Learning to Count with CNN Boosting

Elad Walach(✉) and Lior Wolf(✉)

The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel
alandor@gmail.com, wolf@cs.tau.ac.il

Abstract. In this paper, we address the task of object counting in images. We follow modern learning approaches in which a density map is estimated directly from the input image. We employ CNNs and incorporate two significant improvements to the state of the art methods: layered boosting and selective sampling. As a result, we manage both to increase the counting accuracy and to reduce processing time. Moreover, we show that the proposed method is effective, even in the presence of labeling errors. Extensive experiments on five different datasets demonstrate the efficacy and robustness of our approach. Mean Absolute error was reduced by 20 % to 35 %. At the same time, the training time of each CNN has been reduced by 50 %.

Keywords: Counting · Convolutional Neural Networks · Gradient boosting · Sample selection

1 Introduction

Counting objects in still images and video is a well-defined cognitive task in which humans greatly outperform machines. In addition, automatic counting has many important real-world applications, including medical microscopy, environmental surveying, automated manufacturing, and surveillance.

Traditional approaches to visual object counting were based on object detection and segmentation. This direct approach assumes the existence of adequate object localization algorithms. However, in many practical applications, object delineation is limited by significant inter-object occlusions or by cluttered background. Due to these limitations, in many cases, direct approaches can lead to gross under- or over-counting.

Starting with the seminal work of Lempitsky and Zisserman [1], a density based approach is used to translate the counting problem into a regression problem. This approach is demonstrated in Fig. 1. Each object is represented by a density kernel in a density map F. The goal of the algorithm is to estimate F directly from the image, turning the discrete counting problem to a multivariate regression problem. Counting is then performed by integrating the density function over the entire image.

Recently, a method based on Convolutional Neural Networks (CNNs) has been proposed for the problem of crowd counting [2]. The network estimates the density map, which is then corrected by a second regression step. Like other

B. Leibe et al. (Eds.): ECCV 2016, Part II, LNCS 9906, pp. 660–676, 2016.
DOI: 10.1007/978-3-319-46475-6_41

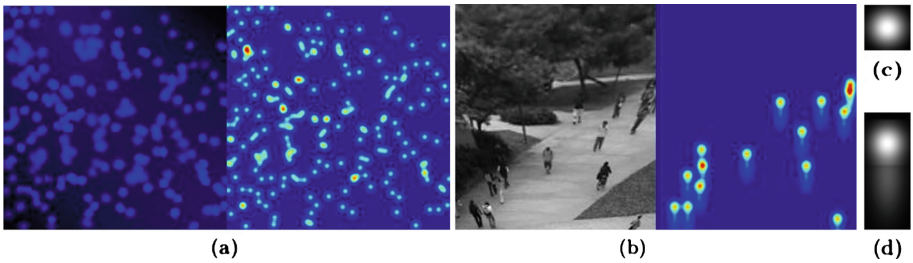


Fig. 1. Examples of density maps. (a) A synthetic fluorescence-light microscopy image [1] (left) and corresponding label density map (right). (b) A perspective normalized crowd image from the UCSD dataset [3] (left) and corresponding label density map (right). In both applications, machine learning techniques are used to estimate the appropriate density map values for each pixel. (c) The kernel used to create the microscopy density map. (d) The kernel used to create the crowd density map.

CNN based methods, this approach allows end-to-end training without the need to design any hand crafted image features and yields state of the art results as demonstrated on current object counting benchmarks: USCD [3] and UCF [4].

In this work, we adopt the CNN approach and introduce several novel modifications, which yield a significant improvement both in accuracy and performance. One such modification is in the boosting process. We propose a layered approach, where training is done in stages. We iteratively add CNNs, so that every new CNN is trained to estimate the residual error of the earlier prediction. After the first CNN is trained, the second CNN is trained on the difference between the estimation and the ground truth. The process then continues to the third CNN and so on.

Our second contribution is an intuitive yet powerful sample selection algorithm that yields both higher accuracy and faster training times. The idea is to streamline the training process by reducing the impact of the low quality samples, such as trivial cases or outliers. We propose to use the error of each sample as a measure of its quality. Our assumption is that very low errors indicate trivial cases. Conversely, very high errors indicate outliers. Accordingly, for a number of training epochs, we mute both low and high error samples. Reducing the impact of outliers is instrumental in increasing the overall accuracy of the method. At the same time, an effective decrease in the overall number of training samples reduces the training time of each CNN in the boosted ensemble.

It should be noted that layered boosting and selective sampling complement each other. Boosting increases the overall number of trained network layers, which drives up the training time. Boosting also leads to an over emphasis of misclassified samples such as outliers. Selective sampling mitigates both these undesirable effects.

2 Previous Work

The straightforward approach to counting is based on counting objects detected by an image segmentation process, see, for example, [5,6]. However, such methods are limited by the accuracy of the underlying detection methods. Accordingly, direct approaches tend to have difficulties in handling severe occlusions and cluttered backgrounds.

A direct machine learning approach was suggested in [4,7,8], which estimates the number of objects based on a predetermined set of image features such as image histograms. Naturally, the use of 1D statistics leads to a great computational efficiency. However, these global approaches tend to disregard 2D information on the object location. As a result, in some complex counting applications, accuracy may be affected.

Lempitsky et al. [1] introduced an object counting method that is based on pixel-level object density map regression. The method was shown to perform well, even in the face of a high number of objects that occlude each other. Following this work, Fiaschi et al. [9] used random forest regression in order to estimate the object density and improve training efficiency. Pham et al. [10] suggested additional improvements using modified random forests.

Deep learning is often used for tasks that are related to crowd counting such as pedestrian detection [11,12] and crowd segmentation [13]. Two recent contributions [2,14] have used deep models specifically for the application of crowd counting. In [2], a dual-loss function was suggested for the estimation of both the density function and the crowd count simultaneously. In [14], a method was proposed for counting extremely dense crowds using a CNN. In this work, the step of estimating the density function was not performed. Instead, the CNN directly estimated the number of people in the crowd. In addition, the utility of augmenting the training data with negative samples (no people) was explored.

In contrast to other methods, our approach proposes to estimate the density map directly with a single loss function. In order to mitigate the difficulty of training deep regression networks, we propose the use of relatively shallow networks augmented by the boosting framework.

Boosting Deep Networks: Boosting is a well-known greedy technique for ensemble learning. The basic idea is to literately train a new classifier that learns to fix the errors of the previous classifiers. In general, boosting is most powerful when used to combine weak models, and boosting stronger models is often not beneficial [15]. Specifically, only a few attempts have been made for boosting deep neural networks.

In [16], a hybrid method based on boosting is proposed. First, object candidates are determined based on low-level features extracted from the bottom layers of a trained CNN. AdaBoost [17] is then used to build a final classifier.

In this paper, we employ boosting in a straightforward manner, working iteratively with the same network. The method is general and the same network architecture is used for all the applications. Despite the simplicity of the proposed approach, it yields excellent results.

Sample Selection: Training deep networks is often done by utilizing very large datasets. Many methods have been proposed for data augmentation in order to increase the training set size even further. However, not all training samples are created equal. For instance, [18] proposed a sample selection scheme to choose the best samples within a sample augmentation framework. For each training sample, a continuous stream of augmented samples is created. Then, in between epochs, the network evaluates the error of each of the synthesized samples. Only high error samples are retained to form the final training set.

Sample selection is often used as a part of cascaded architectures. Cascades have been used, e.g., for face detection using either hand crafted features [19], or deep learning [20]. When constructing the next level of the cascade, the samples that did not pass the previous classifiers are filtered out.

Another commonly used method is the one of harvesting hard negative samples, which is used for face detection and for object detection in general, e.g. see [21]. Recently, in the domain of face recognition [22], it has been proposed to construct a dataset of individuals that are similar to each other and are therefore harder to identify. The face recognition network is then fine-tuned on this more challenging sample set.

3 Density Counting with CNNs

Following previous work, we define the density function as a real-valued function over the pixel grid, such that its integral over the image domain matches the object counts. The input to our method is a single image I, and, during training, a set S of image locations that correspond to the centers of the objects to be counted. The density map F is the image obtained by placing a 2D kernel P at each location $p \in S$:

$$F(x) = \sum_{p \in S} P(x - p), \tag{1}$$

where x denotes 2D image coordinates. For the microscopy dataset, we follow [1] and employ P that is a normalized 2D Gaussian kernel. For the crowd-counting experiment, we follow [2] and use a specific smoothing kernel designed for this task. This filter is a human-shaped structure composed as a superposition of two Gaussians, one for the head and one for the body. In Fig. 1, examples of ground truth density maps are presented for both microscopy and crowd counting, as well as the kernels used.

Counting in the density map domain is done by spatial integration. It is important to note, however, that using the definition above, the sum of the ground truth density F over the entire image will not match the object count exactly. This effect is caused by objects that lie very close to the image boundary so that part of the associated probability mass is located outside of the image. However, for most applications, this effect can be neglected.

Any regression algorithm can be used for counting within such a framework. The objective of the regression model is to learn the mapping from the image pixels to the density map of the image. In our deep learning method, similar

to [2], a CNN is used in order to map an image patch to the corresponding patch of the density image.

Patches randomly selected from the training images are treated as training samples, and the corresponding patches of the density map form the labels. As with other density based methods, counting is performed by summing the estimated density map. Note that unlike [2], we do not use the density map as a feature for a second regression model. Instead, we directly integrate over the density map. We have used two slightly different CNN architectures to address the two counting problems. In both cases, the input consists of patches of the input image I, and the output is an estimated density map for each patch. The CNN architecture for microscopy counting can be seen in Fig. 2(a), and the one used for crowd counting is depicted in Fig. 2(b). The differences are mainly in the size of the input patch, which, in turn, is determined by the size of the objects to be counted. In both cases, the architecture is built out of several convolutional blocks, which contain interleaving 2×2 pooling layers. After each convolutional block, we add a dropout layer [23], with a parameter 0.5. The final block is composed of a single convolutional layer (for crowd) and a series of fully connected layers without pooling. After each layer, except for the topmost hidden layer, we employ a ReLU activation function [24].

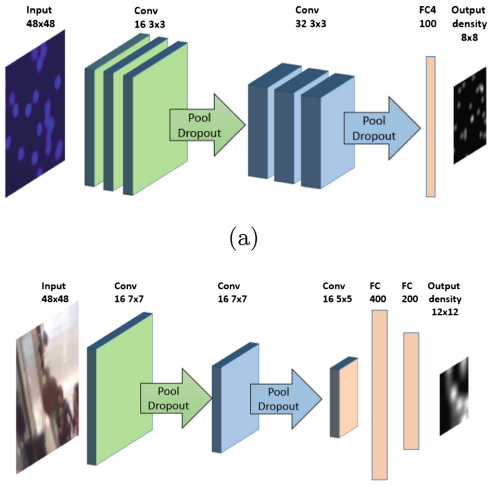


Fig. 2. The proposed CNN architecture. The basic network architecture is composed of 3 blocks. The first two blocks contain convolutional layers followed by max-pooling. The final block is composed of a single convolutional layer (for crowd) and a series of fully connected layers without pooling. (a) The cell counting problem. We use 2 convolutional blocks. Each block consists of 3×3 convolution layers, ending with 2×2 pooling and dropout. After the two convolutional blocks, we add a fully connected layer with 100 neurons. (b) The crowd counting problem. There are two 7×7 convolutions, each followed by 2×2 pooling. Finally, a 5×5 convolution layer followed by two fully connected layers are added.

Since we use two 2×2 pooling layers, the output density patch is $1/4$ the size of the original patch, in each dimension. We use the Euclidean (L2) distance as the loss function. This is in contrast to [2], which employs a dual regression loss function in order to overcome the relatively more challenging training of regression problems. In our case, we train using RMSProp [25] instead of Stochastic Gradient Descent and are able to train even without modifying the loss function. Weights were initialized using the Xavier-improved method [26].

At test time, patches are extracted from the test image using a sliding window approach. We adjust the stride such that there is a 50 % overlap. The density estimation of each pixel in the output image is obtained by averaging all the predictions of the overlapping patches that contain the given pixel. The final object count in the image is then obtained by summing all values of the recovered density map.

4 Gradient Boosting of CNNs

Gradient boosting machines belong to a family of powerful machine-learning ensemble techniques that have shown considerable success in a wide range of practical applications. Common ensemble techniques, such as random forests, rely on simple averaging of models in the ensemble. Boosting methods, including gradient boosting, are based on a different, constructive strategy for the ensemble formation. There, new models are added to the ensemble sequentially. At each iteration, a new base-learner model f_n is trained to fix the errors of the previous ensemble F_{n-1}

$$f_n(x) = \arg\min_{f_n(x)} \underbrace{E_x[\overbrace{E_y(\Psi[y, f_n + F_{n-1}(x)])}^{\text{expected loss for one sample}}|x]}_{\text{expectation over the entire dataset}}, \tag{2}$$

where $f_n(x)$ is the n-th base-learner, Ψ is the loss function and $F_{n-1} = \sum_1^{n-1} f_i$.

One can consider several different strategies for boosting, i.e. different ways to find the error-minimizing function. A well-known formulation is that of the gradient-descent method, which is called gradient boosting machines or GBMs [27,28]. The principle idea is to construct the next learner f_n to be maximally correlated with the negative gradient of the loss function of the current ensemble F_{n-1}. Therefore, this method follows gradient descent in the function space.

For the Euclidean loss, this amounts to a simple strategy of fitting a model to the current error. In our case, in each step, we fit a new CNN model $f_n = CNN_n$ to the error of the last round $F(x) - F_{n-1}$ and update the ensemble accordingly

$$F_n \leftarrow F_{n-1} + CNN_n(\theta) \tag{3}$$

An overview of our boosting mechanism is presented in Fig. 3. The validation error can be used to obtain a stopping criterion for the boosting process. As shown in Sect. 6, the gradient boosted CNN converges after a few rounds and greatly improves the prediction accuracy.

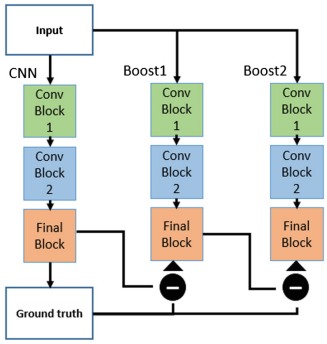


Fig. 3. The proposed boosting scheme. Each net has the same basic architecture with 2 convolutional blocks followed by the final block. In each iteration, we fix the trained ensemble and train a CNN to predict the current error.

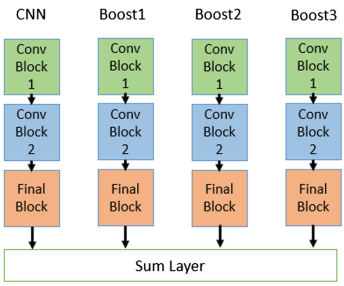


Fig. 4. Fine-tuning a four-ensemble network. Each of the networks was trained on the error of the previous ensemble. Then, all the CNNs were joined via a sum layer and training continues jointly to all network weights.

Fine-Tuning. In many applications of deep neural networks, fine-tuning is used in order to improve the results. In the framework of boosted CNNs, instead of just summing the outputs of the ensemble regressors, we can fine-tune the weights of the entire network by employing backpropagation to the resulting structure. Our fine-tuning method is presented in Fig. 4. We compose all the base-networks using a sum layer. A sum layer merges several inputs by doing an element-wise sum. Then, using backpropagation, we retrain the entire ensemble simultaneously. In order to avoid overfitting, we keep early layers fixed and allow training of the last two layers of each network, only.

5 Sample Selection

There are several instances in which we would like to weigh down certain samples. In particular, we would like to mitigate the adverse impact of trivial samples and outliers. Trivial samples are the ones that are correctly classified early on. For example, it is easy to classify points sampled from two classes, each a multivariate Gaussian, if these points lie far away from the separating hyperplane. For the task of visual object counting, uniform background patches are easy to classify, and the trained network learns to classify such background patches fairly early in the training process. Continuing to train on such patches would consume unnecessary training resources.

Algorithm 1. The sample selection scheme.

$\forall s \in allSamples$:
  $Sleep[s] \leftarrow 0$
for each training epoch **do**
  $activeSamples \leftarrow \{s | sampleSleep[s] = 0\}$
  $\forall s \in allSamples\ ,\ \ Sleep[s] \leftarrow \max(0, Sleep[s] - 1)$
  $net \leftarrow processEpoch(net, activeSamples)$
  $\forall s \in activeSamples\ ,\ \ Err(s) \leftarrow loss\ (\text{net, s})$
  $\Theta_{low} \leftarrow percentile(Err, activeSamples, 30)$
  $\Theta_{high} \leftarrow percentile(Err, activeSamples, 97)$
  $badSamples \leftarrow \{s \in activeSamples | Err[s] < \Theta_{low} \vee Err[s] > \Theta_{high}\}$
  $\forall s \in badSamples\ ,\ \ Sleep[s] \leftarrow 4$
end for

Another source of training inefficiency is due to the presence of outliers. In many practical applications, outliers are caused by the mislabeled samples. Such errors in the input data are clearly detrimental to the classifier's efficacy. Indeed, it was shown that some boosting algorithms including AdaBoost are extremely sensitive to outliers [29].

Accordingly, we propose to reduce the impact of low quality training samples by decreasing their participation in the training process. This raises the question of identifying the low quality samples described above. In our method, we employ the current error of each individual training sample as a measure of the sample's quality. A very low L2 distance between the estimated and true target, indicates a trivial sample, while a high error indicates an outlier. Therefore, samples with either high or low errors are deemed to be of low quality.

After each epoch, we continue to train only on the samples with an error rate between Θ_{low} and Θ_{high}, where Θ_{low} and Θ_{high} are thresholds chosen as a certain percentiles of the errors of the entire training set. Based on initial cross validation tests performed on 20 % of the training data of the UCSD dataset, we set Θ_{high} to be the 97 percentile and Θ_{low} as the 30 percentile throughout our experiments. The examples that did not meet the threshold criteria are removed from the training process for several epochs. Specifically, in our experiments, a low quality sample "sleeps" for four epochs. Algorithm 1 shows our proposed sample selection scheme.

One can think of other sample selection schemes. For instance, another scheme could be weighting each sample's gradients according to its error. However, a temporal elimination of a sample has clear advantages in terms of the training time, since irrelevant training samples are completely removed and not just weighted down.

Note that for the above mentioned parameters, at each epoch, 33 % of the active samples are removed. Viewed as a timed process in which the number of active samples converge, we obtain that at each round the same number of samples are removed. Let N be the total number of training samples. At the steady state, the following equation holds: $N = x + 4 \times 0.33 \times x$, where x is

the number of active samples. Therefore, the number of active training samples converges to $x \approx 0.43N$. In other words, at any given epoch, after an initial number of epochs, only about 43 % of all the samples actively participate in the training process.

Our sample selection approach is simple and straightforward. Nevertheless, as demonstrated in the experiments below, it yields a significant improvement both in terms of training time and, especially for noisy labeled data, also in terms of accuracy.

6 Experiments

We compare our algorithm to the state of the art in two domains: Bacterial cell images and crowd counting. Overall, our experiments are more extensive than any previous counting paper. The only dataset that seems to be missing is the Expo crowd dataset [2], which is not publicly available. Additional experiments holding out 5 % of the tagging are done in order to demonstrate the method's robustness to outliers. Finally, we present surprising results for depth estimation from a single image, which are obtained using the same network we propose for the completely different task of crowd counting.

6.1 Bacterial Cells Microscopy Images

The dataset presented in [1] is composed out of 200 simulated fluorescence microscopy images of cell cultures, each containing 171 ± 64 cells on average. 100 images are reserved for training and validation, and the remaining 100 for testing. Each image has a labeled equivalent with dots marked at the center of each cell. The label density map was calculated by smoothing this point density map using a Gaussian kernel with $\sigma = 3$ pixels.

Following [9], we discard the green and red channels of the raw images and use only the blue channel. From each training image, we take 1600 random 32×32 patches. Our CNN architecture is the one presented in Fig. 2(a).

The results are summarized in Table 1. As in other counting benchmarks, the mean absolute error (MAE) is used for evaluating the accuracy of each method. As can be seen, for the given data set, without boosting, the single network does not achieve good results. However, with the increase in the number of boosting stages, MAE is reduced yielding a significant (more than 30 %) improvement over the state of the art results. While the improvement in accuracy following the booting rounds is significant, it seems that more than four rounds (five networks) are not necessary and even detrimental. This is probably due to overfitting the remaining error. On this very small dataset of 100 images, we could not improve results using fine-tuning.

The boosted classifier, which consists of multiple networks, has increased capacity. Therefore, we perform additional experiments in order to rule out the possibility that the increased performance is simply due to the increase in the capacity. For this purpose, we have added more convolutional layers creating

Table 1. MAE on the microscopy dataset. We present literature results as well as results for our boosted network, following 1–6 rounds of boosting. Deeper networks (without boosting) and ensemble of multiple CNNs are also shown. In our terminology, 1 boost means two networks. For completeness, we also present (rightmost column) MAE on the test set without sample selection.

Method	MAE test	Method	MAE test	MAE validation	No selection
Detection + correction [1]	4.9	Our model (no boosting)	6.82	8.59	6.93
Density + MESA [1]	3.5	Boosted CNN (1 boost)	3.20	3.46	3.42
Regression trees [9]	3.2	Boosted CNN (2 boosts)	2.81	3.00	2.71
Ensemble of 2 CNNs	6.71	Boosted CNN (3 boosts)	2.42	2.50	2.39
Ensemble of 3 CNNs	6.54	Boosted CNN (4 boosts)	2.19	2.16	2.21
Ensemble of 4 CNNs	6.42	Boosted CNN (5 boosts)	2.33	2.18	2.41
Ensemble of 5 CNNs	6.45	Fine-tuned (4 boosts)	2.19	2.16	2.22
Ensemble of 6 CNNs	6.44	CNN twice as deep	5.40	5.62	5.22
Ensemble of 7 CNNs	6.43	CNN three times as deep	16.42	17.39	14.68

networks twice and three times as deep. The results show that a network twice as deep is better than the shallow network we use for boosting. However, boosting significantly outperforms increase in the network depth. The network three times as deep is much worse than even the shallow network, probably due to overfitting and the difficulty of training very deep networks.

It is well known that one can improve results by creating an ensemble of CNNs trained from different random starting points. Hence, we have performed a second experiment applying this technique. The same CNN, as the one utilized for boosting, is used for the ensemble experiments. Clearly, the boosting approach outperforms that of ensemble averaging.

6.2 Crowd Counting Benchmarks

The UCSD dataset [3] is comprised of a 2000-frame video chosen from one surveillance camera on the UCSD campus. The video in this dataset was recorded at 10 fps with a frame size of 158×238. The labeled ground truth is at the center of every pedestrian. The ROI and perspective map are provided in the dataset. We follow the setup of [2], and perform perspective normalization such that a ground area of 3-m by 3-m is mapped to a 48 pixel by 48 pixel region.

The benchmark sets aside frames 601–1400 as the training data and the remaining 1200 frames as the test set. For training, we extract 800 48×48 random patches from each training image.

Unlike some density map models that use regression on the density map [2] as a post-processing step, our estimated count is the direct integral over the density map. Comparison with other methods performing crowd counting on the UCSD dataset is presented in the Table 2. Once again, the MAE metric is employed for the accuracy evaluation. As can be seen, the proposed boosted CNN model outperforms the best state of the art method by over 30 %. In this dataset,

Table 2. MAE for different techniques applied on the UCSD crowd dataset. For completeness, we also present (rightmost column) MAE on the test set for a single Gaussian kernel instead of the human shaped kernel that is composed out of two guassians.

Method	MAE test	MAE validation	MAE test Single-Gaussian
Density + MESA [1]	1.7	-	-
Crowd CNN Model with global regression [2]	1.6	-	-
COUNT forest [10]	1.6	-	-
Our CNN model (no boosting)	1.63	1.42	1.57
Boosted CNN (1 boost)	1.15	1.32	1.19
Boosted CNN (2 boosts)	1.25	1.69	1.19
Fine-tuned model (1 boost)	1.10	1.28	1.18
Twice as deep	1.82	1.88	1.91
Three times as deep	2.42	2.63	2.88
Ensemble of 2 CNNs	1.55	1.55	1.63
Ensemble of 3 CNNs	1.53	1.51	1.56

one round of boosting seems optimal. The low-capacity of this benchmark is also manifested in the low performance of the twice as deep network. In this dataset, fine-tuning the boosted network, which contains two CNNs (1 boost), does improve performance. However, for this specific benchmark, the gain is relatively small (about 5 % only).

The mall crowd counting dataset [32] contains over 60,000 pedestrians in 2,000 video sequences taken in a city-mall. We follow the dataset setting in [32] and employ frames 1–800 for training and the remaining 1200 frames as the test set. 400 random patches of size 48×48 are extracted from each training image. Table 3 presents the MAE for different state-of-the-art methods and for our approach. It is interesting to note the large initial error. We hypothesize that this large error is caused by the large variability in this dataset. It is remarkable that the boosting network, as is, without any additional adaptations is able to amend this situation. Indeed, using 2-boosting stages yields 17 % decrease in the MAE (in comparison to the best literature technique). The fine-tuned network, now composed of 3 CNNs, provides additional 3 % improvement.

The UCF 50 crowd counting dataset [4] contains only 50 densely crowded images. Following the dataset setting in [4], we split the dataset randomly and perform 5-fold cross-validation. We ignore the perceptive effect, which varies from one image to the next and is hard to estimate. To mitigate the effect of the views, we employ a 2D gaussian kernel similar to the microscopy dataset. The results are presented in Table 4. Once again, the benefits of boosting are observed. In this case, optimum is achieved for a single boosting round, which obtains a 20 % drop in MAE compared to the best literature method.

Table 3. MAE for different techniques applied on the mall crowd-counting dataset.

Method	MAE test	MAE validation
CA-RR [30]	3.43	-
COUNT forest [10]	2.50	-
One CNN	9.54	8.51
Boosted CNN (1 boost)	2.43	3.19
Boosted CNN (2 boosts)	2.08	2.31
Boosted CNN (3 boosts)	2.13	2.78
Fine-tuned (2 boosts)	2.01	2.25
Twice as deep	10.41	11.24
Three times as deep	15.37	14.42
Ensemble of 2 CNNs	6.52	7.21
Ensemble of 3 CNNs	6.67	7.19

Table 4. MAE for different techniques applied on the UCF crowd-counting dataset.

Method	MAE test	MAE validation
Density + MESA [1]	493.4	-
Idrees et al. [4]	468.0	-
Zhang et al. [2]	467.0	-
One CNN (no boost)	434.4	452.4
Boosted (1 boost)	376.2	425.2
Boosted (2 boosts)	382.2	560.3
Find-tuned (1 boost)	364.4	341.4
Twice as deep	539.2	500.3
Three times as deep	914.2	712.8
Ensemble of 2 CNNs	414.2	553.2
Ensemble of 3 CNNs	474.0	680.5

6.3 Robustness to Outliers

The sample selection process has a dramatic effect on the training time, since, at the steady state, only 43 % of samples are used at each epoch, while the amount of epochs stays the same. However, its effect on accuracy is more limited. Table 1 shows MAE, for the cell counting dataset, with and without sample selection. As one can see, the effect on accuracy is small. For example, with four boosting steps, sample selection reduces MAE from 2.21 to 2.19.

However, as shown below, in more ambiguous tagging situations, or in cases where tagging is inaccurate, sample selection becomes crucial. In order to simulate this effect, we randomly removed 5 % of points from the set S of the true object locations for the training samples.

The results are summarized in Table 5 for the cell counting benchmark and Table 6 for the mall crowd counting dataset. It is interesting to see that even a very limited 5 % corruption in the ground truth causes a very significant (up to 3-fold) increase in the MAE. However, introduction of the selective sampling

Table 5. Impact of the sample selection on the MAE when 5 % of the cells are randomly untagged in the cell microscopy benchmark.

Boost	No selection	With selection	The selection proposed in [31]
none	18.80	14.55	15.72
1	10.09	7.88	8.18
2	8.12	6.25	6.32
3	8.49	4.94	5.12
4	9.12	4.96	5.42

Table 6. Impact of the sample selection process on the MAE when a random subset containing 5% of the people in the mall dataset are untagged.

Boost	No selection	With selection	The selection proposed in [31]
none	9.03	7.82	11.39
1	6.92	2.97	4.44
2	5.49	2.52	2.46
3	3.76	2.25	2.48
4	4.01	2.64	2.81

method allows over 40 % error rate reduction (4.94 instead of 8.49 for three boosting steps in the cell dataset).

It is interesting to note that, in the mall dataset, our method applied on the noisy ground truth data achieves better accuracy than the best literature result obtained on the uncorrupted truth set (2.25 compared to 2.50).

In addition, we evaluated the sample selection scheme proposed in [31], which employs a robust loss function based on Tukeys biweight function that weighs the training samples based on the residual magnitude. As can be seen in Tables 5 and 6, our method yields lower error than state of the art.

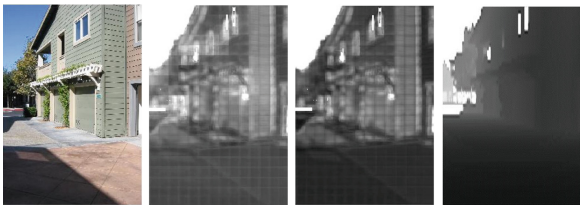
6.4 Depth Estimation

Since the proposed boosting method, at the core of our method, is general, it can be applied outside the realm of object counting. There are many image to image regression problems with a similar structure to the density estimation task. We arbitrarily select the problem of depth estimation from a single image.

The Make3D range image dataset [33,38] is used in the following depth estimation experiments. Each image in this dataset is of size 2272×1704 and is supplied with a 55×305 depth map. We adhere to the benchmark splits provided in [36], which consist of 400 training images and 134 test images.

For training, we resize the images by half and sample 800 112×112 patches. We use the same CNN architecture as the crowd counting(!). The only difference is the number of neurons in the final fully-connected layers. Since the last layer represents the estimated depth for a patch of size 28×28 pixels the final fully-connected layers contain 1000 and then 784 neurons instead of the original sizes of 400 and 200. The ground truth depth data of Make3D is inaccurate for depths greater than 80 m. Therefore, we follow the commonly applied post-processing described in [37] that classifies sky pixels and sets their depth to 80 m.

The results on the Make3D benchmark are evaluated using the RMS (root mean-squared) measure. Table 7 presents our boosting results in comparison to the literature, and Fig. 5 shows an example. The only literature methods, we are aware of, that outperform us are the Discrete-continuous CRF [37] and Deep convolutional neural fields [36]. Note that our method is local and does not use



(a) (b) (c) (d)

Fig. 5. Examples of Make3D [33] depth maps. (a) A test image. (b) Estimation with a single CNN. (c) Estimation after 1 additional round of boosting. (d) Ground truth.

Table 7. Results on the depth estimation Make3D dataset [33]. We present literature results as well as results for our boosted network.

Method	RMS(m) test	RMS(m) validation
Depth MRF [33]	16.7	-
Feedback cascades [34]	15.2	-
Depth transfer [35]	15.10	-
DCNF [36]	12.89	-
Discrete-continuous CRF [37]	12.60	-
Our CNN model (no boosting)	14.36	13.69
Boosted CNN (1 boost)	13.69	13.31
Boosted CNN (2 boosts)	13.61	12.57
Boosted CNN (3 boosts)	13.89	13.12
Fine-tuned model (2 boosts)	13.28	12.52

CRF models as the other leading methods do. The application of our method is direct, and does not include the common practice of working with superpixels. Nevertheless, our simplified approach is within 5.5 % of the state of the art.

7 Conclusions and Future Work

In this work, we propose two contributions that improve the effectiveness of CNNs: gradient boosting and selective sampling. The efficacy of these techniques was evaluated in the domain of visual object counting. We applied the techniques on four public benchmarks and showed that our approach yields a 20 %–30 % reduction in the counting error rate. When training the CNNs, we are able to obtain more than 50 % reduction in training time of each CNN.

An additional advantage of the proposed approach is its simplicity. We are using the same basic architecture for three different counting applications (microscopy, indoor and outdoor crowd) and achieve improved results in comparison to the state-of-the-art methods tuned to each specific application. Interestingly, in all the cases we had a similar degree of accuracy improvement over the literature, even though each benchmark has its own leading method.

In this paper, we explored the basic premise of the above methods. However, there are several improvements that we would like to explore in the future. These include an adaptive parameterization for the sample selection parameters: high and low thresholds and number of muted epochs. This can be based, for example, on the relative contribution of each sample to the weight updates.

Finally, it is our intention to extend the proposed techniques to more CNN regression applications. Such problems exist in a variety of domains including tasks that differ significantly from the task of counting such as age estimation in face images and human pose estimation.

Acknowledgments. This research is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

1. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, pp. 1324–1332. Curran Associates Inc. (2010)
2. Zhang, C., Li, H., Wang, X., Yang, X.: Cross-scene crowd counting via deep convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015
3. Chan, A.B., sheng John, Z., Vasconcelos, L.N.: Privacy preserving crowd monitoring: counting people without people models or tracking. In: CVPR, pp. 1–7 (2008)
4. Idrees, H., Saleemi, I., Seibert, C., Shah, M.: Multi-source multi-scale counting in extremely dense crowd images. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2013, pp. 2547–2554. IEEE Computer Society, Washington DC (2013)
5. Dong, L., Parameswaran, V., Ramesh, V., Zoghlami, I.: Fast crowd segmentation using shape indexing. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8, October 2007
6. An, S., Peursum, P., Liu, W., Venkatesh, S.: Efficient algorithms for subwindow search in object detection and localization. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 264–271, June 2009
7. Ryan, D., Denman, S., Fookes, C., Sridharan, S.: Crowd counting using multiple local features. In: Digital Image Computing: Techniques and Applications, DICTA 2009, pp. 81–88, December 2009
8. Chan, A.B., Liang, Z.S.J., Vasconcelos, N.: Privacy preserving crowd monitoring: counting people without people models or tracking. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–7, June 2008
9. Fiaschi, L., Koethe, U., Nair, R., Hamprecht, F.A.: Learning to count with regression forest and structured labels. In: 2012 21st International Conference on Pattern Recognition (ICPR), pp. 2685–2688, November 2012
10. 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: The IEEE International Conference on Computer Vision (ICCV), December 2015
11. Zeng, X., Ouyang, W., Wang, M., Wang, X.: Deep learning of scene-specific classifier for pedestrian detection. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part III. LNCS, vol. 8691, pp. 472–487. Springer, Heidelberg (2014)
12. Zeng, X., Ouyang, W., Wang, X.: Multi-stage contextual deep learning for pedestrian detection. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 121–128, December 2013
13. Kang, K., Wang, X.: Fully convolutional neural networks for crowd segmentation (2014). CoRR arXiv:1411.4464
14. Wang, C., Zhang, H., Yang, L., Liu, S., Cao, X.: Deep people counting in extremely dense crowds. In: Proceedings of the 23rd ACM International Conference on Multimedia, MM 2015, pp. 1299–1302. ACM, New York (2015)

15. Li, X., Wang, L., Sung, E.: A study of adaboost with SVM based weak learners. In: Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, IJCNN 2005, vol. 1, pp. 196–201, July 2005
16. Karianakis, N., Fuchs, T.J., Soatto, S.: Boosting convolutional features for robust object proposals (2015). CoRR arXiv:1503.06350
17. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting (1997)
18. Yamashita, T., Watasue, T., Yamauchi, Y., Fujiyoshi, H.: Improving quality of training samples through exhaustless generation and effective selection for deep convolutional neural networks. In: ICPR 2012 (2012)
19. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, vol. 1, pp. I–511. IEEE (2001)
20. Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5325–5334 (2015)
21. Sung, K.K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 39–51 (1998)
22. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Web-scale training for face identification (2014). CoRR arXiv:1406.5266
23. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
24. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G.J., Dunson, D.B. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2011), vol. 15, Journal of Machine Learning Research - Workshop and Conference Proceedings, pp. 315–323 (2011)
25. Tieleman, T., Hinton, G.: Lecture 6.5–RmsProp: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning (2012)
26. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015). CoRR arXiv:1502.01852
27. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**, 1189–1232 (2000)
28. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. **38**(4), 367–378 (2002)
29. Long, P.M., Servedio, R.A.: Random classification noise defeats all convex potential boosters. Mach. Learn. **78**(3), 287–304 (2009)
30. Chen, K., Gong, S., Xiang, T., Loy, C.C.: Cumulative attribute space for age and crowd density estimation. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2467–2474, June 2013
31. Belagiannis, V., Rupprecht, C., Carneiro, G., Navab, N.: Robust optimization for deep regression (2015). CoRR arXiv:1505.06606
32. Chen, K., Loy, C.C., Gong, S., Xiang, T.: Feature mining for localised crowd counting. In: BMVC
33. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: NIPS 18. MIT Press (2005)

34. Li, C., Kowdle, A., Saxena, A., Chen, T.: Towards holistic scene understanding: feedback enabled cascaded classification models. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, pp. 1351–1359. Curran Associates Inc. (2010)
35. Karsch, K., Liu, C., Kang, S.B.: Depth extraction from video using non-parametric sampling. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 775–788. Springer, Heidelberg (2012)
36. Liu, F., Shen, C., Lin, G., Reid, I.D.: Learning depth from single monocular images using deep convolutional neural fields (2015). CoRR arXiv:1502.07411
37. Liu, M., Salzmann, M., He, X.: Discrete-continuous depth estimation from a single image. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014
38. Saxena, A., Sun, M., Ng, A.Y.: Make3D: learning 3D scene structure from a single still image. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 824–840 (2009)