AdaCrowd: Unlabeled Scene Adaptation for Crowd Counting

Mahesh Kumar Krishna Reddy, Mrigank Rochan, Yiwei Lu, Yang Wang

Abstract—We address the problem of image-based crowd counting. In particular, we propose a new problem called unlabeled scene adaptive crowd counting. Given a new target scene, we would like to have a crowd counting model specifically adapted to this particular scene based on the target data that capture some information about the new scene. In this paper, we propose to use one or more unlabeled images from the target scene to perform the adaptation. In comparison with the existing problem setups (e.g. fully supervised), our proposed problem setup is closer to the real-world applications of crowd counting systems. We introduce a novel AdaCrowd framework to solve this problem. Our framework consists of a crowd counting network and a guiding network. The guiding network predicts some parameters in the crowd counting network based on the unlabeled images from a particular scene. This allows our model to adapt to different target scenes. The experimental results on several challenging benchmark datasets demonstrate the effectiveness of our proposed approach compared with other alternative methods.

Index Terms—Crowd Counting, Scene Adaptation, Computer Vision, Deep Learning

I. INTRODUCTION

In recent years, image-based crowd counting has become an active area of research due to its potential applications in numerous real-world domains, such as traffic monitoring, smart city planning, surveillance, security, etc. Given an input image, the goal of crowd counting is to estimate the number of people in the image. Most recent work in this area formulates the problem as estimating a density map for the input image. The pixel values in the density map indicate the crowd densities at different locations in the image. The final crowd count can be obtained by summing the values in the estimated density map. Most of recent work uses various forms of convolutional neural networks (CNN) trained in a supervised manner to perform the density estimation given an image. There has been lots of effort towards designing effective CNN architectures [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] with increasing capacities to address the problem of crowd counting. However, these approaches require a large number of training images which are expensive to collect. This becomes more problematic if we want to train a crowd counting system tuned to a specific environment (which is the case of many surveillance applications), due to the burden of collecting a large number of training images from the target environment.

Recent works [12], [13] propose to address few-shot scene-adaptive crowd counting. The problem setup in [12], [13] is motivated by the following observation. In many real-world applications, the crowd counting model only needs to work well in a very specific scene. Let us consider the application of video surveillance. Since the surveillance camera is fixed at a particular location, all test images belong to the same scene. As a result, we only need the crowd counting model to work well in this particular camera scene. The problem setup in [12], [13] assumes that we have access to a small number (e.g. one to five) of labeled image from the target scene where the model will be deployed. The model is then learned to effectively adapt to a new target scene using these few labeled examples.

Compared with the standard supervised setting, the few-shot scene-adaptation setup in [12], [13] brings crowd counting closer to real-world deployment. However, this setup still has some limitations. First of all, it still requires at least one labeled image from the end-user. Although it has drastically reduced the data requirement compared with the supervised case, it is still a burden for the end-user. Second, when deploying to the target scene, the scene adaptation method in [12], [13] involves fine-tuning several layers of a CNN model. This requires
running gradient updates and backpropagation through the network for several iterations. In practice, the computation requirement is still too high for typical surveillance cameras with limited computing capabilities.

Inspired by [12], [13], we push the envelope even further by proposing a new problem called Unlabeled Scene Adaptive Crowd Counting. Similar to [12], [13], our objective is to have a model adapted to a specific target scene during deployment. But different from [12], [13], we do not require any labeled images from the target scene. Our adaptation method only requires one or more unlabeled images from the target scene for adaptation (see Fig. 1). Since unlabeled images are fairly easy to collect in practice, this problem setup greatly reduces the data annotation effort from the user. In addition, our proposed approach also significantly reduces the required computation during adaptation. It only involves some feedforward computation without backpropagation.

We call our model AdaCrowd. It consists of two neural networks: a crowd counting network and a guiding network. The crowd counting network has some parameters that will be adapted to each scene. The guiding network is learned to map the unlabeled images to the adaptable parameters in the crowd counting network. The parameters of these two networks are learned in a way that allows effective adaptation to a new scene with using only unlabeled images from that scene.

**Summary of Contributions:** The contributions of this work are manifold. First, we present a new problem called unlabeled scene adaptive crowd counting. Different from the few-shot adaptation setup in [12], [13], our problem formulation only uses unlabeled images from the target scene for adaptation. Second, we develop an approach termed as AdaCrowd for learning the model parameters, so that they can effectively adapt to a new scene given the unlabeled images from that scene without fine-tuning at test time. Finally, we conduct an extensive evaluation of the proposed approach on several challenging benchmark datasets. Our approach significantly outperforms other alternatives.

## II. Related Work

In this section, we review prior work in two main lines of research most relevant to our work, namely crowd counting and few-shot learning.

**Crowd Counting:** Some early work uses regression-based approaches. The traditional methods resort to hand-crafted features (such as LBP, HOG, GLCM) to train various regression models (such as linear [14], piecewise linear [13], [16] and Gaussian process regression [17]) to predict the crowd count. More recently, some work [18], [19], [20] employs deep CNNs for end-to-end crowd count estimation.

A lot of recent work uses density-based methods that learn to estimate a density map of an input image. [21] learn a linear mapping function from the local image patches to their corresponding density representation, where an integral over the density map results in the crowd count. [22] propose to learn a non-linear mapping function by employing a random forest regression on multiple local image patches based on the voting mechanism for their corresponding densities to address the large variation in crowd images. Some follow up works [23], [2], [24], [3], [4], [6], [25], [8], [9], [10], [11], [26], [27], [28] exploit the non-linearity of convolutional neural networks to learn the crowd density maps on either image patches or on the whole images. Early CNN-based methods exploit multi-column architecture [1], [5], [6], [7] with different receptive fields to learn features at diverse scales. [4] propose a method to choose the best regression network with a specific receptive field for an image patch based on the label prediction by a classifier. [2] use dilated convolutional layers to expand the receptive field in place of pooling layers to address scale variation.

In terms of domain adaptation in crowd counting, [8] address the limitation of labeled real-world data by using domain adaptation from synthetic images to real images. However, this method requires prior knowledge of the crowd density statistics in the target scene to carefully select matching images in the synthetic data to overcome negative adaptation. [12] propose an approach of adaptation from a source to target scene using one labeled image. [13] propose a few-shot based meta-learning approach to perform scene adaptive crowd counting. However, this approach still requires labeled data from the target scene in order to adapt the model. [29] propose to use some meta information about the target scene (e.g. camera tilt angle, height, or perspective map) to adapt the weights in convolutional layers. In practice, the meta information is not always available, so this limits the applicability of this method. In contrast, the unlabeled scene adaptation setup proposed in our work requires minimal effort in terms of data collection from the end users and can be broadly applied in practical scenarios.

**Few-Shot Learning:** Few-shot learning in general has been widely studied in the context of image classification for learning novel classes with few data points. Some early work [30], [31] use a Bayesian method to transfer the prior knowledge about appearance from learned categories to novel categories. Recently, meta-learning [32], [33], [34], [35], [36], [37] has been explored for the quick adaptation to new categories. More specifically, the metric-based meta-learning approaches [33], [38], [39], [35], [36], [37] learn the metric as a kernel function to measure the similarity between embedding vectors of two data points. In model-based meta-learning methods [38], [39], the goal is to update the model’s parameters for fast learning. The parameter update can be either accomplished from an internal architecture [39] or by prediction from another network [38]. In contrast, the optimization-based meta-learning methods [32], [34], [40] tweak the gradient-based optimization to learn from few examples and converge within a small number of gradient steps.

## III. Problem Setup

In this section, we first review three standard setups for crowd counting that have been studied in the literature. The various limitations of these setups in real-world deployment motivates us to introduce a new problem setup for crowd counting. We believe our setup is closer to real-world scenarios compared with existing problem setups.
Supervised: This is the most common setup in previous work. This setup treats crowd counting as a purely supervised learning problem. The goal is to learn a function $f_{\theta} : x \rightarrow y$ that maps an image $x$ to a density map $y$. The model parameters $\theta$ are learned from labeled training images. There has been lots of previous work on designing powerful models [23], [1], [2], [24], [3], [4], [5], [6], [7], [25], [8], [9], [10], [11].

Domain Adaptation (DA): The standard supervised approach implicitly assumes that the training and test images are similar. In practice, training and test images often come from different domains, e.g., they might be collected from two different scenes. Due to domain shift, a model trained on the source domain often does not perform well in the target domain. Domain adaptation is a standard approach to address this domain shift. Most recent work focuses on unsupervised domain adaptation (UDA). In UDA, the source domain contains labeled data, whereas the target domain only contains unlabeled data. Most approaches of UDA [41], [42] use a domain adaptation loss to minimize the discrepancy between features of source and target domains.

The domain adaptation setup also has limitations in practical deployment. First of all, DA assumes that we have enough unlabeled data from the target domain. This might be infeasible in practice. For instance, if we consider an end user’s environment as the target domain, we may not have the authority to collect images from the target domain. Second, even if we have enough unlabeled data in the target domain, most DA approaches still require running an algorithm to perform many iterations of gradient updates and backpropagation. In practice, a crowd counting system might be deployed directly on end-user’s surveillance cameras or other devices that may not have enough computing capabilities to run the domain adaptation algorithms.

Few-Shot Scene Adaptation: Some recent works [12], [13] introduce a new problem setup called few-shot scene adaption. During deployment, this setup only needs a small number (e.g. one to five) of labeled images from a target scene. The training data consists of labeled images from multiple scenes. The model is learned in a way that enables it to quickly adapt to a new scene with only a few labeled examples. These works argue that it is often easier to get a small number of images (even with labels) in the target scene. For example, after a surveillance camera is installed, there is often a calibration process and it is possible to collect (and even label) a few images during this calibration process. Although this setup brings us closer to real-world scenarios, it still requires a few labeled images from the target domain (or scene). Additionally, the methods in [12], [13] involve several rounds of gradient updates and backpropagation to adapt model parameters based on the few labeled images. In practice, this is still taxing for end-users. In this paper, we push the limit on data requirements and overcome gradient updates at test time by proposing the following setup.

Unlabeled Scene Adaptation (this paper): This setup is similar to the few-shot case. The key difference is that this setup does not require labeled examples from the target scene during deployment. Instead, it only requires a small number of unlabeled images from the target scene as they are fairly easy to collect in practice. Similar to [12], [13], our training data consist of labeled images from multiple scenes. We propose a novel approach that learns to adapt to a target scene with only the scene-specific unlabeled images. Our approach requires minimal data collection effort from end-users. In addition, it only involves some feedforward computation (i.e. no gradient update or backpropagation) for adaption. This makes it more practical and suitable for real-world deployment.

IV. OUR APPROACH

In Fig. 2 we provide an overview of our AdaCrowd framework. The framework consists of two main networks, the crowd counting network and the guiding network. The crowd counting network is a CNN model that takes an input image and outputs its corresponding density map. Compared with standard crowd counting models, the key difference of our crowd counting network is that it has several special layers called the guided batch normalization (GBN) layers. A GBN layer plays a role similar to the standard batch normalization (BN). The main difference is that the affine parameters of a BN layer are determined from the mini-batch of data, while the parameters of the GBN layer are directly predicted by the guiding network. Most parameters of the crowd counting network are shared across different scenes. But the parameters of GBN layers change to adapt to different scenes. Due to this adjustable nature of GBN parameters, our model can learn to adapt to different scenes. Another important component in the architecture is the guiding network. This network takes the unlabeled images from a specific target scene as its input and outputs the GBN parameters for this scene. During training, the guiding network learns to predict GBN parameters that work well for the corresponding scene. At test time, we use the guiding network to adapt the crowd counting network to a specific target scene.

Guided Batch Normalization: In our AdaCrowd framework, we propose a new conditional normalization layer called the Guided Batch Normalization (GBN) layer. For ease of presentation, let us consider one particular layer in a CNN model and see how to design the GBN layer to be inserted after this layer in the model. We consider a mini-batch of $B$ examples $x_i : i = 1, 2, ..., B$, where $x_i$ is the CNN feature map at this layer for the $i$-th example, i.e. $x_i \in \mathbb{R}^{H \times W \times D}$ where $H \times W$ are the spatial dimensions and $D$ is the channel dimension. Similar to Batch Normalization [43], in GBN we normalize the activation to have zero mean and unit variance along the channel dimension over the mini-batch during training. However, unlike BN which learns the affine transformation parameters $\gamma$ and $\beta$ over the examples $x_i : i = 1, 2, ..., B$ in the mini-batch, in GBN, we directly predict them using the guiding net. The overall computation in GBN can be summarized as follows:

$$\hat{x}_i[h, w, d] = GBN(x_i[h, w, d]; \gamma_d, \beta_d), \quad \text{where} \quad (1a)$$

$$GBN = \gamma_d \cdot ((x_i[h, w, d] - \mu_d) / \sigma_d) + \beta_d \quad \forall \ i \in \{1..B\}, h \in \{1..H\}, w \in \{1..W\}, d \in \{1..D\} \quad (1b)$$

where $x_i[h, w, d]$ denotes the $[h, w, d]$ entry of the feature map $x_i$ and $\hat{x}_i[h, w, d]$ is the normalized output of the activation...
Crowd Counting Network: The objective of the crowd counting network is to generate the density map for the input image. For our AdaCrowd framework, we can use any backbone crowd counting network by inserting several GBN layers into the backbone.

The proposed GBN layer is related to conditional batch normalization (CBN) \[34]. However, the noticeable differences between them are: (1) CBN layer is specially devised to consider information from linguistic data, whereas GBN is designed to use information from spatial input data such as images. (2) CBN depends on the learned weights for $\gamma$ and $\beta$ obtained from BN layer in a pre-trained network for initialization. However, in GBN we directly predict $\gamma$ and $\beta$ from external visual data and consequently do not require a pre-trained network for GBN weight initialization. (3) Another technical difference is that, CBN learns to shift the pre-trained $\gamma$ and $\beta$ by small margin, whereas we completely replace the values for $\gamma$ and $\beta$ as each scene is visually different. Therefore, GBN is more intuitive and flexible when applying affine transformation on spatial data. We also visualize the operations in a GBN block in Fig. 3.

Crowd Counting Network: The objective of the crowd counting network is to generate the density map for the input image. For our AdaCrowd framework, we can use any backbone crowd counting network by inserting several GBN layers into the backbone.

$x_d[h, w, d]$. The mean $\mu_d$ and standard deviation $\sigma_d$ are calculated from the activation in channel $d$:

$$
\begin{align}
\mu_d &= \frac{\sum_{i=1}^{B} \sum_{h=1}^{H} \sum_{w=1}^{W} x_d[i, h, w, d]}{B \times H \times W} \quad (2a) \\
\sigma_d &= \sqrt{\frac{\sum_{i=1}^{B} \sum_{h=1}^{H} \sum_{w=1}^{W} \left( x_d[i, h, w, d] - \mu_d \right)^2}{B \times H \times W} + \epsilon} \quad (2b)
\end{align}
$$

where $\mu_d, \sigma_d \in \mathbb{R}$ and $\epsilon$ is used for numerical stability. In Eq. 1 the affine parameters $\gamma_d, \beta_d \in \mathbb{R}$ control the scaling and shifting operations corresponding to the $d$-th channel dimension. Let us use $\gamma$ and $\beta$ to denote the concatenations of $\{\gamma_d : \forall d\}$ and $\{\beta_d : \forall d\}$, respectively. The variables $\gamma, \beta \in \mathbb{R}^D$ are the parameters of this GBN layer and their values vary across different scenes. In general, for the $p$-th GBN layer, we can denote the affine parameters for the feature channel $d$ by $\gamma_p^d, \beta_p^d$.

The proposed GBN layer is related to conditional batch normalization (CBN) \[34\]. However, the noticeable differences between them are: (1) CBN layer is specially devised to consider information from linguistic data, whereas GBN is designed to use information from spatial input data such as images. (2) CBN depends on the learned weights for $\gamma$ and $\beta$ obtained from BN layer in a pre-trained network for initialization. However, in GBN we directly predict $\gamma$ and $\beta$ from external visual data and consequently do not require a pre-trained network for GBN weight initialization. (3) Another technical difference is that, CBN learns to shift the pre-trained $\gamma$ and $\beta$ by small margin, whereas we completely replace the values for $\gamma$ and $\beta$ as each scene is visually different. Therefore, GBN is more intuitive and flexible when applying affine transformation on spatial data. We also visualize the operations in a GBN block in Fig. 3.

To simplify the notation, we use $\phi$ to denote the concatenation of all parameters in GBN layers (i.e. $\gamma$ and $\beta$ from all GBN layers) in the crowd counting network. We use $\psi$ to denote other parameters in the crowd counting network. The crowd counting network can be written as a function $f$ parameterized by $\{\phi, \psi\}$ as:

$$
\hat{y} = f(x; \phi, \psi) \quad (3)
$$

where $f$ maps the input image $x$ to the predicted density map $\hat{y}$.

Note that $\psi$ will be the same for different scenes, while $\phi$ will change across scenes since $\phi$ is the output of the guiding network (described below). By predicting $\phi$ specifically to a scene, we achieve scene adaptive crowd counting.

Guiding Network: The goal of the guiding network is to predict the GBN parameters $\phi$ using one or more unlabeled images from a scene. For ease presentation, we assume that
we only one unlabeled image (denoted by \( z \in \mathbb{R}^{H \times W \times 3} \)) in the following. We will describe how to handle with case of multiple unlabeled images later.

Given the unlabeled image \( z \), we use it as the input to the guiding network to predict the GBN parameters \( \phi \) as:

\[
\phi = g(z; \theta) \tag{4}
\]

where \( g(\cdot; \theta) \) denotes a function parameterized by \( \theta \). In this paper, \( g(\cdot; \theta) \) is implemented as a CNN where the input \( z \) is an unlabeled image. But in general, \( g(\cdot; \theta) \) can be any arbitrary parametric function.

Our framework can be easily extended to the more general case where we have \( K \) (\( K > 1 \)) unlabeled images from the target scene. In this case, We can simply average \( \phi = \frac{1}{K} \sum_{i=1}^{K} \phi_i \) over \( K \) inputs.

**Learning and Inference:** Note that the counting network is parameterized by \( \{ \phi, \psi \} \). But the model parameters we need to learn are \( \{ \psi, \theta \} \), since \( \phi \) will be directly predicted from the guiding network (see Eq. 4). We would like to learn model parameters that have the following property. Suppose we have a new scene represented by \( z \) (unlabeled image). We would like the counting network with GBN parameters \( \phi \) obtained via Eq. 4 to perform well on images from this scene. To achieve this, we learn the model parameters from a set of labeled training images collected from multiple scenes. The model parameters are learned in a way that is amenable to effective adaptation to a new scene based on its scene-specific unlabeled data \( z \).

The learning algorithm works iteratively. In each iteration, the algorithm constructs a unlabeled scene adaptation task that mimics the scenario during testing. For a scene \( S^t \), we use \( \{(x_1, y_1), ..., (x_N, y_N)\} \) to denote the \( N \) training examples from this scene, where \( (x_i, y_i) \) correspond to the \( i \)-th (image, label) pair. We construct the task as follows. We randomly select one image to construct the scene-specific unlabeled data \( z \). Without loss of generality, let us assume that \( x_1 \) is selected to construct \( z \). We feed \( z \) to Eq. 4 to compute the GBN parameters \( \phi \) from the guiding network parameterized by \( \theta \). We then define a loss function that measures the “goodness” of the counting network with parameters \( \{ \phi, \psi \} \). Since our goal is for the learned model to perform well on other images from the same scene, a reasonable loss function is to measure the performance on the remaining \( N - 1 \) images from this scene. We can define the loss for this scene \( S^t \) as follows:

\[
\mathcal{L}(\{\psi, \theta\}; S^t) = \sum_{i=2}^{N} ||f(x_i; \{\phi, \psi\}) - y_i||^2 \tag{5a}
\]

\[
= \sum_{i=2}^{N} ||f(x_i; g(z; \theta), \psi)) - y_i||^2 \tag{5b}
\]

Here we use the \( L_2 \) distance to measure the difference between the predicted (i.e. \( f(x_i; \{g(z; \theta), \psi\}) \)) and the ground-truth (i.e. \( y_i \)) density maps.

The overall objective is to learn \( \{ \psi, \theta \} \) that minimize Eq. 5 across all scenes during training, i.e.:

\[
\min_{\{\psi, \theta\}} \sum_{t} \mathcal{L}(\{\psi, \theta\}; S^t) \tag{6}
\]

In Eq. 6 we sum over the loss across all scenes to optimize the parameters during training. In Algorithm 1 we provide an overview of the learning mechanism.

**Algorithm 1:** Training for unlabeled scene adaptive crowd counting

**Input:** Training images from multiple scenes \( \{ S^t \} \)

**Initialize:** the model parameters \( \{ \psi, \theta \} \);

**while not done do**

**for each scene \( S^t \) do**

Sample one image to obtain \( z \);

Compute GBN parameters \( \phi \) using Eq. 4;

Evaluate \( \nabla_{\{\psi, \theta\}} \mathcal{L}(\{\psi, \theta\}; S^t) \) in Eq. 5;

end

Update \( \{ \psi, \theta \} \leftarrow \sum_t \nabla_{\{\psi, \theta\}} \mathcal{L}(\{\psi, \theta\}; S^t) \);

end

Let \( \{ \psi^*, \theta^* \} \) be the learned model parameters that minimize Eq. 6. During testing, we are given a new scene represented as \( z_{new} \) (unlabeled image). For any crowd image \( x \) from this scene, we can predict its label \( \hat{y} \) as:

\[
\hat{y} = f(x; \{ \phi^*, \psi^* \}), \quad \text{where} \quad \phi^* = g(z_{new}; \theta^*) \tag{7}
\]

The idea of using adjustable GBN parameters for adaptation is inspired by the work in image translation [45]. Similar to [45], the affine transformation in the normalization layers is spatially invariant, so it can only obtain global appearance information. In the context of crowd counting, different scenes are often characterized by some factors (e.g. camera angle, height) that cause changes in the global scene appearance. By adapting parameters in the GBN layer (which are spatially invariant), the parameter \( \phi \) obtained via the guiding network from the unlabeled images intuitively captures the global information (e.g. scene geometry) about the target scene. The local structures needed in crowd counting are implicitly captured by the parameter \( \psi \) and are shared across different scenes.

**V. Experiments**

In this section, we first describe the datasets and the experiment setup in Sec. V-A. We then introduce several baseline methods used for comparisons in Sec. V-B. We present experimental results in Sec. V-C.

**A. Datasets and Setup**

**Datasets:** We experiment with the following datasets.

- **WorldExpo’10 [10]:** The WorldExpo’10 dataset consists of 3980 labeled images covering 108 different surveillance camera scenes. The dataset is split into training set (103 scenes and 3380 images) and testing set (5 scenes and 600 images). Note that although the dataset consists of ROI maps, we do not use them during the training of our AdaCrowd model. In our experiments, we resize the images to 512 × 672.

- **Mall [46]:** The Mall dataset consists of 2000 frames captured from a surveillance camera in a mall. The frame
size is 640 × 480. The training and test sets consist of 800 and 1200 images, respectively.

- **PETS [47]**: The PETS dataset is a multi-view dataset of crowd scene from 8 views. We use the first 3 views following [48] and similarly we consider sequences S1L3 (14_17, 14_33), S2L2 (14_55) and S2L3 (14_41) as the training set consisting of 1105 images. For test set, we consider S1L1 (13_57, 13_59), S1L2 (14_06, 14_31) consisting of 794 images. We treat each view as a scene in our experiments and hence we consider 3 scenes in [47].

- **FDST [49]**: The FDST dataset is made up of 13 different camera scenes. The training set consists of 60 videos resulting in 9000 frames and the testing set consists of 40 videos resulting in 6000 frames. The dataset contains images of two different resolutions (1920 × 1080 & 1280 × 720). Following [49], we resize all the frames to 640 × 360 in our experiments.

- **CityUHK-X [29]** The CityUHK-X dataset consists of a total of 55 camera scenes. The training set comprises of 43 scenes and the testing set comprises of the 12 scenes. The frame size is 384 × 512.

- **Venice [50]** The Venice dataset comprises of 4 different crowd sequences captured from a mobile camera and in total the dataset contains 167 labeled frames. The training set considers 80 images from a single sequence, while the test set contains the other three video sequences and we treat each test video as a different scene for scene adaptation.

Our problem setup requires data from multiple scenes during the training phase. To the best of our knowledge, WorldExpo’10 [10] is the only real-world crowd counting dataset with large number of scenes that can be used for training in our problem setup. Therefore, we use WorldExpo’10 for training and use other datasets to test for scene adaptation.

**Implementation**: We implement AdaCrowd framework with different backbone (VGG [52] or ResNet [51]) networks. The learning rate for all the experiments is set to 1e-5. We use Adam [53] optimizer with gradient clipping norm of 1 and we train all the models for 110 epochs.

**Evaluation Metrics**: We use two standard metrics following previous works in crowd counting [2, 10, 13] to evaluate the performance of our model, namely mean absolute error (MAE) and root mean square error (RMSE) as shown below:

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |\omega_i^y - \omega_i^\hat{y}| \tag{8}
\]

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |\omega_i^y - \omega_i^\hat{y}|^2} \tag{9}
\]

where \(N\) represents the total camera scene images, we denote the ground-truth and the predicted density map for a given \(i\)-th input image as \(y_i\) and \(\hat{y}_i\), respectively. Additionally, the crowd count at a particular spatial location \((h, w)\) is given by \(p_{h,w}\) and the total crowd count in a density map with spatial resolution \(H \times W\) can be computed by \(\omega_i = \sum_{h=1}^{H} \sum_{w=1}^{W} p_{h,w}\). For MAE and RMSE, lower values indicates better performance.

**B. Baselines and Backbone Architectures**

We consider the following state-of-the-art crowd counting networks as baselines for comparison. These baselines train the networks in the standard supervised manner on images available during training, then directly apply the trained networks to test different scenes without adaptation.

- **CSRNet [2]**: The first sub-network in CSRNet is based on VGG [52] and consists of layers till \textit{conv}_4.3 to extract the features with dimension \(H/8 \times W/8 \times 512\) from the image input. The extracted features are used to construct the density map using a series of dilated convolutional layers in the second sub-network.

- **CSRNet w/ Batch Normalization [2]**: This baseline is based on CSRNet. We add batch normalization layers between every \textit{conv} and \textit{relu} layer in the second sub-network to generate the density map.

- **ResNet FCN [8]**: This baseline is based on the ResNet101 [51] architecture to extract the image features with dimension \(H/8 \times W/8 \times 1024\). Similar to CSRNet, we first extract the image features and then generate the density map using the second sub-network comprising of dilated convolutional layers.

- **ResNet FCN w/ Batch Normalization [8]**: This baseline is similar to ResNet FCN. We add batch normalization layers to the density map generator sub-network like in CSRNet w/ BN.

- **ResNet SFCN [8]**: This baseline uses ResNet101 [51] for the feature extractor part of the network. The density map generator consists of dilated convolutional layers and spatial fully connected layers as proposed in [8].

- **ResNet SFCN w/ Batch Normalization [8]**: This baseline is based on ResNet SFCN. We add batch normalization layers in the density map generator like in other BN based baselines.

We use CSRNet, ResNet FCN and ResNet SFCN (with GBN layers inserted) as the backbone architecture in our AdaCrowd framework. To be specific, we can derive an AdaCrowd variant for each of the BN-based baseline methods by replacing all BN layers with GBN layers. In our approach, we generate the parameters for the GBN layers using the guiding network from the unlabeled data \(z\).

**C. Experimental Results**

**Quantitative Results**: In Table I, we show the average results of training with 103 scenes and testing on 5 different scenes on WorldExpo’10. In Table II, we show the results of training on WorldExpo’10 and testing on other datasets (Mall, PETS, FDST, and CityUHK-X). As other datasets (Mall, PETS, FDST) have same scenes in both training and testing, we use unlabeled image from the train set when sampling \(z\) and
TABLE I
Quantitative results for training and testing on WorldExpo’10. We report results using different backbone architectures. “Ours” uses one unlabeled image z. We show the results on CSRNet [2], FCN [51], and SFCN [8]. The results for our approach show the mean and standard deviation (%) over 5 random trials. We show the best results in bold.

| Backbone | Method       | Adaptive | MAE (↓) | RMSE (↓) |
|----------|--------------|----------|---------|----------|
| VGG-16   | CSRNet       | -        | 19.56   | 28.34    |
|          | CSRNet w/ BN | -        | 18.57   | 29.91    |
|          | Ours w/ CSRNet | ✓        | 17.32 ± 0.1 | 27.03 ± 0.05 |
| ResNet-101 | FCN       | -        | 23.07   | 34.16    |
|          | FCN w/ BN   | -        | 21.65   | 33.3     |
|          | Ours w/ FCN | ✓        | 20.91 ± 0.3 | 29.61 ± 0.25   |
| ResNet-101 | SFCN     | -        | 25.47   | 35.91    |
|          | SFCN w/ BN  | -        | 23.84   | 35.52    |
|          | Ours w/ SFCN | ✓        | 14.56 ± 0.4 | 22.75 ± 0.2   |

TABLE II
Quantitative results for the cross-dataset testing for one unlabeled image. We train WorldExpo’10 and test on Mall, PETS, FDST, and CityUHK-X. We show results of different backbone networks and report mean and standard deviation (%) of our models over 5 random trials. We show the best results in bold.

| Method       | Adapt. | WorldExpo → Mall | WorldExpo → PETS | WorldExpo → FDST | WorldExpo → CityUHK-X |
|--------------|--------|------------------|------------------|------------------|----------------------|
|              |        | MAE (↓) | RMSE (↓) | MAE (↓) | RMSE (↓) | MAE (↓) | RMSE (↓) | MAE (↓) | RMSE (↓) |
| CSRNet [2]   | -      | 9.94   | 10.41   | 17.99 | 19.80   | 12.74 | 13.09   | 39.85 | 48.07   |
| CSRNet w/ BN | -      | 8.72   | 9.92    | 18.63 | 20.49   | 7.30  | 7.81    | 22.69 | 32.33   |
| Ours w/ CSRNet | ✓      | 4.0 ± 0.08 | 5.0 ± 0.09 | 17.43 ± 0.04 | 19.70 ± 0.05 | 7.14 ± 0.4 | 7.77 ± 0.2 | 20.38 ± 0.004 | 29.06 ± 0.003 |
| FCN [51]     | -      | 9.03   | 9.6     | 20.38 | 22.67   | 7.14  | 7.85    | 27.56 | 37.73   |
| FCN w/ BN    | -      | 9.63   | 10.32   | 19.59 | 21.61   | 8.89  | 9.23    | 27.3  | 37.56   |
| Ours w/ FCN  | ✓      | 4.12 ± 0.2 | 5.12 ± 0.2 | 13.74 ± 0.1 | 16.15 ± 0.09 | 6.13 ± 0.4 | 6.69 ± 0.3 | 21.77 ± 0.012 | 30.94 ± 0.0124 |
| SFCN [8]     | -      | 15.17  | 15.53   | 19.62 | 21.35   | 8.72  | 8.94    | 27.34 | 35.78   |
| SFCN w/ BN   | -      | 10.8   | 11.39   | 19.94 | 22.26   | 6.60  | 6.96    | 26.15 | 36.61   |
| Ours w/ SFCN | ✓      | 6.99 ± 1.2 | 8.0 ± 1.0 | 18.41 ± 0.2 | 20.63 ± 0.1 | 5.76 ± 0.3 | 6.57 ± 0.3 | 20.14 ± 0.006 | 28.27 ± 0.013 |

Fig. 4. Qualitative results of our approach on different datasets. We visualize the density maps and show both ground-truth and predicted count at the top-left corner of each image.
TABLE III
QUANTITATIVE RESULTS FOR ONE-UNLABELED IMAGE BASED CROSS-DATASET TESTING. WE TRAIN THE ALL THE MODELS ON WORLDEXPO’10 AND TEST ON VENICE DATASET. WE PRESENT THE RESULT FOR SFCN BASED NETWORK BY REPORTING THE MEAN AND STANDARD DEVIATION (%) FOR OUR PROPOSED APPROACH. WE SHOW THE BEST RESULTS IN BOLD.

| Method   | WorldExpo → Venice (1 input) | MAE (↓) | RMSE (↓) |
|----------|------------------------------|---------|----------|
| SFCN     | 147.48                       | 157.48  |
| SFCN w/ BN | 147.75                       | 157.04  |
| Ours w/ SFCN | 133.83 ±0.15                | 146.44 ±0.15 |

evaluate the performance on all images from the corresponding test set. We perform 5 trails for each of our models on every dataset with different data on unlabeled image z. In case of CityUHK-X dataset, the test scenes are different from train scenes. Therefore, we randomly select images from the test for z and evaluate on the remaining images from the test scenes and similarly repeat this procedure 5 times. We finally report the mean score with the standard deviation (%) in Table II and III. In all the cases, our AdaCrowd significantly outperforms other baselines.

In Table III, we present the results of cross-device and cross-dataset setting for unlabeled scene adaptation, we train the models on WorldExpo’10 and evaluate for scene adaptation on Venice [50] dataset. Similar to other quantitative results, we report the mean and standard deviation for our proposed approach on SFCN based network over 5 random trails. Our proposed method on SFCN performs better than the baselines on cross-device scene adaptation task.

Qualitative Results: In Fig. 4 we present qualitative examples from different datasets. We visualize the predicted density map and the ground-truth count.

Fig. 5. Performance vs. number of training scenes – we present the ablation study on the relation between network performance and number of training scenes on WorldExpo’10 dataset with CSRNet and ResNet SFCN networks.

Ablation Analysis: We perform additional analysis on the proposed AdaCrowd framework to gain further understanding of the proposed approach. In this analysis, we increase the number of unlabeled images to 5. We show the results of training and testing on WorldExpo’10 in Table IV. In general, increasing the number of unlabeled data slightly gives better results. One possible explanation is that using more unlabeled data can make the algorithm more robust to noise.

In Fig. 5, we present the analysis on how the performance varies when we vary the number of training scenes. This analysis uses the CSRNet and ResNet SFCN architectures on the WorldExpo’10 dataset. From the plot, the performance (low error for MAE and RMSE) is positively correlated with the number of training scenes. Therefore, AdaCrowd framework performs better at test time if more scenes are available during training.

In Fig. 6, we study the performance of the SFCN based network by varying the number of GBN layers in the second sub-network (decoder) of the framework that generates the output density map. Interestingly, increasing the GBN layers to 6 yields the best performance on WorldExpo’10 dataset and this is also equivalent to a GBN layer for every dilated convolutional layers in the second sub-network.

In Fig. 6, we present the analysis on how the performance varies when we vary the number of GBN layers on the network’s performance when trained on WorldExpo’10 dataset with ResNet SFCN architecture.

In Fig. 7, we show some failure cases due to factors such as illumination, occlusion or image quality that have drastic effect in the target scene images. However, our approach still performs better than other methods in these cases. As future work, we will explore to incorporate other meta-information (e.g. illumination estimation) into our framework to deal with
TABLE IV
Comparison of our approach with 1 vs. 5 inputs (unlabeled image) by training and testing on WorldExpo’10. In all the cases, the results of using 5 inputs are slightly better than using 1 input. We report the mean and standard deviation (%) for all the methods.

| Method          | 1 input MAE (±) | 5 inputs MAE (±) | 1 input RMSE (±) | 5 inputs RMSE (±) |
|-----------------|-----------------|------------------|------------------|------------------|
| Ours w/ CSRNet  | 17.32 ± 0.1     | 17.21 ± 0.6      | 27.03 ± 0.05     | 26.85 ± 0.02     |
| Ours w/ FCN     | 20.91 ± 0.3     | 20.88 ± 0.2      | 29.61 ± 0.2      | 29.53 ± 0.4      |
| Ours w/ SFCN    | 14.56 ± 0.4     | 14.47 ± 0.4      | 22.75 ± 0.2      | 22.61 ± 0.4      |

these challenging cases.

VI. CONCLUSION

We have introduced a new problem called the unlabeled scene adaptive crowd counting. Our goal is to adapt a crowd counting model to a target scene using some unlabeled data from that scene. In contrast to the existing problem setups, this new problem setup is closer to the real-world deployment of crowd counting systems. We have proposed a novel framework AdaCrowd with GBN layers for solving this problem. Our proposed approach employs a guiding network to predict GBN parameters of the crowd counting network based on the unlabeled data of a scene. The model parameters are learned in a way that allows effective adaptation to new scenes given their unlabeled data. Our experimental results demonstrate that our proposed approach outperforms other alternative methods.

REFERENCES

[1] D. Kang and A. Chan, “Crowd counting by adaptively fusing predictions from an image pyramid,” in British Machine Vision Conference (BMVC), 2018.
[2] Y. Li, X. Zhang, and D. Chen, “Csnet: Dilated convolutional neural networks for understanding the highly congested scenes,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[3] D. B. Sam, N. N. Sajan, H. Maurya, and R. V. Babu, “Almost unsupervised learning for dense crowd counting,” in AAAI Conference on Artificial Intelligence (AAAI), 2019.
[4] D. B. Sam, S. Surya, and R. V. Babu, “Switching convolutional neural network for crowd counting,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[5] V. A. Sindagi and V. M. Patel, “Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting,” in IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS), IEEE, 2017.
[6] ——. “Generating high-quality crowd density maps using contextual pyramid CNNs,” in IEEE International Conference on Computer Vision (ICCV), 2017.
[7] E. Walach and L. Wolf, “Learning to count with cnn boosting,” in The European Conference on Computer Vision (ECCV). Springer, 2016.
[8] Q. Wang, J. Gao, W. Lin, and Y. Yuan, “Learning from synthetic data for crowd counting in the wild,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[9] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in International Conference on Learning Representations (ICLR), 2016.
[10] C. Zhang, H. Li, X. Wang, and X. Yang, “Cross-scene crowd counting via deep convolutional neural networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[11] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, “Single-image crowd counting via multi-column convolutional neural network,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[12] M. A. Hossain, M. K. Krishna Reddy, M. Hosseinzadeh, O. Chanda, and Y. Wang, “One-shot scene-specific crowd counting,” in British Machine Vision Conference (BMVC), 2019.
[13] M. K. K. Reddy, H. Hossain, M. Rochan, and Y. Wang, “Few-shot scene adaptive crowd counting using meta-learning,” in IEEE Winter Conference on Applications of Computer Vision (WACV), 2020.
[14] N. Paragiou and V. Ramesh, “A mrf-based approach for real-time subway monitoring,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2001.
[15] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, “Privacy preserving crowd monitoring: Counting people without people models or tracking,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.
[16] K. Chen, C. C. L. Loy, S. Gong, and T. Xiang, “Feature mining for localised crowd counting,” in British Machine Vision Conference (BMVC), 2012.
[17] A. Marana, L. d. F. Costa, R. Lotufo, and S. Velastin, “On the efficacy of texture analysis for crowd monitoring,” in SIBGRAPI. IEEE, 1998.
[18] P. Chattopadhyay, R. Vedantam, R. R. Selvaraju, D. Batra, and D. Parikh, “Counting everyday objects in everyday scenes,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[19] C. Shang, H. Ai, and B. Bai, “End-to-end crowd counting via joint learning local and global count,” in IEEE International Conference on Image Processing (ICIP), 2016.
[20] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao, “Deep people counting in extremely dense crowds,” in ACM International Conference on Multimedia. ACM, 2015.
[21] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” in Advances in Neural Information Processing Systems (NeurIPS), 2010.
[22] ——. “Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation,” in IEEE International Conference on Computer Vision (ICCV), 2015.
[23] Z.-Q. Cheng, J.-X. Li, Q. Dai, X. Wu, and A. G. Hauptmann, “Learning spatial awareness to improve crowd counting,” in IEEE International Conference on Computer Vision (ICCV), 2019.
[24] Z. Ma, X. Wei, X. Hong, and Y. Gong, “Bayesian loss for crowd count estimation with point supervision,” in IEEE International Conference on Computer Vision (ICCV), 2019.
[25] J. Wan and A. Chan, “Adaptive mixture map generation for crowd counting,” in IEEE International Conference on Computer Vision (ICCV), 2019.
[26] X. Liu, J. Yang, and W. Ding, “Adaptive mixture regression network with local counting map for crowd counting,” in The European Conference on Computer Vision (ECCV), 2020.
[27] X. Jiang, L. Zhang, T. Zhang, P. Lv, B. Zhou, Y. Pang, M. Xu, and C. Xu, “Density-aware multi-task learning for crowd counting,” IEEE Transactions on Multimedia (TMM), 2020.
[28] L. Liu, W. Jia, J. Jiang, S. Amirgholipour, Y. Wang, M. Zeibots, and X. He, “Denet: A universal network for counting crowd with varying densities and scales,” IEEE Transactions on Multimedia (TMM), 2020.
[29] D. Kang, D. Dhar, and A. Chan, “Incorporating side information by adaptive convolution,” in Advances in Neural Information Processing Systems (NeurIPS), 2017.
[30] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2006.
[31] R. Salakhutdinov, J. Tenenbaum, and A. Torralba, “One-shot learning with a hierarchical nonparametric bayesian model,” in ICML Workshop on Unsupervised and Transfer Learning, 2012.
[32] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in International Conference on Machine Learning (ICML), 2017.
[33] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in International Conference on Machine Learning (ICML), 2015.
[34] A. Nichol, J. Achiam, and J. Schulman, “On first-order meta-learning algorithms,” arXiv preprint arXiv:1803.02999, 2018.
[35] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in Advances in Neural Information Processing Systems (NeurIPS), 2017.

[36] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[37] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., “Matching networks for one shot learning,” in Advances in Neural Information Processing Systems (NeurIPS), 2016.

[38] T. Munkhdalai and H. Yu, “Meta networks,” in International Conference on Machine Learning (ICML), 2017.

[39] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in International Conference on Machine Learning (ICML), 2016.

[40] S. Ravi and H. Larochelle, “Optimization as a model for few-shot learning,” in International Conference on Learning Representations (ICLR), 2017.

[41] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in International Conference on Machine Learning (ICML), 2015.

[42] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Deep transfer learning with joint adaptation networks,” in International Conference on Machine Learning (ICML), 2017.

[43] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine Learning (ICML), 2015.

[44] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville, “Modulating early visual processing by language,” in Advances in Neural Information Processing Systems (NeurIPS), 2017.

[45] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, “Few-shot unsupervised image-to-image translation,” in IEEE International Conference on Computer Vision (ICCV), 2019.

[46] C. Change Loy, S. Gong, and T. Xiang, “From semi-supervised to transfer counting of crowds,” in IEEE International Conference on Computer Vision (ICCV), 2013.

[47] J. Ferryman and A. Shahrokni, “Pets2009: Dataset and challenge,” in IEEE International workshop on performance evaluation of tracking and surveillance, 2009.

[48] Q. Zhang and A. B. Chan, “Wide-area crowd counting via ground-plane density maps and multi-view fusion cnns,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[49] Y. Fang, B. Zhan, W. Cai, S. Gao, and B. Hu, “Locality-constrained spatial transformer network for video crowd counting,” in IEEE International Conference on Multimedia and Expo (ICME), 2019.

[50] W. Liu, M. Salzmann, and P. Fua, “Context-aware crowd counting,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[52] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2014.

[53] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in International Conference on Learning Representations (ICLR), 2015.