CountNet: End to End Deep Learning for Crowd Counting

Bryan Wilie
Bandung Institute of Technology
Bandung, Indonesia
brywilie25@gmail.com

Samuel Cahyawijaya
Bandung Institute of Technology
Bandung, Indonesia
samuel.cahyawijaya@gmail.com

Widyawardana Adiprawita
Bandung Institute of Technology
Bandung, Indonesia
wadiprawita@stei.itb.ac.id

***Abstract*—We approach crowd counting problem as a complex end to end deep learning process that needs both a correct recognition and counting. This paper redefines the crowd counting process to be a counting process, rather than just a recognition process as previously defined. Xception Network is used in the *CountNet* and layered again with fully connected layers. The Xception Network pre-trained parameter is used as transfer learning to be trained again with the fully connected layers. *CountNet* then achieved a better crowd counting performance by training it with augmented dataset that robust to scale and slice variations.**

Keywords—transfer learning, crowd counting, deep learning

I. INTRODUCTION

Crowd counting task in deep learning community is aiming to count every head of a human being present in a crowd shown in a photograph. The crowd in the photo is usually present in a different density, hence the name of the crowd counting, in a dense and sparse crowd. The crowd counting problem is actually a counting problem, done by estimating the number of people in the crowd, in regards to the distribution of the crowd density at one gathering area.

One of the uniqueness in this deep learning task is that not only the whole photograph can be a training data, but also slices of the photograph can represent a whole photograph as a no crowd slices, sparse crowd slices, dense crowd slices, and the mixture of them all. This brings advantage to data collection process. Whilst in another deep learning task, researcher needs to acquire hundred thousands of data, 50 high resolution photograph can already make the same abundances. From this abundances, we can get millions of training data, just by using some augmenting processes.

Besides that, crowd counting is a vast perspective problem. Some of the perspective in the earlier work is the detection [12, 13, 20, 21, 22, 24] and regression [2, 3, 4, 5, 8, 23] network. Detection approach crowd counting is successful for scenes with low crowd density, but the performance on a very dense crowd is still problematic. This happened because on these dense crowd environment, usually only partial of the whole humans are visible, only head to shoulder for horizontally taken photos, and only the top of the head, for a orthogonally taken photos. For this method, parts to be detected by the method is too small, and the counting method will not detect any object that is not a crowd. This is why this method tends to underestimate the counting in a dense crowd settings, and that is still a challenge for the detection method.

While counting by detection needs big part of a human body being located, crowd counting by regression simply estimates crowd counts without knowing the location of each person. Density estimation is sometimes used as an intermediate result, and then using a linear operation, e.g. sum, a crowd counting method get the overall crowd count results [2, 3]. The regression part, in [5] for example, is using fully CNN model for counting in highly congested scenes. Different with detection based crowd counting, regression based counting tends to overestimate sparse crowd settings counting prediction. This is happening because regression method is trying to find an n-dimensional polynomial function of linear and non-linear relationship between pixels and counting from each of the pixel. The performance of this method relies on the statistical stability of the pixels data. Thus, regression method needs is to explore intrinsic statistical principle of the whole data.

Density estimation method itself is good for regression crowd counting if the intermediate output is handled again by a human processor or processed in an optimized hand engineered feature mapping. As described before, one of feature mapping used is a linear operation, which is to sum each pixels to get the crowd count. This approach should be working smoothly if the density making process could be inversed without a loss, or had an inverse for each pixels translation into the density, or if the blur filtered area's total pixels value can be retained after the filtering process, so that the density making process do not change the ground truth crowd count. Crowd counting is a task with a rich variety of low and high level features and not only has many non-linearity in its inputs, but also has many non-linearity in its outputs. This actually is not a simple counting task, it is actually a task to generalize massive non-linearity provided by differences of the crowd density.

This research approaches crowd counting task as an end to end deep learning process. This process is partly different with some previous implementation of crowd counter. Some implementation only apply deep learning algorithm until it produces the output of predicted density map, thus the title, density estimation, and then sum the predicted density map to get the predicted crowd count. By that term, the algorithm performance is limited by the chosen counting method, and the end to end deep learning process is opening that limit so that the machine can also learn better counting method as well. The limitation is illustrated in Fig. 1, and the end-to-end solution is illustrated in Fig. 2.

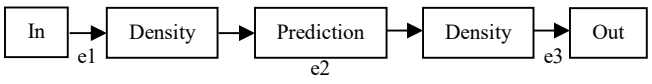

Fig. 1. Previous implementations, introducing errors as e1, e2, and e3

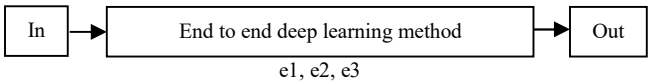

Fig. 2. End to end deep learning implementations.

As we can see on Fig. 1., the previous' implementations introduces three kinds of errors, e1, e2, and e3, from the

density making process, slicing process, prediction process and the output counting process respectively. While the prediction error (e2) is purely handled by the iterative training from the network, other errors such as density making error, slicing error and output counting error is not seen and calculated by the iterative training. That's why the network will have errors at least from the sum of e1, e2 and e3. Different approach in Fig. 2. let the iterative training process see and calculate the counting errors and takes it as loss in a learning process. This way, we let the end-to-end deep learning process also learn from the counting errors by not limiting the potential performances of the counting system. This specific advantage makes the end-to-end approach a more favorable approach for crowd counting.

Although the process tends not to limit the potential performances of the counting system, it needs a network that can generalize well to every linearity and nonlinearity available in the dataset and also a general training data. If the learning algorithm is a large-enough neural network and if it is trained with enough training data, it has the potential to do very well, and perhaps even approach the optimal error rate [15]. This end to end deep learning method can be confidently applied also because the result is not a decision, nor that the prediction error could turn into a fatal error.

Covered in this paper is a crowd counting in a spatial term, not in a temporal term. This paper also redefines regression and detection method as a crowd counting with recognition-priority method, as the result is expected to reflect the ground truth density as close as possible and try to count the crowd with the density prediction being summed. *CountNet* is a counting-priority method and in it, the counting step will be learned by the algorithm itself, not by a hardwired sum method. This is handled by using extra fully connected layers. Extensive data augmentation used is by sampling patches from the multi-scale image representation to make the counting system robust to the scale variation. Our approach is trained and evaluated on the challenging UCF_CC_50 extremely dense crowds dataset and has achieved better result.

In summary, we make the below contributions:

- We find that crowd counting task is a problem of finding a generalization of many non-linearly distributed crowd density that can't be counted or categorized easily.
- Based on that understanding, we create an end to end deep learning method, *CountNet*, and tune every hyper parameters of it so that the algorithm can be trained with a good amount of a general kind data, do the model fitting properly, and expect many linearity and non-linearity to be captured by the algorithm.
- Experimental results reveal that our method can achieve a better result on a challenging crowd counting dataset

II. Related Works

A. Crowd Counting by Detection

Earlier works approach crowd counting by detecting the counted crowd. Some of the approaches are using region proposal generators, low-level features [21, 22], binary classifiers like Naive Bayes classifier [24], and Random Forest [20]. There are also CNN based object detectors [12, 13] used to approach this crowd counting task. Detection based approach is successful for predicting scenes with low crowd density. On the occasions with high crowd density like in our chosen dataset, these approach performances are still not that good. The detection based crowd counting tends to underestimate the crowd counting predictions on the high crowd density, because only part of the whole objects are visible for localization be done by object detectors, or because the object in those highly congested areas are simply too small to be detected.

B. Crowd Counting by Regression

Counting by regression is not requiring the exact local position of each person in the crowd. In fact, some of the preliminary work use edge and detection features to learn mapping from image patterns to crowd counts. A deep convolutional neural network (CNN) is also used in a switchable training scheme for crowd counting task in [2]. The early approach of crowd counting is using a concatenated deep VGG network and parallel shallow network [3], which at the time, performed the state-of-the-art result. Despite its error to the large crowd dataset, the result is actually precise at some images in the dataset. Some others used a fully convolutional network architecture [5] with a number of max pooling layer and a 2D integrator which is an element-wise sum, to deliver the crowd count, or with a cascaded multi-task learning settings for density estimation [23], or with a combination with Gating CNN [8], which is proven to be specialized in a specific appearance and has the robustness to large appearance changes. Others implemented a multi-scale convolutional neural network to extract scale-relevant features from crowd images using a single column network based on the multi-scale blob [4]. Counting by regression is quite reliable in crowded settings and tends to overestimate predictions in a low crowd settings. This overestimations and errors in the regression based approach mainly come from the statistical stability of the data and the neediness of more instances to help explore intrinsic of that statistical stability principles.

C. Crowd Counting by Other Methods

Combining the best from both sides, there is an approach that combines results from the regression network and the detection network [11]. This superposition combination method is unique as it trains another network called 'Quality-Net block' to captures the different importance weight of two density maps by dynamically assessing the qualities of the regression prediction and detection prediction for each pixel. This approach also covers the non-linearity introduced by different crowd density that some tried to grasp using a multi-task learning method. Other novel crowd counting approach leverages abundantly available unlabeled crowd imagery in a learning-to-rank framework [19]. This approach learn from unlabeled datasets by incorporating learning-to-rank in a multi-task network which simultaneously ranks images and estimates crowd density maps.

III. Crowd Counting by CountNet

Our solution formulates the crowd counting task as a counting problem, with an understanding of linearity and non-linearity of the training set and test set. From our experiments, we understand that other than the non-linearity

of the features explored by previous' researches, there are also non-linearity introduced by each of different crowd density. To handle the size variations on the dataset, we slice the dataset to a respective size so that all training set and test set are images of typical size. Facilitating those reasons, we need to employ some of preprocessing methods, selecting the best pre-trained model that can generally captures linearity and non-linearity, and also set our learning parameters to maximize our convergence time and result. Those employment are described below.

A. Preprocessing

Enhancing features captured by the algorithm, we augment the data with several augmenting technique used before in [3] and some addition of hand-engineered sampling technique designed by ourselves. Features enhancement used here is mainly to handle scale and direction variation and to address dense crowd region.

First we do a multiscale pyramidal scaling from 0.5 to 1.3 with 0.1 steps incremented times the original full scale image resolution. The scaling will make the algorithm more robust to scale variation so that the algorithm can be trained to recognize people in more scale variation. Then the scaled image is sliced in patches of same size, so that the input to the network is in controlled size. After that, the slices then flipped in the left/right direction to further augment the dataset. This data augmentation has obtained us around 2 million slices to train. Before we sample this 2 million, we shuffle all this total sample randomly. By this augmentation, the algorithm trained is more robust to scale and direction variation and then can generalize well to most of the crowd we have in our data.

Second, we sample high relative crowd count patches more often and include also the other levels of relative crowd count patches. In our reported result, we use a maximum of 200,000 slices of training set, consisted of a maximum of 10,000 slices from the lowest relative crowd count patches, 10,000 slices again from the low relative crowd count patches, another 10,000 slices from the medium relative crowd count patches, then 60,000 slices from high relative crowd count patches, and lastly a maximum of 110,000 slices from the highest relative crowd count patches available in our 2 million of total slices. This lowest to highest relative crowd count level is calculated from the maximum and minimum crowd count range divided by five to indicate five categories, lowest, low, medium, high, highest. The highest category's upper bound is a half times the maximum crowd count from all slices. This is done to make space for more data variations from the same crowd count group. From our training results, the sampling method ended up not using 200,000 slices as training sample. This happened because there's not enough slices available from the relative crowd count group. For example, one group only have a 79,776 slices from a maximum of 110,000 slices. We tend to not augment our training set more than what's already done because our infrastructure usage is already approaching its limit from that total training set sampled.

B. Xception: Depthwise Separable Convolutions [6]

We have tried several networks to be the main predictor of our end to end deep learning system, and we choose Xception: Depthwise Separable Convolutions [6] network, developed by Keras' own author, François Chollet. Xception is a convolutional neural network architecture based entirely on depthwise separable convolution layers that map the cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks, entirely decoupled.

The Xception architecture has 36 convolutional layers forming the feature extraction base of the network. The 36 convolutional layers are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules. In short, the Xception architecture is a linear stack of depthwise separable convolution layers with residual connections. [6]

We also choose this architecture because Xception have one the best top-1 accuracy and top-5 accuracy on the ImageNet validation dataset while also have the lowest parameters count, size, and depth, compared to the recent InceptionV3 and InceptionResNetV2 [7]. By the performance on the ImageNet validation dataset, the network proven to have a good proportion of linear and nonlinear generalization in such a compact parameters count, size and depth.

Our proposed end to end deep learning network will output a predicted count for the input slice. To achieve this, we omit the top layer of the Xception network (by setting the include_top = False), and add fully connected network sized 1024 with relu activation to introduce non-linearity aspect to the network's final counting, and then add fully connected network sized 256 also with relu activation, fully connected network sized 16, and lastly, a fully connected network sized 1 to output a final predicted count. This final predicted count will be the prediction of the input slice crowd count. The illustration of our proposed end to end deep learning network is shown in Fig 3.

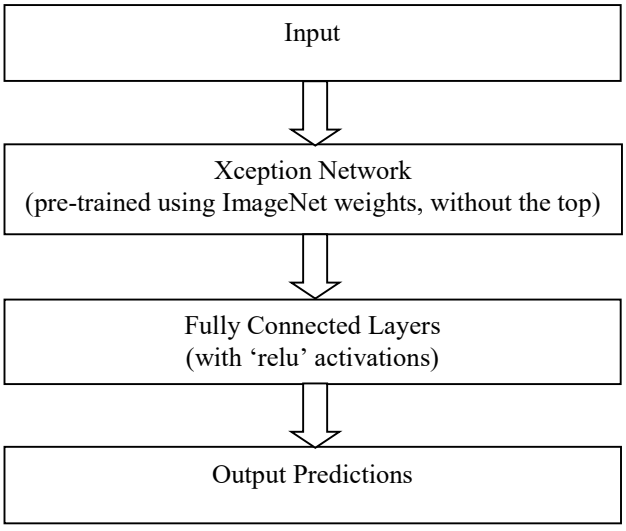

Fig. 3. Our approach of End to End Deep Learning Network

C. Learning Parameters

We use cyclical learning rates as described in [10], so that by creating new learning rate policies, our training could converge faster than other linearly and even exponentially decreasing learning rate policies. For the learning rate policies, we choose base learning rate at 1e-6, max learning rate at 1e-2, and gamma of 0.99994 in "exp range" mode. We set step size equal to 2 − 10 times the number of

iterations in an epoch. Number of iterations will be derived from number of total training set data in each epoch divided by batch size. This step size is a representation of half a cycle from a full cycle of a cyclical learning rates. The learning tends to converge at around 3 to 4 epochs, so we train our network for 5 epochs at each of our training, to make sure the convergence happens before the training finished.

D. Training Settings

For training purposes, this transfer learning method using pre-trained weights from Xception network to be trained again with the fully connected layers with glorot uniform initializer as the default initializer from Keras. We train our model using maximum of 200,000 samples from lower half of the crowd distribution (to make space for more data variation) and validate our model for each epoch, using around 1,000 samples. The maximum of 200,000 samples then will be trained as 64 mini-batches, and shuffled at each epoch. This training is done for 5 epochs for each fold of the 5 folds cross validation method.

We only train the model until the validation loss converges, as it marks the beginning of overfitting error or variance error. If the validation loss goes lower than the previous validation loss convergence point, we also take that weights as the best weights of the training session. We don't take weights that has bigger validation loss than the convergence points. This way, our model is preserved from the overfitting training result.

IV. Experimental Results

This end to end deep learning approach is evaluated for crowd counting on the challenging UCF_CC_50 [1] dataset, contains 50 grayscale crowd images with various sizes and with number of crowd count per image varies between 96 and 4631 people averaging at 1280 individuals per image.

Similar to recent works, to ensure generalization and exclude overfitting problem from testing trained dataset, our approach is evaluated using 5-folds cross validation method. The whole 50 images from UCF_CC_50 dataset is being divided randomly into 5 splits with each splits then containing 10 images. In each of the 5 folds, we consider 4 splits (40 images) for training the end to end deep learning network and 1 remaining split (10 images) to test the network. The maximum of 200,000 slices will be obtained from each of these 40 training images in regards to previously described data sampling and augmentation method. On UCF dataset, this procedures yield around 170,000 slices of training patches per fold. We train our CountNet: end to end deep learning network using Keras [15] and Tensorflow [14] deep learning framework on Tesla V-100 GPU and 64 GB of RAM. Our network was trained using Adam Stochastic Optimizer [17, 18] with learning rate policies later overruled by cyclical learning rates policies, and calculate loss as mean absolute error. The average training time per fold is around 5500 seconds.

A. Results

Measuring performance of our approach, we use Mean Absolute Error (MAE) to quantify the error of every predictions made by our approach. MAE computes the mean of absolute difference between the ground truth counts and the predicted counts for all images in this UCF dataset. The result illustrates that compared to the others implementation that creates a new network, this simple transfer learning method with a network that already has the accuracy proven for ImageNet can achieve a better accuracy than the others.

Our approach contains a random sampling and random initialization, in which making the network prone to robustness error. To completely show the result of our trainings, we use average to describe our MAE. So each trainings has its own MAE, but because we trained it 4 times, we calculate each of our trainings MAE as average MAE from 4 trainings, thus we call it average of 4 MAEs. The average of 4 MAEs is 335.3, with the details of the 4 MAEs in each folds written in Table 1.

TABLE I. 4 Times 5 Folds Training in Detail

4 Times 5 Folds Training				
5 Folds Training	**Training 1**	**Training 2**	**Training 3**	**Training 4**
Fold 1	261.8	388.1	445.3	491.8
Fold 2	269.9	287.1	323.9	279.9
Fold 3	239.2	302.3	284.2	215.8
Fold 4	554.2	444.3	494.1	458.7
Fold 5	162.5	321.2	310.4	171.8
5 Folds MAE	**297.5**	348.6	371.6	323.6
Average MAE	**335.3**			

Although our proposed method has an average MAE on 335.5, roughly seeing, the MAE is ranging from 300 to 370 and the method have a lowest prediction MAE on 297.5. This shows that our method have certain robustness problem in it. This problem should be addressed in future research so that the estimated predictions have a reliable results.

Below also shown in Fig. 4, is the detailed predicted count for each images in the UCF dataset, compared with its actual count taken from Training 1. Vertical axes is for amount of crowd count in each images. Horizontal axes is for image number in UCF_CC_50 dataset. Red lines and dots for prediction counts, blue lines and dots indicates ground truth counts.

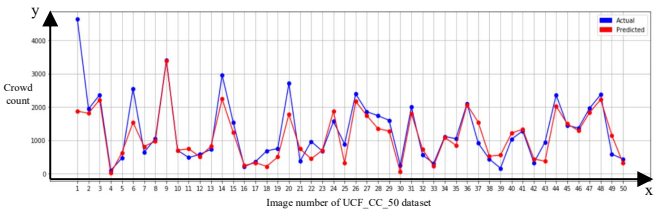

Fig. 4. Training 1 Result: Comparing Prediction and Ground Truths.

Most of the prediction is close to the ground truth counts, but we still see some of the predictions missed the ground truth counts disorderly. This prediction error present mainly because of a lack of training data. We have tried to train the proposed method with our 2 million training data, but we can't go on because of the bottleneck in our infrastructure settings.

B. Comparison with Some Early Contributions

We compare our proposed method to some of the related work and existing method referenced in this paper. The comparison is shown in Table II.

TABLE II. RESULTS COMPARISON

Methods	*MAE*
Idrees *et al.* [1]	419.5
Zhang *et al.* [2]	467.0
CrowdNet [3]	452.5
MsCNN [4]	363.7
MCNN [25]	377.6
Walach *et al.* [26]	364.4
Marsden *et al.* [5]	338.6
Proposed method	335.3

The comparison above shows that the proposed method has already achieved better results when compared to several earlier approach. The proposed method, by the MAE, proved that it is better than: global consistency constraint counts on the head detections from texture elements [1], counting by convolutional neural network (CNN) trained alternatively with crowd density and count [2], counting by using density estimation from concatenated deep and shallow network [3], using multiscale CNN [4], multi-column CNN [25], CNN with layered boosting and selective sampling [26], and Fully CNN architecture [5]. Should the robustness problem be handled in the future, this proposed method could achieve state-of-the-art performance on the crowd counting task.

V. CONCLUSION

In this paper, we proposed an end-to-end deep learning approach to deal with the crowd counting task. We showed that by using pre-trained network, the Xception network, and adding fully connected network at the top of the pre-trained network, we can achieve a better crowd counting performance by training it with augmented dataset that robust to scale and slice variations. The proposed method has achieved a better result from earlier methods that also tested on the challenging highly dense crowd dataset, the UCF_CC_50. Experimental result shows that the proposed method can achieve even better result by addressing the robustness problem on the estimated predictions.

ACKNOWLEDGMENT

The Titan XP used for this research was donated by the NVIDIA Corporation, and this work was also supported by Amazon Web Service (AWS) Educate, & Lembaga Pengembangan Inovasi dan Kewirausahaan Institut Teknologi Bandung (LPIK ITB).

REFERENCES

[1] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in CVPR 2013

[2] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in CVPR 2015

[3] L. Boominathan, S.S. Kruthiventi, and R.V. Babu, "Crowdnet: a deep convolutional network for dense crowd counting," in ACM 2016

[4] L. Zeng, X. Xu, B. Cai, S.Qiu, and T. Zhang, "Multi-scale convolutional neural networks for crowd counting," in ICCV 2017

[5] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "Fully convolutional crowd counting on highly congested scenes," in VISAPP 2017

[6] F. Chollet "Xception: Deep learning with depthwise separable convolutions," in CVPR 2017

[7] Documentation for individual models, (2018, July 10). Retrieved from https://keras.io/applications/#xception

[8] S. Kumagai, K. Hotta, and T. Kurita, "Mixture of counting cnns: adaptive integration of cnn's specialized to specific appearance for crowd counting," in Machine Vision and Applications 2018

[9] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor, "ResnetCrowd: a residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification," in AVVS 2017

[10] L. N. Smith , "Cyclical learning rates for training neural networks," in WACV 2017

[11] J. Liu, C. Gao, D. Meng, and A. G. Hauptmann, "DecideNet: counting varying density crowds through attention guided detection and density estimation," in CoRR 2017

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," in NIPS, 2015.

[13] R. Girshick, "Fast r-cnn," in ICCV 2015

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane,´R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[15] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[16] A. Ng, Machine Learning Yearning. Deeplearning.ai, 2018

[17] D. Kingma, and J. Ba, "Adam: a method for stochastic optimization," in ICLR 2015

[18] S. J. Reddi, S. Kale, S. Kumar, "On the convergence of adam and beyond," in ICLR 2018

[19] X. Liu, J. v. d. Weijer, and A. D. Bagdanov, "Leveraging unlabeled data for crowd counting by learning to rank," in CVPR 2018

[20] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, "Count forest: co-voting uncertain number of targets using random forest for crowd density estimation," in ICCV 2015

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in CVPR 2005

[22] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in CVPR 2007

[23] V. A. Sindagi and V. M. Patel, "CNN-based cascaded multi-task learning of high-level prior and density estimation for crowd counting," in AVSS 2017

[24] A. B. Chan and N. Vasconcelos, "Bayesian poisson regression for crowd counting," in ICCV 2009

[25] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single image crowd counting via multi-column convolutional neural network," in CVPR 2016

[26] E.Walach and L.Wolf, "Learning to count with cnn boosting," in ECCV 2016