Counting in Dense Crowds using Deep Learning

Logan Lebanoff
Center for Research in Computer Vision
University of Central Florida
Orlando, FL 32816, USA
loganlebanoff@knights.ucf.edu

Haroon Idrees
Center for Research in Computer Vision
University of Central Florida
Orlando, FL 32816, USA
haroon@cs.ucf.edu

Abstract

We propose a method for counting the number of people in images of large crowds using deep learning. We fine-tune a pre-trained convolutional neural network with image patches from 50 crowd images. Various loss functions were used to find the best results for training, including a novel loss function called difference-to-sum ratio loss. Our algorithm can handle images containing crowds ranging from 100 to 4500 people. The experimental results demonstrate better performance in accuracy over previous methods.

1. Introduction

Crowds occur in everyday situations: concerts, political speeches, rallies, marathons, and in stadiums. Manual counting of each individual can be extremely time-consuming and close to impossible. Experienced personnel can estimate the number of people at an event, but even then the job can be laborious and still inaccurate. Computer vision solutions can reduce the accuracy and time taken to count in crowds.

Crowd counting can help with safety and surveillance in crowds, such as deployment of police officers and detection of unusual behavior. Public transportation can also use crowd counting to aid in the development of infrastructure to improve pedestrian flow traffic and improve the flow of people in shopping malls. In addition, this solution can be a first step in more complex analysis involving crowds, such as tracking and anomalous behavior detection. Better crowd analysis can then prevent dangerous situations like stampedes in pilgrimages and overcrowding during parades and concerts.



Groundtruth: **2550** Result: **2407**



Groundtruth: **1961** Result: **1982**

Figure 1. Our sample crowd counting results from two crowd images from the testing dataset.

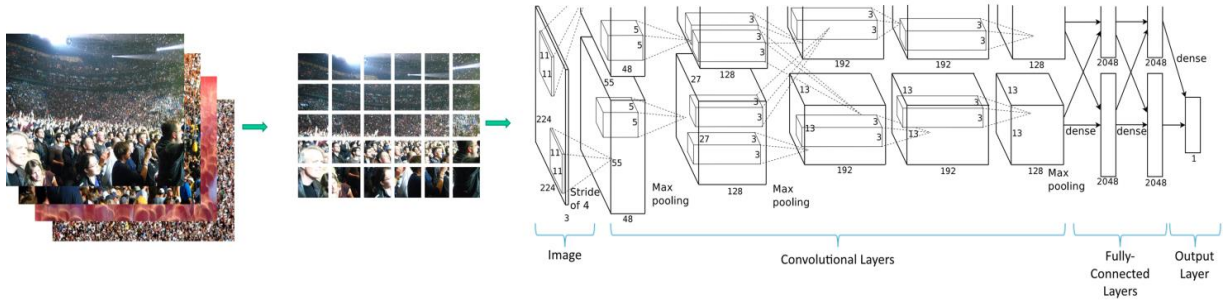


Figure 2. This figure shows the pipeline for training the network to count the number of people in a crowd. Each crowd image is divided into patches, which are then input to the network. For our regression approach, the output layer has a single number, representing the number of people in the image.

Most existing methods for counting in crowds only perform well in small to medium density situations. Conversely, our method is specifically trained to count the number of people in very dense crowd images with up to 4500 people.

Traditional human, head, and face detectors do not perform well due to the extreme density of the crowd images and the small scale of individuals in those images. Instead, humans can be treated as a kind of texture, with the shape of head and shoulders for each person. The proposed approach uses this fact. The idea behind using deep learning is that the neural network can learn these features of humans in crowds. After learning these features, the network can effectively count the number of instances that exhibit the features.

Another contribution of the proposed method is the use of a new loss function for the last layer of the network, which we call difference-to-sum ratio loss. This function calculates error more effectively for our problem; the loss function takes into account the percentage error as opposed to the absolute error.

2. Related Work

Our problem differs from existing research done in this area. Most work already done for counting people works well only for groups from about 10 to 60 people. Chan *et al.* [1] tested on the USCD dataset with density of 11 - 46 individuals per frame. Chen *et al.* [2] tested on the Mall dataset, which has 13 - 53 people per frame. The PETS dataset, which was used by Ferryman and Ellis [3], contains 3 - 40 people per frame. These methods use image segmentation or estimate the coarse density range in local regions.

Our method can more accurately estimate extremely dense crowds ranging from 100 to 4500 people in one image. The existing methods above perform poorly on this data, due to the small scale of each individual.

Ge *et al.* [4] and Li *et al.* [5] use human detections to count individuals in crowds. However, this kind of method is not effective for high density crowds because detections of humans or heads are difficult. This problem is due to low resolution, occlusion and clutter, and very few pixels per person.

In addition, many of the crowd datasets used to train and test for counting are applied on

videos. These models rely on information taken from multiple frames of the same video to perform well. In contrast, the proposed approach is designed to run on still images rather than video.

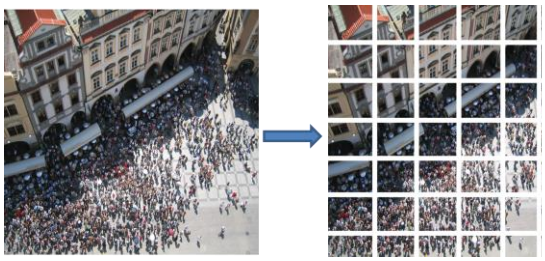


Figure 3. This figure shows a crowd image split into patches of only one size, as per our previous method.

3. Approach

Given a crowd image, we wish to estimate the number of people in that image. We have 50 crowd images that have been annotated with the number of individuals and the location of each individual in the image. Each image is divided into smaller patches of varying sizes, which makes up the training dataset.

These patches are then input into a pre-trained CNN for training. The last two layers of the network are trained for ten epochs using the dataset.

Finally the trained network runs on the testing dataset. The testing images are similarly divided into patches and passed into the trained network. The output of the network is the expected number of people in each image patch. The counts for the patches are then combined to get the overall count for the whole crowd image.

3.1 Patch Division

We have 50 annotated crowd images to use to train and test. However, 50 images are not nearly sufficient to train a CNN. This is one of the reasons we split the images into patches -- to effectively generate more data.

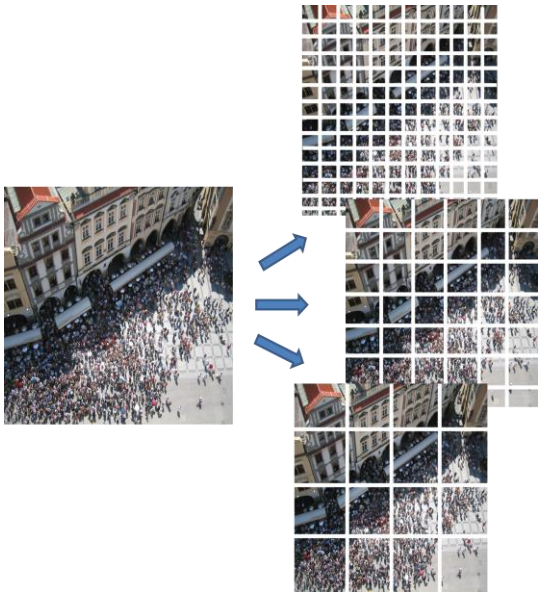


Figure 4. This figure shows a crowd image split into patches of varying sizes, as done in the proposed approach.

Another justification for dividing into patches is to make the human pseudo-detections more scale-invariant. Our previous method divided the images into 384 x 384 patches. In the proposed method, the images are split into patches of three different sizes: 288 x 288, 384 x 384, and 480 x 480 pixels. By using different scales for training, the network is more robust to scale and is less prone to overfitting from lack of data.

3.2 Loss Functions

We employed several loss functions for the CNN. These functions depend on whether crowd counting is treated as a classification problem or a regression problem.

3.2.1 Classification

In classification, the output layer is the same size as the number of classes. For our crowd counting problem, the maximum number of people in one patch is 260, so we create 261 classes for each possible count (0 - 260).

The network outputs a vector of size 261, where each value represents the confidence score for its index. For testing, the index with the highest confidence score is chosen as the result from our method. Calculating the weighted average did not perform better than simply taking the best score.

Softmax loss is most commonly used for classification problems, such as those for the ImageNet dataset. The equations for softmax loss (1) and its derivative for back propagation (2) are defined as:

$$y = -\sum_{ij}\left(x_{ijc} - \log\sum_{d=1}^{D} e^{x_{ijd}}\right) \quad (1)$$

$$\frac{dz}{dx_{ijd}} = -\frac{dz}{dy}\left(\delta_{d=c} - y_{ijc}\right) \quad (2)$$

3.2.1 Regression

Treating crowd counting as a regression problem should perform better than classification. When calculating the error, the network knows the distance from the result to the correct value. Because of this information, the regression model should learn quickly and compute the weights more accurately than a classification model.

The output layer of our regression model contains a single number, representing the number of people that our model counted in the image patch. Euclidean loss is often used as a loss function for regression, and its equation and derivative are shown:

$$y = \frac{1}{2}\sum(x - x_0)^2 \quad (3)$$

$$\frac{dy}{dx} = \sum(x - x_0) \quad (4)$$

However, Euclidean loss only accounts for the absolute difference in error. The Euclidean loss treats the error between 0 and 10 the same as the error between 100 and 110. Clearly, this is not the best way to calculate the loss.

In order to solve this discrepancy, we introduce a new loss function, which we call difference-to-sum ratio loss. This new ratio loss function minimizes the normalized error, so that the error for smaller counts is penalized the same as that for larger counts. The equations are as follows:

$$y = \sum(x + 2x_0(ln(x + x_0) \quad (5)$$

$$\frac{dy}{dx} = \sum(\frac{x - x_0}{x + x_0}) \quad (6)$$

3.3 Pre-trained Models

We decided to fine-tune a pre-trained CNN rather than train a model from scratch to save training time. Two pre-trained networks were used in experiments: the AlexNet model and the VGG 16-layer model. Both were trained on the ImageNet classification dataset.

AlexNet contains eight weight layers, while VGG-16 contains sixteen weight layers. AlexNet actually performed better in our experiments, which we believe is because VGG-16 is a much deeper network, and deeper learning generally requires more data to train. The dataset used contains about 15,000 image patches, whereas we should

expect to use maybe millions of images for a very deep network like VGG-16.

4. Experiments

The dataset used for the proposed approach was collected from 50 crowd images that were publicly available on the Internet. Each image contains between 94 and 4543 people, averaging 1280 people per image. The pictures were taken from a variety of events, including concerts, political speeches, pilgrimages, and sports events. Each image was annotated manually, including the location of each head in the image.

For training, 40 images were randomly selected in the training set, while the remaining 10 images were used for testing. Two measures were used for the results of our experiments: mean of the Absolute Difference (AD), and mean of Normalized Absolute Difference (NAD). The NAD was calculated by normalizing the AD with the ground truth counts for each image. Our best result (24% error) has a 7% improvement over our previous method (31% error). This result used the difference-to-sum ratio loss function, along with the AlexNet pre-trained model and varying-sized patches. The results of our experiments can be seen in Tables 1 - 4.

Table 1: Type of loss

	Absolute Difference (AD)	**Normalized Absolute Difference (NAD)**
Euclidean Loss	334.00	51.47
Difference-to-Sum Ratio Loss	293.71	42.83
Softmax Loss	255.04	31.41

Table 2: Type of Deep Network

	Absolute Difference (AD)	**Normalized Absolute Difference (NAD)**
AlexNet	255.04	31.41
VGG 16-layer	399.18	59.44

Table 3: Effect of patches

	Absolute Difference (AD)	**Normalized Absolute Difference (NAD)**
Same-sized patches	295.13	37.96
Varying-sized patches	288.40	23.59

Table 4: Final Results

	Absolute Difference (AD)	**Normalized Absolute Difference (NAD)**
Previous method	324.42	30.91
Proposed method	288.40	23.59

5. Conclusion

Our approach can count in crowds of higher densities than most others in existing literature, and performs better than previous methods for dense crowds. We introduce the difference-to-sum ratio loss function, a novel loss function useful for calculating error in a normalized fashion. Potential improvements include training on larger datasets and possibly using a deeper pre-trained model like VGG-16.

References

[1] A. Chan, Z. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008.

[2] K. Chen, C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. In *BMVC*, 2012.

[3] J. Ferryman and A. Ellis. Pets2010: Dataset and challenge. In *AVSS*, 2010.

[4] W. Ge and R. Collins. Marked point processes for crowd counting. In *CVPR*, 2009.

[5] M. Li, Z. Zhang, K. Huang, and T. Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *ICPR*, 2008.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.