Crowd Counting with Decomposed Uncertainty

Min-hwan Oh,[1]* **Peder Olsen,**[2]† **Karthikeyan Natesan Ramamurthy**[3]
[1]Columbia University, New York, NY 10025
[2]Microsoft, Azure Global Research, Redmond, WA 98052
[3]IBM Research, Yorktown Heights, NY 10598
[1]m.oh@columbia.edu, [2]peolsen@microsoft.com, [3]knatesa@us.ibm.com

Abstract

Research in neural networks in the field of computer vision has achieved remarkable accuracy for point estimation. However, the uncertainty in the estimation is rarely addressed. Uncertainty quantification accompanied by point estimation can lead to a more informed decision, and even improve the prediction quality. In this work, we focus on uncertainty estimation in the domain of crowd counting. With increasing occurrences of heavily crowded events such as political rallies, protests, concerts, etc., automated crowd analysis is becoming an increasingly crucial task. The stakes can be very high in many of these real-world applications. We propose a scalable neural network framework with quantification of decomposed uncertainty using a bootstrap ensemble. We demonstrate that the proposed uncertainty quantification method provides additional insight to the crowd counting problem and is simple to implement. We also show that our proposed method exhibits state-of-the-art performances in many benchmark crowd counting datasets.

Introduction

Recently, convolutional neural networks (CNN) have been shown to have successes in a wide range of tasks in computer vision, such as object detection, image recognition, face recognition, and image segmentation. Inspired by these successes, many CNN based crowd counting methods have been proposed. Along with density estimation techniques (Lempitsky and Zisserman 2010), CNN based approaches have shown outstanding performances over previous works that were relying on handcrafted feature extraction.

With increasing occurrences of heavily crowded events such as political rallies, protests, concerts, etc., automated crowd analysis is becoming an increasingly crucial task. The stakes can be very high in many of these real-world applications. Hence, we ask the following question:

Question: Can we use the existing neural network based methods for real-world crowd analysis?

One problem with existing CNN methods is that they only offer point estimates of counts (or density map) and do not address the uncertainty in the prediction, which can come from the model and also from data itself. When given a new unlabeled crowd image, how much can we trust the output of the model if it only provides a point estimate? Probabilistic interpretations of outputs of the model via uncertainty quantification are very important. Uncertainty quantification accompanied by point estimation can lead to a more informed decision, and even improve the prediction quality. This can is crucial for the practitioners of these crowd counting methods. With the quantification of prediction confidence at hand, one can treat uncertain inputs and special cases explicitly. For instance, a crowd counting model might return a density map (or a count) with less confidence (high uncertainty) in some area of a given scene. In this case, the practitioner could decide to pass the image – or the specific part of the image that the model is uncertain about – to a human for validation.

While Bayesian methods provide a mathematically plausible framework to deal with uncertainty quantification, often these methods come with a prohibitively computational cost. In this work, we propose a simple and scalable neural network framework using a bootstrap ensemble to quantify decomposed uncertainty for crowd counting. The key highlights of our work are:

- To the best of our knowledge, this work is the first to address uncertainty quantification of neural network predictions for crowd counting. Our method is shown to produce accurate estimated uncertainty.
- Our method decomposes uncertainties coming from the model and also from input data, which can be of independent interest for image analysis.
- Our proposed method achieves state-of-the-art level performances on multiple crowd counting benchmark datasets. The proposed architecture is more efficient than previously known state-of-the-art methods in terms of computational complexity.
- Our proposed uncertainty quantification framework is generic and independent of the architecture of an underlying network. Combined with its simplicity for implemen-

*Part of the work was done during the internship at IBM.

†Much of the work was done while the author was at IBM.

tation, we show the adaptability to other architecture.

Related Work

The previous literature on crowd counting problems can be categorized into three kinds of approaches depending on methodology: detection-based, regression-based and density-based methods.

Detection-based crowd counting is an approach to directly detect each of the target objects in a given image. A typical approach is to utilize object detectors (Leibe, Seemann, and Schiele 2005; Li et al. 2008; Wang and Yung 2009) often using moving-windows (Dollar et al. 2012). Then, the counts of targets in an image are automatically given as a byproduct of detection results. However, objects can be highly occluded in many crowded scenes and many target objects can be in drastically different scales, making detection much more challenging. These issues make detection-based approaches infeasible in dense crowd scenes.

Regression-based approaches (Chan and Vasconcelos 2009; Chen et al. 2012; Kumagai, Hotta, and Kurita 2017; Ryan et al. 2009; Shang, Ai, and Bai 2016) are proposed to remedy the occlusion problems which are obstacles for detection-based methods. Regression-based methods directly map input crowd images to scalar values of counts, hence bypassing explicit detection tasks. Particularly, a mapping between image features and the crowd count is learned. Typically the extracted features are used to generate low-level information, which is learned by a regression model. Hence, these methods leverage better feature extraction (if available) and regression algorithms for estimating counts (Shang, Ai, and Bai 2016; Arteta et al. 2014; Chan and Vasconcelos 2009; Chen et al. 2012; Seguí, Pujol, and Vitria 2015). However, these regression-based methods mostly ignore the spatial information in the crowd images.

Density-based crowd counting, originally proposed in (Lempitsky and Zisserman 2010), preserves both the count and spatial distribution of the crowd, and have been shown effective at object counting in crowd scenes. In an object density map, the integral over any sub-region is the number of objects within the corresponding region in the image. Density-based methods are generally better at handling cases where objects are severely occluded by bypassing the hard detection of every object, while also maintaining some spatial information about the crowd. (Lempitsky and Zisserman 2010) proposes a method that learns a linear mapping between the image feature and the density map. (Pham et al. 2015) proposes learning a non-linear mapping using random forest regression. However, earlier approaches still depended on hand-crafted features.

Density-based crowd counting using CNN. In recent years, the CNN based methods with density targets have shown performances superior to the traditional methods based on handcrafted features (Fu et al. 2015; Wang et al. 2015; Zhang et al. 2015). To address perspective issues, (Zhang et al. 2016) leverages a multi-column network using convolution filters with different sizes in each column to generate the density map. As a different approach to address perspective issues, (Onoro-Rubio and López-Sastre 2016) proposes taking a pyramid of input patches into a network. (Sam, Surya, and Babu 2017) improves over (Zhang et al. 2016) and uses a switching layer to classify the crowd into three classes depending on crowd density and to select one of 3 regressor networks for actual counting. (Zhang et al. 2017) incorporates a multi-task objective, jointly estimating the density map and the total count by connecting fully convolutional networks and recurrent networks (LSTM). (Sindagi and Patel 2017b) uses global and local contexts to generate a high-quality density map. (Li, Zhang, and Chen 2018) introduces the dilated convolution to aggregate multi-scale contextual information and utilizes a much deeper architecture from VGG-16 (Simonyan and Zisserman 2014). (Cao et al. 2018) proposes an encoder-decoder network with the encoder extracting multi-scale features with scale aggregation modules and the decoder generating density maps by using a set of transposed convolutions.

Limitations of the current state of the art: While density estimation and CNN based approaches have shown outstanding performances in the problems of crowd counting, less attention has been paid to assessing uncertainty in predictive outputs. Probabilistic interpretations via uncertainty quantification are important because (1) lack of understanding of model outputs may provide sub-optimal results and (2) neural networks are subject to overfitting, so making decisions based on point prediction alone may provide incorrect predictions with spuriously high confidence.

Uncertainty in Neural Networks

Much of the previous work on the Bayesian neural network studied uncertainty quantification founded on parametric Bayesian inference (Blundell et al. 2015; Gal and Ghahramani 2016). In this work, we consider a non-parametric bootstrap of functions.

Bootstrap ensemble

Bootstrap is a simple technique for producing a distribution over functions with theoretical guarantees (Bickel and Freedman 1981). It is also general in terms of the class of models that we can accommodate. In its most common form, a bootstrap method takes as input a dataset $\mathcal{D}$ and a function f_θ. We can transform the original dataset $\mathcal{D}$ into K different datasets $\{\mathcal{D}_k\}_{k=1}^K$'s of cardinality equal to that of the original data $\mathcal{D}$ that is sampled uniformly with replacement.

Then we train K different models. For each model f_{θ_k}, we train the model on the dataset $\mathcal{D}_k$. So each of these models is trained on data from the same distribution but a different dataset. Then if we want to approximate sampling from the distribution of functions, we sample uniformly an integer k from 1 to K and use the corresponding function f_{θ_k}.

In cases of using neural networks as base models f_{θ_k}, bootstrap ensemble maintains a set of K neural networks $\{f_{\theta_k}\}_{k=1}^K$ independently on K different bootstrapped subsets of the data. It treats each network as independent samples from the weight distribution. In contrast to traditional Bayesian approaches discussed earlier, bootstrapping is a frequentist method, but with the use of the prior distribution, it could approximate the posterior in a simple manner. Also, it scales nicely to high-dimensional spaces, since it only requires point estimates of the weights. However, one major drawback is that computational load increase linearly with respect to the number of base models. In the following section, we discuss how to mitigate this issue and still maintain reasonable uncertainty estimates.

Measures of uncertainty

When we address uncertainty in predictive modeling, there are two major sources of uncertainty (Kendall and Gal 2017):

1. **epistemic uncertainty** is uncertainty due to our lack of knowledge; we are uncertain because we lack understanding. In terms of machine learning, this corresponds to a situation where our model parameters are poorly determined due to a lack of data, so our posterior over parameters is broad.
2. **aleatoric uncertainty** is due to genuine stochasticity in the data. In this situation, an uncertain prediction is the best possible prediction. This corresponds to noisy data; no matter how much data the model has seen, if there is inherent noise then the best prediction possible may be a high entropy one.

Note that whether we apply a Bayesian neural network framework or a bootstrap ensemble framework, the kind of uncertainty which is addressed by either of the methods is epistemic uncertainty only. Epistemic uncertainty is often called model uncertainty and it can be explained away given enough data (in theory as data size increases to infinity this uncertainty converges to zero). Addressing aleatoric uncertainty is also crucial for the crowd counting problem since many crowd images do possess inherent noise, occlusions, perspective distortions, etc. that regardless of how much data the model is trained on, there are certain aspects the model is not able to capture. Following (Kendall and Gal 2017), we incorporate both epistemic uncertainty and aleatoric uncertainty in a neural network for crowd counting. We discuss how we operationalize in a scalable manner in the following section.

Calibration of Predictive Uncertainty

Many methods for estimating predictive uncertainty often fail to capture the true distribution of the data (Lakshminarayanan, Pritzel, and Blundell 2017). For example, a 95% posterior confidence interval may not contain the true outcome for 95% of the time. In such a case, the model is considered to be not *calibrated* (Kuleshov, Fenner, and Ermon 2018). Bootstrap ensemble methods we consider in this work are also not immune to this issue. Hence, we address this by incorporating a technique recently introduced in (Kuleshov, Fenner, and Ermon 2018), which calibrates any regression methods including neural networks. The proposed procedure is inspired by Platt scaling (Platt and others 1999) which recalibrates the predictions of a pre-trained classifier in a post-processing step. (Kuleshov, Fenner, and Ermon 2018) shows that the recalibration procedure applied to Bayesian models is guaranteed to produce calibrated uncertainty estimates given enough data.

Proposed Method

Single network with K output heads

Training and maintaining several independent neural networks is computationally expensive especially when each base network is a large and deep neural network. To remedy this issue, we adopt a single network framework that is scalable for generating bootstrap samples from a large and deep neural network (Osband et al. 2016). The network consists of a shared architecture — for example, convolution layers — with K bootstrapped heads branching off independently. Each head is trained only on its bootstrapped sub-sample of the data as described in Section . The shared network learns a joint feature representation across all the data, which can provide significant computational advantages at the cost of lower diversity between heads. This type of bootstrap can be trained efficiently in a single forward/backward pass; it can be thought of as a data-dependent dropout, where the dropout mask for each head is fixed for each data point (Srivastava et al. 2014).

Capturing epistemic uncertainty

To capture epistemic uncertainty in a neural network, we put a prior distribution over its weights, for example a Gaussian prior: $[\theta_s, \theta_1, ..., \theta_K] \sim \mathcal{N}(0, \tilde{\sigma}^2)$, where θ_s is the parameter of the shared network and $\theta_1, ..., \theta_K$ are the parameters of bootstrap heads $1, .., K$. Let x be an image input and y be a density output. Without loss of generality, we define our pixel-wise likelihood as a Gaussian with mean given by the model output: $p(y|f_\theta(x)) = \mathcal{N}\left(f_\theta(x), \sigma^2\right)$, with an observation noise variance σ^2.

For brevity of notations we overload the term $\theta_k = [\theta_k, \theta_s]$ since θ_s is shared across all samples. For each iteration of training procedure, we sample the model parameter $\hat{\theta}_k \sim q(\theta)$ where $q(\theta)$ is a bootstrap distribution. In other words, at each iteration we randomly choose which head to use to predict an output $\hat{y} = f_{\hat{\theta}_k}(x)$. Then the objective is to minimize the loss (for a single image x) given by the negative log-likelihood:

$$\mathcal{L}(\theta) = \frac{1}{D} \sum_i \frac{1}{2\sigma^2} \|y_i - \hat{y}_i\|^2 + \frac{1}{2} \log \sigma^2$$

where y_i is the i-th pixel of the output density y corresponding to input x and D is the number of output pixels. Note that the observation noise σ^2 which captures how much noise we have in the outputs stays constant for all data points. Hence we can further drop the second term (since it does not depend on θ), but for the sake of consistency with the following section where we discuss a heteroscedastic setting, we leave it as is. Now, epistemic uncertainty can be captured by the predictive variance, which can be approximated as:

$$\mathrm{Var}(y) \approx \sigma^2 + \frac{1}{K} \sum_{k=1}^{K} f_{\hat{\theta}_k}(x)^\top f_{\hat{\theta}_k}(x) - \mathbb{E}(y)^\top \mathbb{E}(y) \quad (1)$$

with approximated mean: $\mathbb{E}(y) \approx \frac{1}{K} \sum_{k=1}^{K} f_{\hat{\theta}_k}(x)$. Note that during training procedure we randomly select one output head but during test time we combine individual predictions from K heads to compute the predictive mean and the variance.

Incorporating aleatoric uncertainty

In contrast to homoscedastic settings where we assume the observation noise σ^2 is constant for all inputs, heteroscedastic regression assumes that σ^2 can vary with input x (Le, Smola, and Canu 2005; Nix and Weigend 1994). This change can be useful in cases where parts of the observation space might have higher noise levels than others (Kendall and Gal 2017). In crowd counting applications, it is often the case that images may come from different cameras and scenes. Also due to occlusion and perspective issues within a single image, it is often the case that observation noise can vary from one part of an image (or pixel) to another part (or pixel).

Following (Kendall and Gal 2017), the network outputs both the estimated density map y and the noise variance σ^2. Therefore, in our bootstrap implementation of the network, the output layer has a total of $K+1$ nodes — K nodes corresponding to an ensemble of density map predictions y and an extra node corresponding to σ^2. Let θ_σ be the parameter corresponding to the output node of the noise variance σ^2. Now, as before, we overload the term $\theta_k = [\theta_k, \theta_s, \theta_\sigma]$ since θ_σ is shared across the bootstrap sampling. We draw a sample of model parameters from the approximate posterior given by bootstrap ensemble $\hat{\theta}_k \sim q(\theta)$. But this time as described above, we have two parallel outputs, the density map estimate $\hat{y}$ and the noise variance estimate $\hat{\sigma}^2$:

$$[\hat{y}, \hat{\sigma}^2] = f_{\hat{\theta}_k}(x).$$

Then, we have the following loss given input image x which we want to minimize:

$$\mathcal{L}(\theta) = \frac{1}{D}\sum_i \frac{1}{2\hat{\sigma}_i^2}\|y_i - \hat{y}_i\|^2 + \frac{1}{2}\log\hat{\sigma}_i^2.$$

Note that this loss contains two parts: the least square residual term which depends on the model uncertainty (epistemic uncertainty) and an aleatoric uncertainty regularization term. Now, if the model predicts $\hat{\sigma}^2$ to be too high, then the residual term will not have much effect on updating the weights – the second term will dominate the loss. Hence, the model can learn to ignore the noisy data, but is penalized for that. In practice, due to the numerical stability of predicting σ^2 which should be positive, we predict the log variance $s_i := \log\sigma^2$ instead of σ^2 for the output (Kendall and Gal 2017).

Network architecture

The network architecture is composed of three major components: (1) convolutional layers as the front-end for feature extraction, (2) a dilated convolutional for the back-end which uses dilated kernels to deliver larger reception fields and to replace pooling operations, and (3) the bootstrap ensemble output layer. For the front end, we use ResNet-50 (He et al. 2016) as backbone. Note that ResNet-50 is much deeper than VGG-16 but ResNet-50 (3.8B FLOPs) has much lower complexity than VGG-16 (15.3B FLOPs). For discussion on dilated convolution, we refer the readers to (Li, Zhang, and Chen 2018). For the output layer, we use the K bootstrap ensemble heads for $\hat{y}$ and another output for $\widehat{\sigma^2}$. Each output head in the bootstrap ensemble is a fully connected layer. We call our network DUBNet where "DUB" stands for decomposed uncertainty using bootstrap ensemble. The details of the architecture are shown in Figure 1.

Training procedure

We initialize the front-end layers (the first 10 convolutional layers) in our model with the corresponding part of a pretrained ResNet-50 (He et al. 2016). For the rest of the parameters, we initialize with a Gaussian distribution with mean 0 and standard deviation 0.01. Given a training dataset of input images $X = \{x_1, ..., x_N\}$ and corresponding ground truth density maps $Y = \{y_1, ..., y_N\}$, at each iteration, we sample uniformly at random $k \in \{1, ..., K\}$ to choose an output head k and predict $[\hat{y}_n, \hat{s}_n] = f_{\hat{\theta}_k}(x_n)$ for n-th image as discussed in the previous sections. Algorithm 1 presents a single-image batch training procedure. $\hat{y}_{n,i}$ and $\hat{s}_{n,i}$ are the i-th pixel of the estimated density map and the log variance respectively corresponding to input image x_n. D_n is the number of output pixels of y_n. Due to pooling operations, the number of output pixels is the same as the number of input pixels. Adam optimizer (Kingma and Ba 2014) with a learning rate of 10^{-5} is applied to train the model.

Algorithm 1 Decomposed Uncertainty using Bootstrap

```
Require: Input images $\{x_n\}_{n=1}^N$, GT density $\{y_n\}_{n=1}^N$
Initialize parameters $\theta$
for each epoch do
    for all $n = 1$ to $N$ do
        Sample a bootstrap head $k \sim$ Uniform$\{1, ..., K\}$
        Compute predictions $[\hat{y}_n, \hat{s}_n] = f_{\hat{\theta}_k}(x_n)$
        Compute loss:
            $\mathcal{L}(\theta_k) = \frac{1}{D_n}\sum_i \frac{1}{2\exp(\hat{s}_{n,i})}\|y_{n,i} - \hat{y}_{n,i}\|^2 + \frac{1}{2}\hat{s}_{n,i}$
        Update $\theta_k$ using gradient $\frac{d\mathcal{L}(\theta_k)}{d\theta_k}$
    end for
end for
```

Recalibration of Predictive Uncertainty

Once we have a trained model $f_{\hat{\theta}}$, we compute the mean prediction $\mu(x_n) = \frac{1}{K}\sum_k f_{\hat{\theta}_k}(x_n)$ for an input image x_n. Note that $\mu(x_n)$ is a density map. We sum over all pixels in $\mu(x_n)$ to compute the predicted mean count $\bar{C}_n$. Similarly, using the (pixel-wise) predictive variance in Eq.(1), we compute the predictive standard deviation in counts $\bar{\sigma}_n$ by summing over all pixels. Then we construct a standardized residual $Z_n = (C_n - \bar{C}_n)/\bar{\sigma}_n$ where C_n is the ground-truth count for image x_n and construct a quantile target $\hat{P}(Z_n)$ which is the proportion of data whose standardized residual is below Z_n. Then using each pair $(Z_n, \hat{P}(Z_n))$, we fit an isotonic regression model $\mathcal{R}$. The recalibration procedure is summarized in Algorithm 2.

Note that the recalibration dataset $\tilde{\mathcal{D}}$ is constructed using a validation data (non-training data) and the model $\mathcal{R}$ is

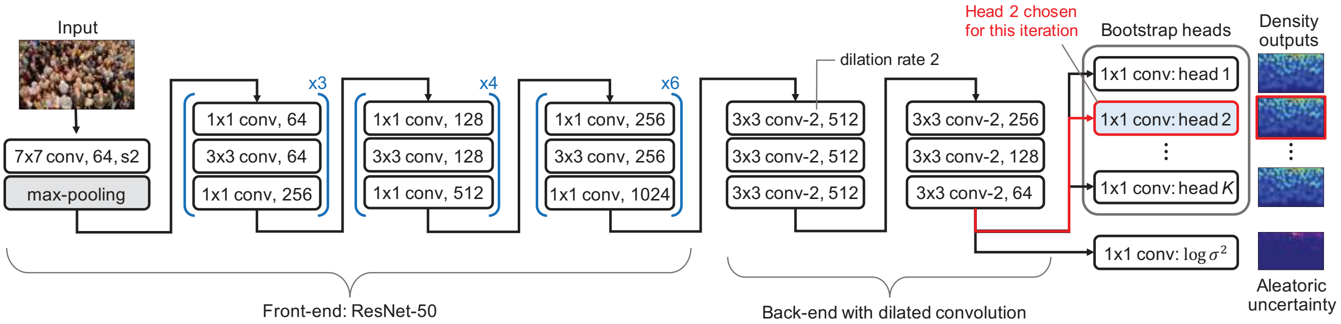

Figure 1: Network architecture of our proposed method, DUBNet. All convolutional layers use stride size 1 and SAME padding to maintain the size of the output the same as the input size. For the front-end, all the convolutional blocks contain identity shortcuts (He et al. 2016). Max-pooling layer is applied with 3×3 windows with stride size 2. The back-end layers use dilated kernels with rate 2. The output layer branches out to K bootstrap heads and the extra log-variance output.

Algorithm 2 Uncertainty Recalibration

```
Require: $\{C_n, \bar{C}_n, \bar{\sigma}\}_{n=1}^{N}$ for validation data
  1: Compute $Z_n = (C_n - \bar{C}_n)/\bar{\sigma}_n$ for all $n$
  2: Construct a recalibration dataset:
         $\tilde{\mathcal{D}} = \left\{ \left( Z_n, \hat{P}(Z_n) \right) \right\}_{n=1,\dots,N}$
     where $\hat{P}(z) = |\{C_m \mid Z_m \leq z, m = 1, \dots, N\}|/N$
  3: Train a isotonic regression model $\mathcal{R}$ on $\tilde{\mathcal{D}}$.
```

fitted on this dataset. Once $\mathcal{R}$ is learned, for a given quantile p, (e.g. 0.95 and 0.05 for 90% confidence interval) one can easily find $Z^p \in \mathbb{R}$ such that $\mathcal{R}(Z^p) \approx p$ (since $\mathcal{R}$ is a monotone function). When using at a test time where we only have $\bar{C}_n$ and $\bar{\sigma}_n$, we can construct a confidence bound by computing $\bar{C}_n + \bar{\sigma}_n Z^p$.

Experiments

In this section, we first introduce datasets and experiment details. We give the evaluation results and perform comparisons between the proposed method with recent state-of-the-art methods. For all experiments, we used $K = 10$ heads for DUBNet. We follow the standard procedure to generate the ground truth density map using Gaussian kernels.

Evaluation metrics

For crowd counting evaluation, the count estimation error is measured by two metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which are commonly used for quantitative comparison in previous works. They are defined as follows:

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N} |\hat{C}_n - C_n|, \quad \mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} (\hat{C}_n - C_n)^2}$$

where N is the number of test samples, C_n is the true crowd count for the n-th image sample and $\hat{C}_n$ is the corresponding estimated count. C_n and $\hat{C}_n$ are given by the integration over the ground truth density map $\sum_i y_{n,i}$ and over an estimated density map $\sum_i \hat{y}_{n,i}$ respectively, where i is the i-th pixel in output images. Note that during test time we use predictive mean over K bootstrap outputs as $\hat{y}$.

Ablation study

We performed ablation studies on UCF-CC 50 and UCF-QNRF datasets to validate the efficacy of our proposed method. We first compare our proposed architecture DUBNet with its variant which does not have a bootstrap ensemble output and an aleatoric uncertainty output; hence has a single fully connected layer in the output layer. We call this variant ”DUBNet w/o DUB extension.”

Table 1: Ablation studies on bootstrap extension and adaptability on different architecture

Methods	UCF-50 MAE	UCF-QNRF MAE
DUBNet w/o **DUB** extension	258.4	120.9
DUBNet	**243.8**	**105.6**
CSRNet (Li et al 2018)	266.1	135.5
CSRNet + **DUB** extension	**244.9**	**127.1**
MCNN (Zhang et al. 2016)	377.6	277.0
MCNN + **DUB** extension	**359.4**	**254.6**

To test the adaptability of our framework to other architecture, we applied the DUB extension to CSRNet (Li, Zhang, and Chen 2018) with branching outputs and an aleatoric uncertainty output. We also applied the same extension to MCNN (Zhang et al. 2016). We compare with the vanilla CSRNet and MCNN respectively. The results in Table 1 show that our proposed framework combining both aleatoric and epistemic uncertainty contributes significantly to performance on the evaluation data, and can be applied to other architecture.

Performance comparisons

We evaluate our method on four publicly available crowd counting datasets: ShanghaiTech (Zhang et al. 2016), UCF-

Table 2: Estimation errors on ShanghaiTech A/B, UCF-CC 50, and UCF-QNRF datasets

	ShanghaiTech A		ShanghaiTech B		UCF-CC 50		UCF-QNRF	
Method	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
MCNN (Zhang et al. 2016)	110.2	173.2	26.4	41.3	377.6	509.1	277	426
Cascaded-MTL (Sindagi and Patel 2017a)	101.3	152.4	20.0	31.1	322.8	397.9	252	514
Switch-CNN (Sam, Surya, and Babu 2017)	90.4	135.0	21.6	33.4	318.1	439.2	228	445
D-ConvNet (Shi et al. 2018)	73.5	112.3	18.7	26.0	288.4	404.7	-	-
L2R (Liu, van de Weijer, and Bagdanov 2018)	73.6	112.0	13.7	21.4	279.6	388.9	-	-
DensityNetwork (Idrees et al. 2018)	-	-	-	-	-	-	132	191
CSRNet (Li, Zhang, and Chen 2018)	68.2	115.0	10.6	16.0	266.1	397.5	135.5	207.4
ic-CNN (Ranjan, Le, and Hoai 2018)	68.5	116.2	10.7	16.0	260.9	365.5	-	-
SANet (Cao et al. 2018)	67.0	**104.5**	8.4	13.6	258.4	334.9	-	-
SFCN (Wang et al. 2019)	64.8	107.5	**7.6**	13.0	**214.2**	**318.2**	**102.0**	**171.4**
CAN (Liu, Salzmann, and Fua 2019)	**62.3**	**100.0**	7.8	**12.2**	**212.2**	**243.7**	107	183
DUBNet (Ours)	**64.6**	106.8	**7.7**	**12.5**	243.8	329.3	**105.6**	**180.5**

CC 50 (Idrees, Soomro, and Shah 2015), and UCF-QNRF (Idrees et al. 2018). For all datasets, we generate ground truth density maps with fixed spread Gaussian kernel. We compare our method with previously published work. In each table, the previous work which provided code or have been validated by a third party other than the original authors have been listed above our method. For completeness, we also list the recent work (without code or validation by a third party) below our method and include the numbers reported by the original authors. We highlight the best two performances in each metric.

ShanghaiTech. The ShanghaiTech dataset (Zhang et al. 2016) contains 1198 annotated images with a total of 330,165 persons. This dataset consists of two parts: Part A which contains 482 images and Part B which contains 716 images. Part A is randomly collected from the Internet and contains mostly highly congested scenes. Part B contains images captured from street views with relatively sparse crowd scenes. We use the training and testing splits provided by the authors: 300 images for training and 182 images for testing in Part A; 400 images for training and 316 images for testing in Part B.

UCF-CC 50. The UCF-CC 50 dataset (Zhang et al. 2015) is a small dataset which contains only 50 annotated crowd images. However, the challenging aspect of this dataset is that there is a large variation in crowd counts which range from 94 to 4543. Along with this variation, the limited number of images makes it a challenging dataset for the crowd counting tasks. Since training and test data split is not provided, as done in the previous literature (Li, Zhang, and Chen 2018; Zhang et al. 2015), We use 5-fold cross-validation to evaluate the performance of the proposed method.

UCF-QNRF. The UCF-QNRF dataset was recently introduced by (Idrees et al. 2018). It is currently the largest crowd dataset which contains 1,535 images with dense crowds with many of them being high-resolution images. Approximately 1.25 million people were annotated with dot annotations These images come with a wider variety of scenes and contains the most diverse set of viewpoints, densities, and lighting variations. The ground truth counts of the images in the dataset range from 49 to 12,865. Meanwhile, the median and the mean counts are 425 and 815.4, respectively. The training dataset contains 1,201 images, with which we train our model. Some of the images are so high-resolution that we faced memory issues in GPU while training. Hence, we down-sampled images that contain more than 3 million pixels. Then, we test our model on the remaining 334 images in the test dataset.

Results The results in Table 2 show that our proposed method is within the top two performers (highlighted in bold) for almost all the benchmark datasets we consider. One thing to note is that our uncertainty quantification framework is not limited to the proposed architecture, but is potentially adaptable to the other state-of-the-art architecture (as shown in the ablation study).

Estimated uncertainty validation

Estimated uncertainty is meaningful if it can capture the true distribution of the data. As mentioned earlier in the paper, we can validate whether the estimated uncertainty is well-calibrated or not by checking whether the estimated p quantile confidence interval (CI) contains the true outcome p fraction of the time. Table 3 shows the fraction of test data in each dataset whose ground truth falls in 90% CI.[1] The results suggest that our estimated uncertainty is accurate.

Table 3: Calibration results of estimated uncertainty

Dataset	Ground truth in 90% CI
ShanghaiTech A	0.907
ShanghaiTech B	0.915
UCF-QNRF	0.890

Discussion on estimated uncertainty

Figure 2 visualize the samples along with estimated density maps and their epistemic and aleatoric uncertainty from test evaluations on the ShanghaiTech data and the UCF-QNRF

[1] UCF-CC 50 dataset is not included since the uncertainty recalibration is difficult to perform due to the limited data size.

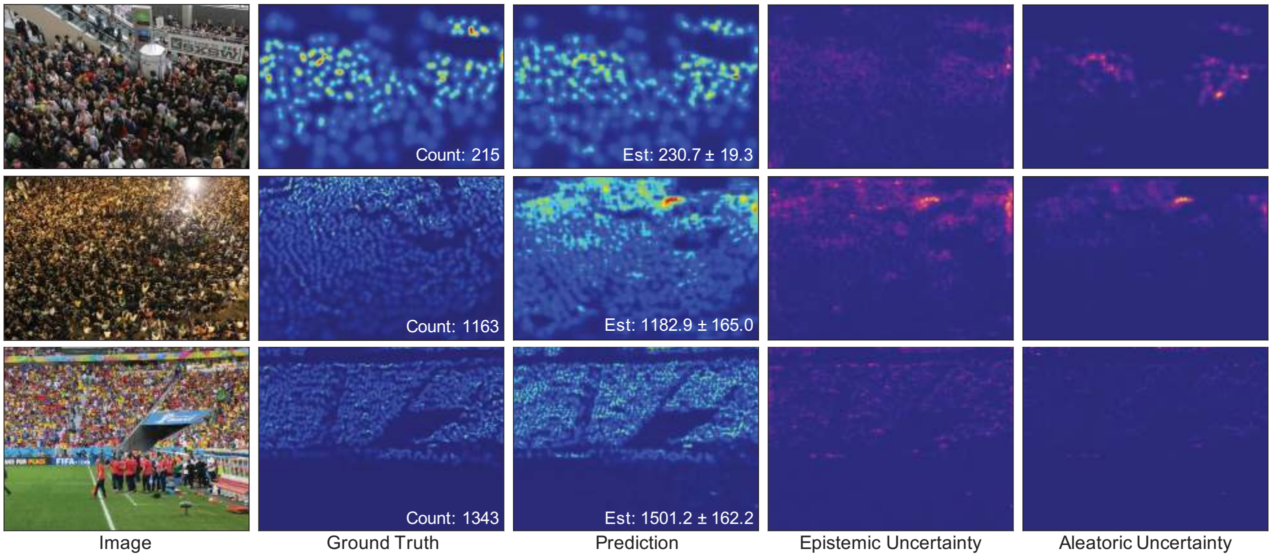

Figure 2: Qualitative results of DUBNet on the ShanghaiTech and the UCF-QNRF datasets. For each image, we demonstrate the ground truth density maps and counts, the estimated density maps and estimated counts with 90% confidence interval. We also present both estimated epistemic and aleatoric uncertainty quantification. More red color means higher uncertainty. Epistemic uncertainty captures the model's lack of knowledge about the data. Aleatoric uncertainty captures inherent noise in the data.

data. The results demonstrate that the model is generally less confident (i.e. higher epistemic uncertainty) in dense crowd regions of the images, which is natural. There appears to be a certain level of a positive correlation between epistemic and aleatoric uncertainty which is expected – since the common issues in crowd images such as occlusion and perspective issues are typically correlated with higher crowd density, this can cause both epistemic and aleatoric uncertainty to be higher. But, we also observe a notable difference in the estimated measures of uncertainty in the samples. We observe that aleatoric uncertainty is more prominent in areas where the image itself has more noise (for example, lighting glare in the second image in Figure 2) and occlusions (right side along the horizontal centerline in the first image in Figure 2). We can observe that even in very crowded scenes, when occlusions and noise are less prominent, the estimated aleatoric uncertainty can be low – for example, the stadium image (the third image in Figure 2) shows very low aleatoric uncertainty over the entire image since there are rarely occlusions or perspective issues due to the stadium seating configuration.

Conclusion

In this paper, we present a scalable and effective framework that can incorporate uncertainty quantification in prediction for crowd counting. The main component of the framework is combining shared convolutional layers and bootstrap ensembles to quantify uncertainty which is decomposed into epistemic and aleatoric uncertainty. Our proposed framework is generic, independent of the architecture choices, and also easily adaptable to other CNN based crowd counting methods. The extensive experiments demonstrate that the proposed method, DUBNet, has the state-of-the-art level performance on all benchmark datasets considered, and produces calibrated and meaningful uncertainty estimates.

References

Arteta, C.; Lempitsky, V.; Noble, J. A.; and Zisserman, A. 2014. Interactive object counting. In *European Conference on Computer Vision*, 504–518. Springer.

Bickel, P. J., and Freedman, D. A. 1981. Some asymptotic theory for the bootstrap. *The Annals of Statistics* 1196–1217.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Cao, X.; Wang, Z.; Zhao, Y.; and Su, F. 2018. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 734–750.

Chan, A. B., and Vasconcelos, N. 2009. Bayesian poisson regression for crowd counting. In *Computer Vision, 2009 IEEE 12th International Conference on*, 545–551. IEEE.

Chen, K.; Loy, C. C.; Gong, S.; and Xiang, T. 2012. Feature mining for localised crowd counting. In *BMVC*, volume 1, 3.

Dollar, P.; Wojek, C.; Schiele, B.; and Perona, P. 2012. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence* 34(4):743–761.

Fu, M.; Xu, P.; Li, X.; Liu, Q.; Ye, M.; and Zhu, C. 2015. Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence* 43:81–88.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Idrees, H.; Tayyab, M.; Athrey, K.; Zhang, D.; Al-Maadeed, S.; Rajpoot, N.; and Shah, M. 2018. Composition loss for counting, density map estimation and localization in dense crowds. *arXiv preprint arXiv:1808.01050*.

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

Kendall, A., and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kuleshov, V.; Fenner, N.; and Ermon, S. 2018. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, 2801–2809.

Kumagai, S.; Hotta, K.; and Kurita, T. 2017. Mixture of counting CNNs: Adaptive integration of CNNs specialized to specific appearance for crowd counting. *arXiv preprint arXiv:1703.09393*.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Le, Q. V.; Smola, A. J.; and Canu, S. 2005. Heteroscedastic gaussian process regression. In *Proceedings of the 22nd international conference on Machine learning*, 489–496. ACM.

Leibe, B.; Seemann, E.; and Schiele, B. 2005. Pedestrian detection in crowded scenes. In *null*, 878–885. IEEE.

Lempitsky, V., and Zisserman, A. 2010. Learning to count objects in images. In *Advances in neural information processing systems*, 1324–1332.

Li, M.; Zhang, Z.; Huang, K.; and Tan, T. 2008. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 1–4. IEEE.

Li, Y.; Zhang, X.; and Chen, D. 2018. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Computer Vision and Pattern Recognition (CVPR)*.

Liu, W.; Salzmann, M.; and Fua, P. 2019. Context-aware crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5099–5108.

Liu, X.; van de Weijer, J.; and Bagdanov, A. D. 2018. Leveraging unlabeled data for crowd counting by learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7661–7669.

Nix, D. A., and Weigend, A. S. 1994. Estimating the mean and variance of the target probability distribution. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference On*, volume 1, 55–60. IEEE.

Onoro-Rubio, D., and López-Sastre, R. J. 2016. Towards perspective-free object counting with deep learning. In *European Conference on Computer Vision*, 615–629. Springer.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, 4026–4034.

Pham, V.-Q.; Kozakaya, T.; Yamaguchi, O.; and Okada, R. 2015. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, 3253–3261.

Platt, J., et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10(3):61–74.

Ranjan, V.; Le, H.; and Hoai, M. 2018. Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 270–285.

Ryan, D.; Denman, S.; Fookes, C.; and Sridharan, S. 2009. Crowd counting using multiple local features. In *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.*, 81–88. IEEE.

Sam, D. B.; Surya, S.; and Babu, R. V. 2017. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, 6.

Seguí, S.; Pujol, O.; and Vitria, J. 2015. Learning to count with deep object features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 90–96.

Shang, C.; Ai, H.; and Bai, B. 2016. End-to-end crowd counting via joint learning local and global count. In *Image Processing (ICIP), 2016 IEEE International Conference on*, 1215–1219. IEEE.

Shi, Z.; Zhang, L.; Liu, Y.; Cao, X.; Ye, Y.; Cheng, M.-M.; and Zheng, G. 2018. Crowd counting with deep negative correlation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5382–5390.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sindagi, V. A., and Patel, V. M. 2017a. CNN-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, 1–6. IEEE.

Sindagi, V. A., and Patel, V. M. 2017b. Generating high-quality crowd density maps using contextual pyramid CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1861–1870.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Wang, L., and Yung, N. H. 2009. Crowd counting and segmentation in visual surveillance. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 2573–2576. IEEE.

Wang, C.; Zhang, H.; Yang, L.; Liu, S.; and Cao, X. 2015. Deep people counting in extremely dense crowds. In *Proceedings of the 23rd ACM international conference on Multimedia*, 1299–1302. ACM.

Wang, Q.; Gao, J.; Lin, W.; and Yuan, Y. 2019. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8198–8207.

Zhang, C.; Li, H.; Wang, X.; and Yang, X. 2015. Cross-scene crowd counting via deep convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 833–841. IEEE.

Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; and Ma, Y. 2016. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 589–597.

Zhang, S.; Wu, G.; Costeira, J. P.; and Moura, J. M. 2017. FCN-RLSTM: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 3687–3696. IEEE.