high and low resolution branches as the scalar weight λ_h is varied. λ_l is kept fixed at 10^{-2}. We can see that the LR-CNN branch performs better when λ_l is comparable with λ_h, and its performance degrades when λ_h is too large. The performance of HR-CNN improves as λ_h is varied from 10^{-4}

Table 4. Training time, number of parameters, and MAE on Part A of the Shanghaitech dataset. ic-CNN was trained on a single GPU machine (Nvidia GTX 1080 TI).

Model	Training Time	Number of Parameters	MAE
MCNN [20]	unknown	1.27×10^5	110.2
Switching CNN [15]	22 hrs	1.2×10^7	90.4
CP-CNN [16]	unknown	6.3×10^7	73.6
ic-CNN (proposed)	10 hrs	7.9×10^6	69.8

Table 5. Ablation study on Shanghaitech Part A data. HR-CNN is the high resolution branch, LR-CNN is the low resolution branch. LR-CNN alone and HR-CNN alone refer to a counting network that contains either LR-CNN or HR-CNN only. ic-CNN is our proposed approach, where both the features and the low resolution prediction map from LR-CNN are shared with HR-CNN. We also compared with two variants where either the low resolution map or the convolutional feature maps from LR-CNN is not shared with the HR-CNN.

Method	MAE	RMSE
LR-CNN alone	78.5	133.2
HR-CNN alone	136.2	204.0
HR-CNN + LR-CNN features (no low-res prediction)	75.1	129.0
HR-CNN + LR-CNN low-res prediction (no features)	77.4	130.4
ic-CNN (proposed)	**69.8**	**117.3**

to 10^2. In the extreme case when λ_h is set to 10^4, there is a large degradation in the performance of the LR-CNN branch, which affects the performance of the HR-CNN branch. When λ_h is 10^4, the low resolution prediction task is possibly ignored, and the network solely focuses on solving the high resolution task. In such a scenario, the low resolution prediction does not contain any useful information, which affects the performance of the high resolution branch HR-CNN. We obtain the best results for the HR-CNN branch when λ_h is set to 10^2. In this case, the high resolution loss does not force the network to completely ignore the low resolution task.

In Table 4, we show the training time and the number of parameters of ic-CNN, MCNN, Switching CNN, and CP-CNN. An ic-CNN takes 10 hours to train, while a Switching CNN takes around 22 hours. An ic-CNN has significantly fewer parameters than a CP-CNN and a Switching CNN. We contacted the authors of MCNN and CP-CNN, but we did not get a response for the training time of these networks.

In Table 5, we analyze the importance of each of the components of our proposed ic-CNN model. We see that both the feature sharing and the feedback of the low resolution prediction are important for ic-CNN. Removing any of these two components leads to significant drop in performance.

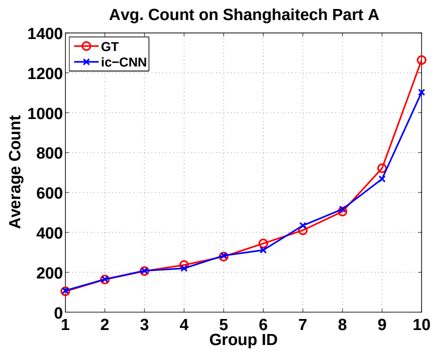

Fig. 3. Performance across different crowd density: We divide the 182 test images from Shanghaitech Part A into 10 groups on the basis of the crowd count. Each group except the last has 18 test images. We average the crowd count across a group to obtain the average count. GT is the ground truth, ic-CNN is prediction from the high resolution branch. For majority of the count groups, the difference between the average counts for ic-CNN and GT is small.

In Figure 3, we analyze the performance of ic-CNN across different groups of images with varying crowd counts.

4.3 Experiments on the WorldExpo'10 Dataset

The WorldExpo'10 dataset consists of 1132 annotated video sequences captured by 108 surveillance cameras. Annotated frames from 103 cameras are used for training and the annotated frames from the remaining 5 cameras are used for testing. We trained ic-CNN networks using random crops of sizes $\frac{H}{2} \times \frac{W}{2}$. We used the networks trained on Shanghaitech Part A for initializing the models for the experiments on the WorldExpo dataset. In Table 6, we compare ic-CNN with other state of art approaches. ic-CNN outperforms these previous approaches on three out of five cases.

4.4 Experiments on the UCF Dataset

The UCF Crowd Counting dataset [6] consists of 50 crowd images collected from the web. Each person in the dataset is annotated with a single dot annotation. The numbers of people in the images vary from 94 to 4545 with an average of 1280 people per image. The average count for the UCF dataset is much larger than the previous two datasets. Following previous works using this dataset, we performe five-fold cross validation and report the MAE and RMSE values. We trained ic-CNN networks using random crops of sizes $\frac{H}{3} \times \frac{W}{3}$. We compare ic-CNN with previous approaches and show the results in Table 7. Since the dataset is small, adding multiple stages to ic-CNN could lead to overfitting. Hence we only use one-stage ic-CNN on the UCF dataset. ic-CNN achieves the best MAE on this dataset, outperforming CP-CNN by a large margin.

Table 6. Performance of different methods on the WorldExpo'10 dataset. Switch CNN(with perspective) refers to the case when perspective maps are used to obtain the crowd density map, while Switch CNN(sans perspective) refers to the case when the perspective map isn't used.ic-CNN is our proposed two branch approach. We outperform other approaches on 3 of 6 cases.

Method	S1	S2	S3	S4	S5	Avg
Crowd CNN [19]	9.8	14.1	14.3	22.2	**3.7**	12.9
MCNN [20]	3.4	20.6	12.9	13.0	8.1	11.6
Switching CNN (sans perspective) [15]	4.4	15.7	10.0	11.0	5.9	9.4
Switching CNN (with perspective) [15]	4.2	14.9	14.2	18.7	4.3	11.2
CP-CNN[16]	**2.9**	14.7	10.5	10.4	5.8	**8.8**
ic-CNN (proposed)	17.0	**12.3**	**9.2**	**8.1**	4.7	10.3

Table 7. Performance of various methods on the UCF Crowd Counting dataset. The proposed method ic-CNN achieves the best MAE.

Method	MAE	RMSE
Lempitsky & Zisserman [10]	493.4	487.1
Idrees et. al [6]	419.5	487.1
Crowd CNN [19]	467.0	498.5
Crowdnet [1]	452.5	-
MCNN [20]	377.6	509.1
Hydra2s [13]	333.7	425.6
Switch CNN [15]	318.1	439.2
CP-CNN [16]	295.8	**320.9**
ic-CNN (proposed)	**260.9**	365.5

4.5 Qualitative Results

In Figure 4, we show some qualitative results on images from the Shanghaitech Part-A dataset obtained using ic-CNN. The first three are success cases for ic-CNN, while the last two are failure cases. In the failure cases, we see that ic-CNN sometimes misclassify tree leaves as tiny people in a crowd. In Figure 5, we show some qualitative results on images from Shanghaitech Part-B dataset.

5 Conclusions

In this paper, we have proposed ic-CNN, a two-branch architecture for crowd counting via crowd density estimation based. We have also proposed a multi-stage pipeline comprising of multiple ic-CNNs, where each stage takes into account the predictions of all the previous stages. We performed experiments on three challenging crowd counting benchmark datasets and observed the effectiveness of our iterative approach.

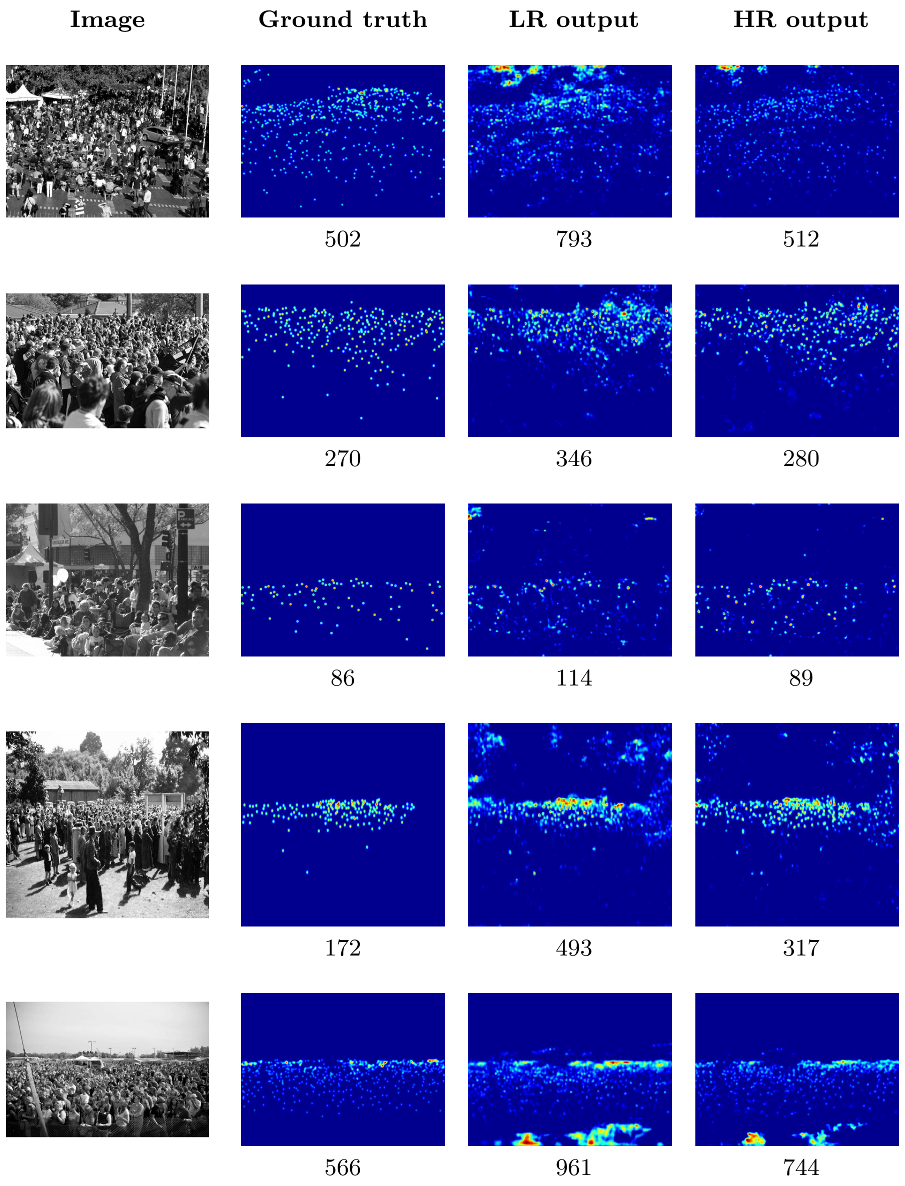

Fig. 4. Qualitative results, some success and failure cases. The four columns show the input image, ground truth annotation map, the low resolution prediction (LR output), and the high resolution prediction map (HR output). The total counts are shown below each density map. The first three rows are success cases for ic-CNN, while the last two are failure cases. ic-CNN sometimes misclassifies tree leaves as people.

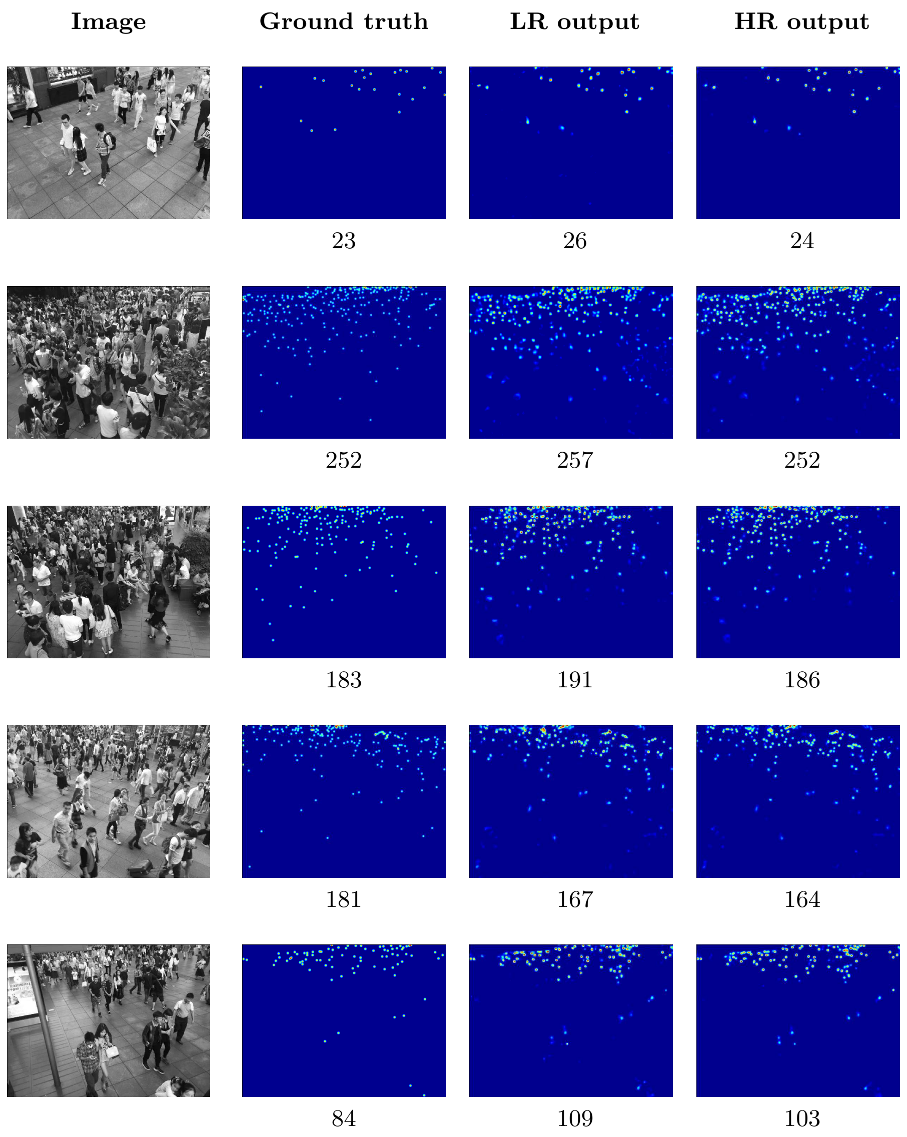

Fig. 5. Qualitative results on the Shanghaitech Part B dataset. The four columns show the input image, the ground truth annotation map, the low resolution prediction (LR output), and the high resolution prediction map (HR output). Underneath each density map is the total count, rounded to the nearest integer.

Acknowledgements. This work was supported by SUNY2020 Infrastructure Transportation Security Center. The authors would like to thank Boyu Wang for participating on the discussions and experiments related to an earlier version of the proposed technique. The authors would like to thank NVIDIA for their GPU donation.

References

1. Boominathan, L., Kruthiventi, S.S., Babu, R.V.: Crowdnet: A deep convolutional network for dense crowd counting. In: Proceedings of the ACM Multimedia Conference (2016)
2. Chan, A.B., Vasconcelos, N.: Bayesian poisson regression for crowd counting. In: Proceedings of the International Conference on Computer Vision (2009)
3. Chen, K., Loy, C.C., Gong, S., Xiang, T.: Feature mining for localised crowd counting. In: Proceedings of the British Machine Vision Conference (2012)
4. Fu, M., Xu, P., Li, X., Liu, Q., Ye, M., Zhu, C.: Fast crowd density estimation with convolutional neural networks. Engineering Applications of Artificial Intelligence **43**, 81–88 (2015)
5. Hoai, M., Zisserman, A.: Talking heads: Detecting humans and recognizing their interactions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014)
6. Idrees, H., Saleemi, I., Seibert, C., Shah, M.: Multi-source multi-scale counting in extremely dense crowd images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2013)
7. Kaufman, E.L., Lord, M.W., Reese, T.W., Volkmann, J.: The discrimination of visual number. The American journal of psychology **62**(4), 498–525 (1949)
8. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012)
9. LeCun, Y., Boser, B., Denker, J.S., Henderson, D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
10. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: Advances in Neural Information Processing Systems (2010)
11. Li, M., Zhang, Z., Huang, K., Tan, T.: Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In: Proceedings of the International Conference on Pattern Recognition (2008)
12. Lin, S.F., Chen, J.Y., Chao, H.X.: Estimation of number of people in crowded scenes using perspective transformation. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **31**(6), 645–654 (2001)
13. Onoro-Rubio, D., López-Sastre, R.J.: Towards perspective-free object counting with deep learning. In: Proceedings of the European Conference on Computer Vision (2016)
14. Pham, V.Q., Kozakaya, T., Yamaguchi, O., Okada, R.: Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In: Proceedings of the International Conference on Computer Vision (2015)
15. Sam, D.B., Surya, S., Babu, R.V.: Switching convolutional neural network for crowd counting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
16. Sindagi, V.A., Patel, V.M.: Generating high-quality crowd density maps using contextual pyramid cnns. In: Proceedings of the International Conference on Computer Vision (2017)
17. Trick, L.M., Pylyshyn, Z.W.: Why are small and large numbers enumerated differently? a limited-capacity preattentive stage in vision. Psychological review **101**(1), 80 (1994)
18. Wang, C., Zhang, H., Yang, L., Liu, S., Cao, X.: Deep people counting in extremely dense crowds. In: Proceedings of the ACM Multimedia Conference (2015)

19. Zhang, C., Li, H., Wang, X., Yang, X.: Cross-scene crowd counting via deep convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
20. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)