Scene Adaptive Crowd Counting

by

Mahesh Kumar Krishna Reddy

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
April 2020

Thesis advisor

Dr. Yang Wang

Author

Mahesh Kumar Krishna Reddy

Scene Adaptive Crowd Counting

Abstract

We consider the problem of scene adaptive crowd counting. Given a target camera scene, our goal is to adapt a model to this specific scene with only a few labeled/unlabeled images. The solution to this problem has potential applications in numerous real-world scenarios that require deploying a crowd counting model specially adapted to a target camera. In this thesis, we propose two novel methods for scene adaptive crowd counting. First, inspired by the recently introduced learning to learn paradigm in the context of few-shot regime, we aim to learn the parameters of a crowd counting model in a way to facilitate fast adaptation to the target scene. Second, we introduce a new problem called *unlabeled scene adaptive crowd counting*. More specifically, we propose to use few unlabeled images from the target scene to perform the adaptation. We introduce a novel `AdaCrowd` framework to solve this problem and it 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 two proposed approaches.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I wish to convey my appreciation to my advisor Dr. Yang Wang for the unrelenting encouragement and mentoring. Also, it is an honor to have Dr. Carson Kai-Sang Leung and Dr. Ekram Hossain on my thesis committee and for their invaluable suggestions in perfecting my thesis.

I am truly grateful to Dr. Yang Wang for the financial assistance and continuous motivation throughout my journey which was vital to the completion of my thesis. I would like to extend my thanks to the support staff in the Department of Computer Science for their help.

I wish to express my gratitude to all the amazing lab mates for the lively discussions, constructive feedback, and collaborations.

Last but not the least, I am truly in debt to my family for providing me the opportunity to pursue my field of interest and for being the pillars of strength.

This thesis is dedicated to my parents and my sister
for their unconditional love and endless support.

Publications

Some of the ideas, materials and figures in this thesis have appeared previously in the following publications and submitted manuscripts:

1. **Mahesh Kumar Krishna Reddy**, Mohammad Hossain, Mrigank Rochan and Yang Wang. Few-shot Scene Adaptive Crowd Counting using Meta-Learning. *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass Village, USA, March 2020.

2. **Mahesh Kumar Krishna Reddy**, Mrigank Rochan, Yiwei Lu and Yang Wang. AdaCrowd: Unlabeled Scene Adaptive Crowd Counting. *The European Conference on Computer Vision (ECCV)*, Glasgow, Scotland, 2020. (Under review)

Chapter 1

Introduction

Crowd counting is a problem of determining the number of people present in a surveillance camera image. The research on crowd counting [3, 4, 9, 10, 11] is drawing increasing attention in computer vision lately. The key reason for this surge in interest is the demand for automated complex crowd scene understanding that appears in computer vision applications such as surveillance, traffic monitoring, etc. Although the contemporary methods for crowd counting are promising, they have some significant limitations. One main limitation of existing methods is the difficulty in adapting to a new crowd scene. This is because these methods typically require a large number of labeled training data which is expensive and time-consuming to obtain.

In this thesis, we analyze the above-mentioned limitation of existing crowd counting methods in two scene adaptive crowd counting problems. First, we consider the few-shot scene adaptive crowd counting similar to [1]. During training, we have access to a set of training images from different scenes (e.g., each scene might correspond to

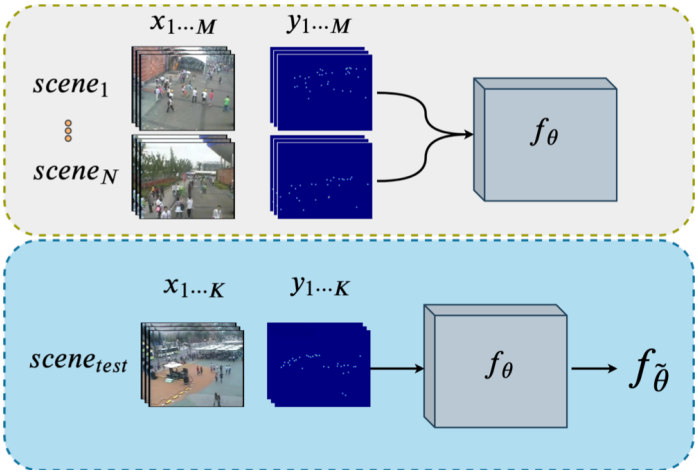


Figure 1.1: Illustration of our problem setting. (Top row) During training, we have access to a set of N different camera scenes where each scene comes with M labeled examples. From such training data, we learn the model parameters θ of a mapping function f_θ such that θ is generalizable across scenes in estimating the crowd count. (Bottom row) Given a test (or target) scene, we assume that we have a small number of K labeled images from this scene, where $K \ll M$ (e.g., $K \in \{1, 5\}$) to learn the scene-specific parameters $\tilde{\theta}$. With the help of meta-learning guided approach we quickly adapt f_θ to $f_{\tilde{\theta}}$ that predicts more accurate crowd count than other alternative solutions.

one specific camera installed at one particular location). During testing, we have a new target crowd scene to which we want to adapt our model. Moreover, we consider that we have a small number (e.g., 1 or 5) of labeled images from this target scene. During training, we learn optimal (generalizable) model parameters from multiple scene-specific data by considering few-labeled images per scene. During testing, we consider the learned parameters to be a good initial point to adapt to a specific new scene. To be precise, we aim at learning the generalizable model parameters in a fashion that it produces more accurate performance when adapting to a new target scene with few gradient descent steps provided only a few labeled images from the target scene. Figure 1.1 shows an illustration of the few-shot scene adaptive problem.

We address the proposed few-shot crowd counting problem using meta-learning [12] that is capable of fast adaptation to new camera scenes.

Second, in comparison with the standard supervised setting, the few-shot scene adaptation setup in [1, 2] 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 [1, 2] 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 [1, 2], we push the envelope even further by proposing a new problem called *Unlabeled Scene Adaptive Crowd Counting*. Similar to [1, 2], our objective is to have a model adapted to a specific target scene during deployment. But different from [1, 2], 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. 1.2). Since unlabeled images are fairly easy to collect in practice, this problem setup greatly reduces the data annotation effort from the user. Besides, our proposed approach also significantly reduces the required computation during adaptation. It only involves some feed-forward 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 parame-

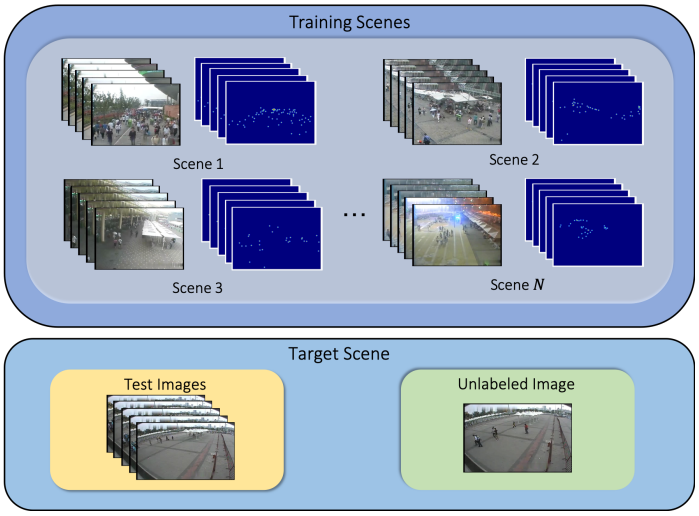


Figure 1.2: Illustration of the unlabeled scene-adaptive crowd counting problem. (Top row) Our training data consists of images and labels collected from N different scenes. (Botton row) During testing, we are given some unlabeled image (z) from a new target scene. Our goal is to produce a crowd counting model specifically adapted to this target scene to predict crowd count on the given test images. Unlike the few-shot adaptation in [1, 2], our problem setup does not require any labeled images from the target scene for adaptation.

ters 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 using only unlabeled images from that scene.

1.1 Contributions

The contributions presented in this thesis are:

- We propose a meta-learning inspired approach to solve the few-shot scene adaptive crowd counting problem. Using the meta-learning, the model parameters are learned in a way that facilitates effective fine-tuning to a new scene with

a few labeled images. Previous work [1] uses a fine-tuning approach to this problem. The limitation of this fine-tuning approach is that it can only update certain layers that are closer to the output in the decoder to a target scene. In contrast, our approach does not have such limitations and can be used to adapt any parameters in the decoder. Second, we perform a thorough evaluation of the performance of our proposed approach on several benchmark datasets and show that the method outperforms other alternative baselines. Our approach also surpasses the fine-tuning approach [1].

- We present a new problem called unlabeled scene adaptive crowd counting. Different from the few-shot adaptation setup in [1, 2], our problem formulation only uses unlabeled images from the target scene for adaptation. 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.

1.2 Thesis Organization

We organize the remainder of this thesis as follows. First, in Chapter 2, we give an overview of the related works. More specifically, in this chapter, we discuss various crowd counting, domain adaptation, and few-shot learning approaches. In Chapter 3, we outline our proposed meta-learning inspired labeled scene adaptive crowd counting approach. We show that using different camera scenes at training time is beneficial for learning generalizable model parameters, which are then helpful to adapt to a specific

scene at test time. In Chapter 4, we propose an unlabeled scene adaptive approach for crowd counting. Unlike the previous supervised approaches, we overcome the constraints such as data annotations, and parameter update at test time. Finally, in Chapter 5, we conclude this thesis.

Chapter 2

Related Work

In this chapter, we provide a brief overview of the works that are closely related to our proposed approaches.

2.1 Crowd Counting

In the context of crowd counting for scene adaptation, we propose the following two approaches: i) labeled scene adaptation to learn scene-specific model parameters with few images, and ii) exploiting unlabeled images to extract scene-specific feature representation to perform unlabeled scene adaptation. In the following subsections, we briefly outline the work relevant to our proposed approaches.

2.1.1 Labeled Scene Adapative Crowd Counting

The research in crowd counting can be grouped into either detection [13, 14], regression [15, 16] or density-based [17, 18] methods as proposed by [19]. Earlier work

focuses on the detection and regression-based approaches. In recent years, density-based approaches using deep learning models have become popular and show superior performance. Zhang *et al.* [4] propose an approach with two learning objectives for density estimation and crowd counting. Additionally, they propose a non-parametric method to fine-tune the model to minimize the distribution difference between the source and target scenes. Zhang *et al.* [11] addresses crowd counting by proposing a multi-column neural network to handle an input image at multiple scales to overcome the problem of scale variations. Sam et *al.* [9] propose to estimate the density of an image patch from a regressor selected based on the density level classifier. Sindagi and Patel [10] propose to encode both local and global input image contexts to estimate the density map. In our work on labeled scene adaptation, we employ the state-of-the-art CSRNet [3] architecture to learn crowd counting. We propose a principled approach to infuse scene adaptation capability to this network for adaptation with only a few labeled training images from a target scene. In the context of crowd counting for scene adaptation, we propose the following two approaches: i) labeled scene adaptation to learn scene scene-specific model parameters with few images, and ii) exploiting unlabeled images to extract scene-specific feature representation to perform unlabeled scene adaptation. In the following subsections, we briefly outline the work relevant to our proposed approaches.

2.1.2 Unlabeled Scene Adaptive Crowd Counting

Although labeled scene adaptive crowd counting gets us closer to the real-world deployment compared to the standard supervised setting, this setup still has some

limitations that restrict it from wide usage in the real-world. The limitations of labeled scene-adaptive crowd counting are as follows: i) requirement of at least one labeled image from the novel scene, and ii) it involves fine-tuning of several layers in the CNN model, which involves backpropagation to update the network parameters.

Inspired by the above observations, we propose an approach for scene adaptation by exploiting unlabeled images. As the unlabeled images are relatively easier to collect in the real-world, this relieves the user from expensive data annotation. Additionally, our proposed approach does not involve gradient updates while adapting to a target camera scene. In our work, we experiment with different architectures such as CSRNet [3], ResNet FCN [20], and ResNet SFCN [20].

2.2 Domain Adaptation

In the context of crowd counting adaptation, Loy *et al.* [6] proposes a non-CNN semi-supervised adaptation method by exploiting unlabeled data in the target domain. The drawback of this approach is that it requires corresponding samples that have common labels between the source and target domains. This information is usually not available in recent crowd counting datasets. Wang *et a*l. [20] propose to generate a large synthetic dataset and perform domain adaptation to the real-world target domain. One drawback of this method is that it requires prior knowledge about the distribution of the target domain to manually select the scenes in the synthetic dataset. Hossain *et al.* [1] proposes a one-shot adaptation approach based on fine-tuning few layers in the decoder network for adapting a crowd counting model to a specific scene. Conversely, our labeled scene adaptive approach neithers requires

manual selection of data or layers in the network to perform scene adaptation on a target camera scene.

Kang *et al.* [21] 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.

2.3 Few-Shot Learning

The goal of few-shot learning is to learn a model from limited training examples for a task. Previously, Li *et al.* [22] propose a method for unsupervised one-shot learning by casting the problem in a probabilistic setting. Lake *et al.* [23] use compositionality and causality for one-shot scenario through Hierarchical Bayesian learning system. Luo *et al.* [24] demonstrate the transferability of representations across domains with few labeled data. A different perspective to tackle few-shot learning is by treating it is as a meta-learning problem (also known as *learning to learn* [25, 26]). The essence of using meta-learning for few-shot learning problem involves a neural network as a learner to learn about a new task with just a few instances.

The recent work in meta-learning can be grouped into metric-based [27, 28, 29, 30], model-based [31, 32] or optimization-based [7, 8, 12]. The metric-based [27, 28, 29, 30] methods in general learn a distance function to measure the similarity between data points belonging to the same class. Memory or model-based [31, 32]

approaches employ a memory component to store previously used training examples. The optimization-based [7, 8, 12] frameworks learn good initialization parameters based on learning from multiple tasks that favour fast adaptation on a new task. The above works primarily target image recognition challenge, in our proposed labeled scene adaptive crowd counting we follow the optimization-based meta-learning mechanism similar to [12] for a more challenging problem of crowd density estimation as it has shown to achieve superior performance compared to other optimization based methods.

Chapter 3

Few-Shot Scene Adaptive Crowd Counting using Meta-Learning

In this chapter, we first describe the problem setup for few-shot scene adaptive crowd counting (Sec. 3.1). We then introduce our proposed approach for scene adaptive crowd counting using meta-learning (Sec. 3.2), and finally we provide an overview of the experimental setup (Sec. 3.3).

3.1 Problem Setup

We formulate our scene adaptive crowd counting as a few-shot learning problem using meta-learning. In a traditional supervised machine learning setting, we are given a dataset $\mathcal{D} = \{D^{train}, D^{test}\}$, where D^{train} and D^{test} are the training and test sets, respectively. The goal is to learn a mapping function $f_\theta : x \rightarrow y$ that maps an input x (e.g. an input image) to its corresponding label y (e.g. the crowd density

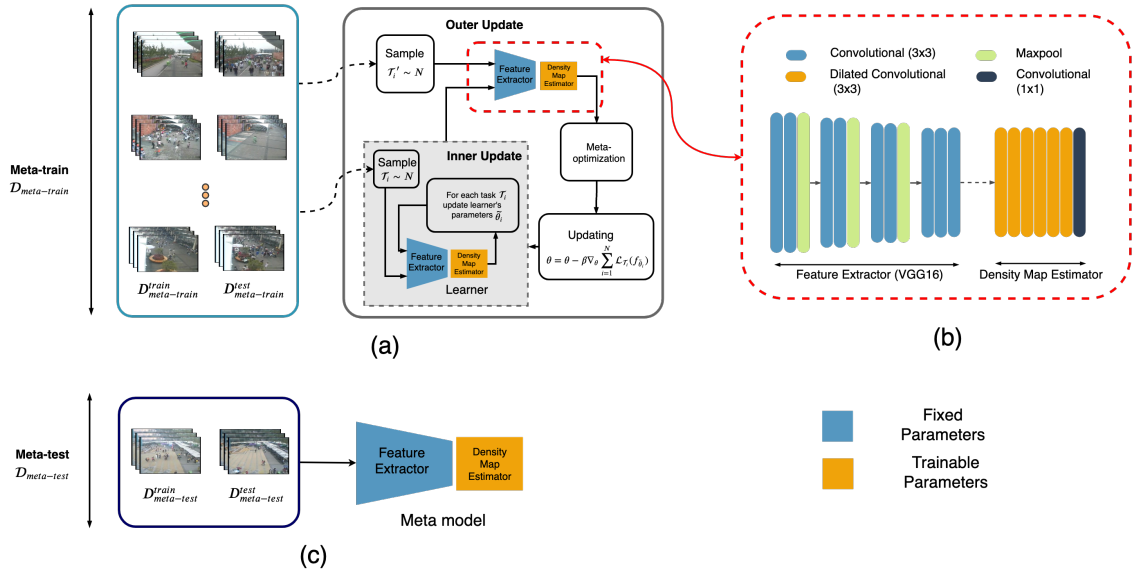


Figure 3.1: An overview of the main components of our model. (a) *Meta-training* stage on $\mathcal{D}_{meta-train}$. The meta-training involves optimizing an inner-update over each scene and an outer-update across different scenes. (b) Backbone crowd counting network. We use the CSRNet [3] as the backbone architecture. It comprises of a feature extractor and a density map estimator. (c) *Meta-testing* on $\mathcal{D}_{meta-test}$. We adapt the trained meta-model with θ to a new target scene by fine-tuning on K images from this scene and test on other images from this scene.

map). We use θ to denote the parameters of the mapping function f_θ. We learn θ by optimizing its corresponding loss function defined on D^{train}. After training, we test the generalization of the learned model f_θ on D^{test}.

In contrast, a few-shot meta-learning model is trained on a set of N tasks during meta-learning (*meta-training*) from $\mathcal{D}_{meta-train}$, where each task has its training and test sets. We use $\mathcal{T}_i = \{D_i^{train}, D_i^{test}\}$ $(i = 1, 2, ..., N)$, where $\mathcal{T}_i \in \mathcal{D}_{meta-train}$ to denote the i-th task (also called episode) during the meta-learning phase. The notations D_i^{train} and D_i^{test} correspond to the training set and the test set of the i-th task, respectively. Note that during the meta-learning phase, both D_i^{train} and D_i^{test} consist of labeled examples. We consider each camera scene as a task in the meta-learning

formulation. Each of the training i-th scene consists of M labeled images. However, in our work, we randomly sample a small number $K \in \{1, 5\}$ and $K \ll M$ labeled images for the i-th scene in each learning iteration to form D_i^{train}. The D_i^{test} is the test set for the i-th scene. This setup reflects the real-world problem of having to learn from a few labeled images. Our goal of the meta-learning is to learn the model in a way that it can adapt to a new scene using only a few training examples from the new scene. During testing (i.e., *meta-testing*) on $\mathcal{D}_{meta-test}$, we are given a new target scene $\mathcal{T}_{new} = \{D_{new}^{train}, D_{new}^{test}\}$, where D_{new}^{train} consists of a few (e.g. K) labeled images from the target scene. The goal is to quickly adapt the model using D_{new}^{train} so that the adapted model performs well on D_{new}^{test} which is the test data for this target scene. In our work, we use the meta-learning approach in [12] called *MAML*. MAML learns a set of initial model parameters during the meta-training stage. The model parameters learned during *meta-training* are used for initializing the model during *meta-testing* and are later fine-tuned on the few examples from a new target task. The adapted model with fine-tuned parameters is expected to perform well on the test images from the target task.

3.2 Our Approach

Consider a crowd counting model f_θ with the model parameters θ. Given an input image x, the output of $f_\theta(x)$ is a crowd density map representing the density level at different spatial locations in the image. The crowd count can be obtained by summing over entries in the generated density map. When learning to adapt to a particular scene $\mathcal{T}_i$, the model parameters are updated using a few gradient steps to

optimize the loss function defined on D_i^{train}. This learning step can be considered as *inner-update* during meta-learning and the optimization is expressed as follows:

$$\begin{aligned} \tilde{\theta}_i &= \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \\ \text{where } \mathcal{L}_{\mathcal{T}_i}(f_\theta) &= \sum_{(x^{(j)}, y^{(j)}) \in D_i^{train}} \| f_\theta(x^{(j)}) - y^{(j)} \|_F^2 \end{aligned} \tag{3.1}$$

where $x^{(j)}$ and $y^{(j)}$ denote a training image and its corresponding ground-truth density map from the scene $\mathcal{T}_i$, respectively. We use $||\cdot||$ to denote the Frobenius norm that measures the difference between the predicted crowd density map $f_\theta(x^{(j)})$ and the ground-truth density map $y^{(j)}$. Here α is the learning rate in the *inner-update* and its value is fixed in our implementation. We then define a loss function on D_i^{test} using $\tilde{\theta}_i$ as follows:

$$\mathcal{L}_{\mathcal{T}_i}(f_{\tilde{\theta}_i}) = \sum_{(x^{(j)}, y^{(j)}) \in D_i^{test}} \| f_{\tilde{\theta}_i}(x^{(j)}) - y^{(j)} \|_F^2 \tag{3.2}$$

During the meta-learning phase, we learn the model parameters θ by optimizing $\mathcal{L}_{\mathcal{T}_i}(f_{\tilde{\theta}_i})$ across N different training scenes. This will effectively learn θ in a way that when we update θ with a few gradient steps in a new scene, the updated parameters $\tilde{\theta}$ will perform well on test images from this scene. This optimization problem (or *outer-update*) is similar to the optimization described in [12] and it is expressed as:

$$\theta = \theta - \beta \nabla_\theta \sum_{i=1}^{N} \mathcal{L}_{\mathcal{T}_i}(f_{\tilde{\theta}_i}) \tag{3.3}$$

Fig. 3.1 shows an illustration of this meta-learning inspired process. The result of the meta-learning phase is the set of model parameters θ. Given a new scene, we use θ

to initialize the model and obtain the scene adaptive parameters $\tilde{\theta}$ by fine-tuning the parameters on the few examples from the target scene with a few gradient updates. The intuition is that well-learned parameters θ should be able to generalize to new scenes with only a few gradient updates. In our implementation for few-shot scene adaptive crowd counting, we compute the second derivatives to optimize Eq. 3.3 during outer-update as described in [12].

Backbone Network Architecture: Our proposed few-shot learning approach for crowd density estimation can be used with any backbone crowd counting network architecture. In this thesis, we use the CSRNet [3] (see Fig. 3.1) as our backbone network since it has shown to achieve state-of-the-art performance in crowd counting. The network consists of a feature extractor and a density map estimator. The feature extractor uses VGG-16 [33] to extract a feature map of the input image. Following [3], we use the first 10 layers (up to Conv4_3_3) of VGG-16 as the feature extractor. The output of the feature extractor has a resolution of 1/8 of the input image. The density map estimator consists of a series of dilated convolutional layers [34] to regress the output crowd density map for the given image.

We use a pre-trained VGG-16 [33] model on ImageNet [35] to initialize the weights of the feature extractor part of our network. The weights of the dilated convolutional layers in the density map estimator part of the network are initialized from a Gaussian with a 0.01 standard deviation. We then train the network end-to-end on the training set of WorldExpo'10 [4] dataset to learn how to produce a density map for an image containing the crowd. We refer to this trained network as "*Baseline pre-*

trained" in the remaining of the thesis. Note that although the baseline pre-trained model is learned on data from multiple training scenes, it is susceptible when used for adaptation in a few labeled data regimes as it is not specifically designed to learn from few images which we discuss in the later section. Therefore, to overcome this limitation, we use this baseline pre-trained network as the initialization for the meta-learning phase. During meta-learning, we fix the parameters of the feature extractor and train only density map estimator on different scene-specific data. We follow the training scheme described in this section to learn to adapt to a scene with a few labeled images.

3.3 Experiments

In this section, we first introduce the datasets and experiment setup (Sec. 3.3.1). We then describe several baselines for comparison (Sec. 3.3.2). We present the experimental results (Sec. 3.3.3).

3.3.1 Datasets and Setup

Datasets: Most of the available datasets for crowd-counting are not specifically designed for the scene adaptive crowd counting problem. Our problem formulation requires that the training images are from multiple scenes. To the best of our knowledge, WorldExpo'10 [4] is the only dataset with multiple scenes. We use this dataset for the training of our model. We also consider two other datasets (Mall [6] and UCSD [5]) for cross-dataset testing. The details of these datasets are described below.

The WorldExpo'10 [4] dataset consists of 3980 labeled images from 1132 video sequences based on 108 different scenes. We consider 103 scenes for training and the remaining 5 scenes for testing. The image resolution is fixed at 576×720. When testing on a target scene, we randomly choose $K \in \{1, 5\}$ images from the available images in this scene and use them for obtaining the scene adaptive model parameters $\tilde{\theta}$ (see Fig. 3.1). We then use the remaining images from this scene to calculate the performance of the parameters $\tilde{\theta}$. The Mall [6] dataset consists of 2000 images from the same camera setup inside a mall. The resolution of each image is 640×480. We follow the standard split, which consists of 800 training images and 1200 test images. Similar to the setup explained earlier, we consider $K \in \{1, 5\}$ images from the training set for fine-tuning the model to obtain the scene adaptive model parameters $\tilde{\theta}$ and later test the model on the test set. The UCSD [5] dataset consists of 2000 images from the same surveillance camera setup to capture a pedestrian scene. The crowd density is relatively sparse, ranging from 11 to 46 persons in an image. The resolution of each image is 238×158. We follow the standard split by considering the first 800 frames for training and 1200 images for testing. We use the same experiment setup of the Mall dataset.

Ground-truth Density Maps: All datasets come with dot annotations, where each person in the image is annotated with a single point. Following [3, 11], we use a Gaussian kernel to blur the point annotations in an image to create the ground-truth density map. We set the value of $\sigma = 3$ in the Gaussian kernel by following [3].

Implementation Details: We use PyTorch [36] for the implementation of our approach. The backbone crowd counting network is implemented based on the source

code from the original CSRNet paper [3]. To generate the *Baseline pre-trained* network, we follow the procedure described in [3]. During the meta-learning phase, we initialize the network with baseline pre-trained model. We freeze the feature extractor and only train the density map estimator of the network. We set the hyper-parameters $\alpha = 0.001$ for the inner-update in SGD (see Eq. 3.1) and $\beta = 0.001$ in the outer-update (see Eq. 3.3) in Adam [37]. We randomly sample a scene for each episode during inner-update.

Evaluation Metrics : To evaluate the results, we use the standard metrics in the context of crowd count estimation. The metrics are: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Deviation Error (MDE) as expressed below:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\delta_i^{\hat{y}} - \delta_i^{y}| \tag{3.4}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|\delta_i^{\hat{y}} - \delta_i^{y}|^2} \tag{3.5}$$

$$MDE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\delta_i^{\hat{y}} - \delta_i^{y}|}{\delta_i^{y}} \tag{3.6}$$

where N is the total number of images in a given camera scene, $\delta_i^{\hat{y}}$ represents the crowd count of the density map generated by the model and δ_i^{y} is the corresponding crowd count of ground-truth density map for the i-th input image. Let $p_{h,w}$ be the value at the spatial location (h, w) in a density map for an image i, the count δ_i for the image can be expressed $\delta_i = \sum_{h=1}^{H}\sum_{w=1}^{W} p_{h,w}$, where $H \times W$ is the spatial size of the density map.

3.3.2 Baselines

We define the following baselines for comparison. Note that these baselines have the same backbone architecture as our approach.

Baseline pre-trained: This baseline is a standard crowd counting model as in [3] trained in a standard supervised setting. The model parameters are trained from all images in the training set. Once the training is done, the model is evaluated directly on images in the new target scene without any adaptation. Note that, the original model in [3] uses the perspective maps and ground-truth ROI to enhance the final scores, we do not use them for the sake of simplicity.

Baseline fine-tuned: In this baseline, we first consider the *Baseline pre-trained* crowd counting model learned θ from the standard supervised setting. For a given new scene during testing, we fix the parameters of the feature extractor and fine-tune only the density map estimator using a few images $K \in \{1, 5\}$ from the target scene.

Meta pre-trained: This baseline is similar to our approach, but without the fine-tuning on the target scene. Intuitively, it is similar to "baseline pre-trained".

3.3.3 Experimental Results

Main Results: Table 3.1 shows the results on the WorldExpo'10 dataset for the 5 test (or target) scenes. We show the results of using both $K = 1$ and $K = 5$ images for fine-tuning in the test scene. This dataset also comes with ground-truth region-of-interest (ROI). We report the results *with* ($w/$) and *without* (w/o) ROI. We repeat the experiments 5 times in each setting with K randomly selected images. We average the scores across the 5 trials and report the standard deviation along with the mean

Target	Methods	1-shot (K=1)			5-shot (K=5)		
		MAE	RMSE	MDE	MAE	RMSE	MDE
Scene 1	Baseline pre-trained	5.55	6.31	0.70	5.55	6.31	0.70
	Baseline fine-tuned	5.45 ± 0.03	6.23 ± 0.03	0.68 ± 0.004	5.06 ± 0.11	5.88 ± 0.10	0.63 ± 0.005
	Meta pre-trained	4.63	5.5	0.529	4.63	5.5	0.529
	Ours w/o ROI	3.47 ± 0.01	**4.19 ± 0.01**	0.50 ± 0.007	3.42 ± 0.03	4.81 ± 0.007	**0.29 ± 0.004**
	Ours w/ ROI	**3.19 ± 0.03**	4.30 ± 0.07	**0.38 ± 0.03**	**3.05 ± 0.06**	**4.19 ± 0.15**	0.31 ± 0.08
Scene 2	Baseline pre-trained	24.07	34.29	0.17	24.07	34.29	0.17
	Baseline fine-tuned	22.74 ± 0.47	32.92 ± 0.66	0.15 ± 0.003	20.84 ± 1.03	30.49 ± 1.37	0.156 ± 0.001
	Meta pre-trained	21.65	30.51	0.185	21.65	30.51	0.185
	Ours w/o ROI	12.05 ± 0.74	16.62 ± 1.10	0.11 ± 0.007	11.41 ± 0.54	15.35 ± 0.51	0.11 ± 0.015
	Ours w/ ROI	**11.17 ± 1.01**	**15.50 ± 1.18**	**0.11 ± 0.012**	**10.73 ± 0.36**	**14.95 ± 0.60**	**0.10 ± 0.003**
Scene 3	Baseline pre-trained	35.54	40.78	0.40	35.54	40.78	0.40
	Baseline fine-tuned	33.89 ± 0.26	39.33 ± 0.25	0.38 ± 0.03	31.05 ± 0.41	36.70 ± 0.43	0.34 ± 0.004
	Meta pre-trained	36.18	42.32	0.402	36.18	42.32	0.402
	Ours w/o ROI	8.15 ± 0.17	11.04 ± 0.42	**0.09 ± 0.04**	8.31 ± 0.54	**10.75 ± 0.54**	0.10 ± 0.009
	Ours w/ ROI	**8.07 ± 0.23**	**10.92 ± 0.21**	0.10 ± 0.007	**8.18 ± 0.24**	10.96 ± 0.31	**0.09 ± 0.002**
Scene 4	Baseline pre-trained	23.95	28.57	0.19	23.95	28.57	0.19
	Baseline fine-tuned	15.69 ± 0.28	18.96 ± 0.27	0.14 ± 0.003	16.67 ± 0.10	19.70 ± 0.16	0.15 ± 0.002
	Meta pre-trained	22.44	28.25	0.183	22.44	28.25	0.183
	Ours w/o ROI	9.74 ± 0.09	11.9 ± 0.12	0.084 ± 0.001	11.21 ± 0.47	16.1 ± 0.45	0.118 ± 0.004
	Ours w/ ROI	**9.39 ± 0.26**	**11.78 ± 0.34**	**0.07 ± 0.02**	**9.41 ± 0.21**	**11.91 ± 0.17**	**0.08 ± 0.002**
Scene 5	Baseline pre-trained	10.70	13.0	0.67	10.70	13.0	0.67
	Baseline fine-tuned	8.9 ± 0.05	11.7 ± 0.04	0.50 ± 0.03	7.79 ± 0.35	10.57 ± 0.66	0.44 ± 0.015
	Meta pre-trained	9.78	12.26	0.605	9.78	12.26	0.605
	Ours w/o ROI	4.09 ± 0.01	7.36 ± 0.01	0.196 ± 0.001	4.28 ± 0.14	7.68 ± 0.60	0.20 ± 0.001
	Ours w/ ROI	**3.82 ± 0.05**	**6.91 ± 0.11**	**0.192 ± 0.001**	**3.91 ± 0.26**	**7.18 ± 0.85**	**0.18 ± 0.001**
Average	Baseline pre-trained	19.96	24.59	0.42	19.96	24.59	0.42
	Baseline fine-tuned	17.33	21.82	0.37	16.28	20.66	0.34
	Meta pre-trained	18.93	23.76	0.38	18.93	23.76	0.38
	Ours w/o ROI	7.5	10.22	0.197	7.7	10.93	0.165
	Ours w/ ROI	**7.12**	**9.88**	**0.172**	**7.05**	**9.83**	**0.155**

Table 3.1: Results on WorldExpo'10 [4] test set with $K = 1$ and $K = 5$ train images in the targe scene. We report the performance our our approach with and without ROI. We also compare with three baselines *Baseline pre-trained*, *Baseline fine-tuned* and *Meta pre-trained*. We compare the results across 5 test scenes and the last two rows represent the average score for our models.

Methods	1-shot (K=1)			5-shot (K=5)		
	MAE	RMSE	MDE	MAE	RMSE	MDE
Baseline pre-trained	7.29	7.96	0.22	7.29	7.96	0.22
Baseline fine-tuned	7.11 ± 0.09	7.80 ± 0.08	0.21 ± 0.003	6.58 ± 0.07	7.32 ± 0.06	0.20 ± 0.002
Meta pre-trained	7.01	7.69	0.230	7.01	7.69	0.230
Ours w/o ROI	2.52 ± 0.08	3.26 ± 0.12	0.078 ± 0.002	2.53 ± 0.18	3.25 ± 0.27	0.078 ± 0.004
Ours w/ ROI	**2.44 ± 0.02**	**3.12 ± 0.03**	**0.076 ± 0.001**	**2.37 ± 0.02**	**3.04 ± 0.01**	**0.073 ± 0.001**

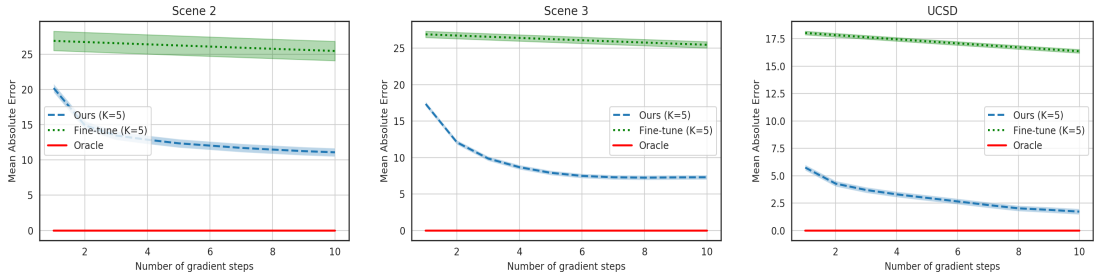
Table 3.2: Results on the Mall [6] dataset with $K = 1$ and $K = 5$ images in the target scene. The meta-training is performed on the WorldExpo'10 training data.

Methods	**1-shot (K=1)**			**5-shot (K=5)**		
	MAE	RMSE	MDE	MAE	RMSE	MDE
Baseline pre-trained	17.07	18.13	0.63	17.07	18.13	0.63
Baseline fine-tuned	16.41 ± 0.24	17.50 ± 0.23	0.60 ± 0.010	14.33 ± 0.16	15.55 ± 0.15	0.54 ± 0.006
Meta pre-trained	16.45	16.7	0.627	16.45	16.7	0.627
Ours w/o ROI	4.32 ± 0.74	5.57 ± 0.98	0.15 ± 0.022	3.82 ± 0.39	4.87 ± 0.58	0.14 ± 0.012
Ours w/ ROI	**3.08 ± 0.13**	**4.16 ± 0.23**	**0.12 ± 0.005**	**3.41 ± 0.26**	**4.22 ± 0.36**	**0.12 ± 0.007**

Table 3.3: Results on the UCSD [5] dataset with $K = 1$ and $K = 5$ images in the target scene. The meta-training is performed on the WorldExpo'10 training data.

of the scores in Table 3.1. We report the results from our models as "*Ours w/o ROI*" and "*Ours w/ ROI*". We compare with the three baselines defined in Sec. 3.3.2. Our models outperform the baselines in most cases. This shows that the meta-learning fine-tuning improves the model's performance. Note that our problem setup requires K labeled images in the test set and hence these K images have to be excluded in the calculation of the evaluation metrics, i.e., we have slightly fewer test images for the results in Table 3.1. Therefore, the performance numbers in Table 3.1 should not be directly compared with previously reported numbers in the crowd counting literature since our problem formulation is completely different. Besides, some previous crowd counting works [3] use additional components (e.g., perspective maps) to enhance the final performance. We do not consider these additional components in our models for the sake of simplicity (also the publicly available source code for [3] does not implement those extra components), so the number for "Baseline pre-trained" in Table 3.1 is slightly worse than the number reported in [3].

Table 3.2 and Table 3.3 show the results on the Mall and UCSD datasets, respectively. Here we use the training data of WorldExpo'10 for the meta-learning. We then use Mall and UCSD for the scene adaptation and evaluation. This cross-dataset testing can demonstrate the generalization of the proposed method. Our model clearly



(a) (b) (c)

Figure 3.2: Quantitative results of the learning curve during *meta-testing*. The graph (a) shows the learning for Scene 2 and (b) shows the result for Scene 3 in WorldExpo [4] test sets, respectively. Similarly, (c) shows the learning on UCSD [5]. Note that our approach continues to learn and achieves a lower MAE compared to the baseline fine-tuning approach in ten gradient steps. We consider $K = 5$ labeled examples in all three cases.

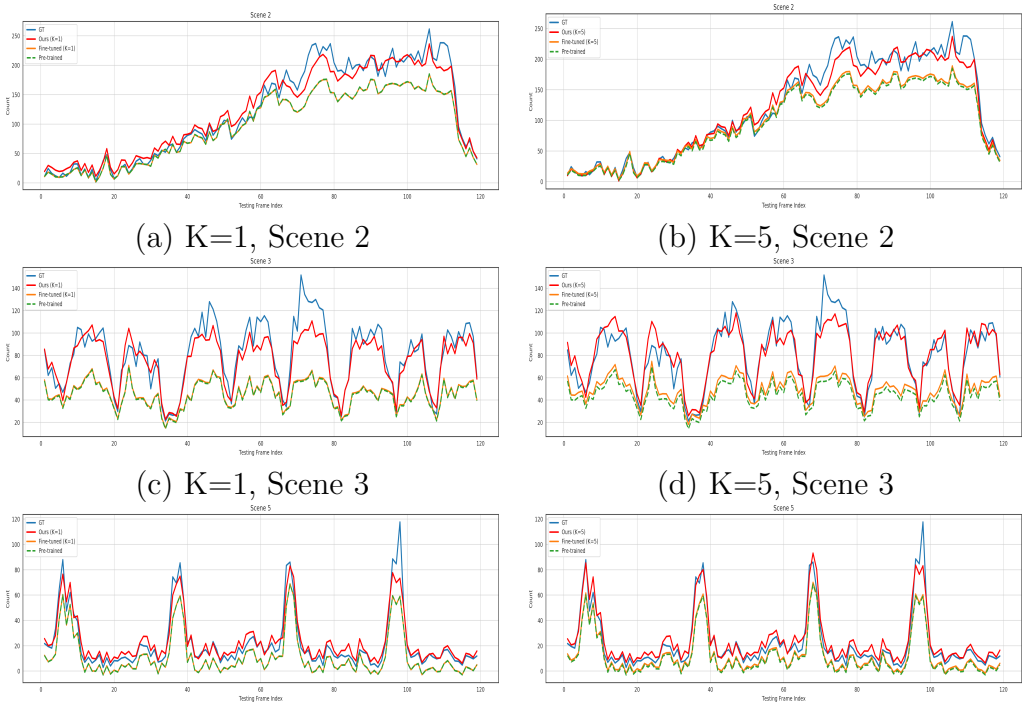
outperforms the baselines.

To gain further insights into our method, we visualize the MAE over the number of gradient steps in Fig. 3.2 for different scenes. In all the cases, our proposed approach has a better start in learning and improves continuously with more gradient steps. In the three cases shown in Fig. 3.2, our approach performs significantly better that fine-tuning with the same number of gradient updates.

Methods	**1-shot (K=1)**	
	MAE	RMSE
Hossain *et al.* [1]	8.23	12.08
Ours w/o ROI	7.5	10.22
Ours w/ ROI	**7.12**	**9.88**

Table 3.4: Comparison of results on the WorldExpo'10 [4] dataset with $K = 1$ images in the target scene with Hossain *et al.* [1]. We use the standing train/test split on WorldExpo'10. Our approach outperforms Hossain *et al.* [1].

In Fig. 3.3, we show the comparison of the crowd count estimations between our approaches and baselines in different scenes in WordExpo'10. Our method consistently produces crowd counts that are closer to the ground-truth compared with



(a) K=1, Scene 2 (b) K=5, Scene 2

(c) K=1, Scene 3 (d) K=5, Scene 3

(e)K=1, Scene 5 (f) K=5, Scene 5

Figure 3.3: Crowd counting performance comparison between the baselines and our approaches in different scene-specific images from WorldExpo'10 [4] dataset. The labels include, (a) $K = 1$ in Scene 2, (b) $K = 5$ in Scene 2, (c) $K = 1$ in Scene 3, (d) $K = 5$ in Scene 3, (e) $K = 1$ in Scene 5 and (f) $K = 5$ in Scene 5. Note that our approaches outperform the baselines in different settings and is robust to varying crowd density.

other baselines.

In Table 3.4, we compare our results with the one-shot scene adaptation proposed in [1] based on the standard WorldExpo'10 data split. In [1], the last two layers in the pre-trained model are fine-tuned to adapt to the target scene. Our approach outperforms [1].

Ablation Studies: Our approach is based on MAML [12]. In the literature, there are other optimized-based meta-learning approaches, e.g., [7, 8]. We perform additional ablation studies on the effect of different meta-learning frameworks. The results are

Target	Methods	1-shot (K=1)			5-shot (K=5)		
		MAE	RMSE	MDE	MAE	RMSE	MDE
WorldExpo (Avg.)	Meta-LSTM [8]	13.33	18.22	0.252	12.7	16.61	0.223
	Reptile [7]	11.63	15.07	0.260	8.20	11.31	0.181
	Ours w/o ROI	7.5	10.22	0.197	7.7	10.93	0.165
	Ours w/ ROI	**7.12**	**9.88**	**0.172**	**7.05**	**9.83**	**0.155**
Mall	Meta-LSTM [8]	3.95 ± 0.04	4.34 ± 0.537	0.12 ± 0.002	3.54 ± 0.44	4.41 ± 0.472	0.10 ± 0.014
	Reptile [7]	2.55 ± 0.07	3.26 ± 0.09	0.079 ± 0.001	2.49 ± 0.23	3.20 ± 0.29	0.078 ± 0.006
	Ours w/o ROI	2.52 ± 0.08	3.26 ± 0.12	0.078 ± 0.002	2.53 ± 0.18	3.25 ± 0.27	0.078 ± 0.004
	Ours w/ ROI	**2.44 ± 0.02**	**3.12 ± 0.03**	**0.076 ± 0.001**	**2.37 ± 0.02**	**3.04 ± 0.01**	**0.073 ± 0.001**
UCSD	Meta-LSTM [8]	14.15 ± 0.48	16.29 ± 0.425	0.463 ± 0.018	13.81 ± 0.10	15.99 ± 0.009	0.45 ± 0.004
	Reptile [7]	5.64 ± 2.05	6.85 ± 2.06	0.20 ± 0.075	4.48 ± 0.88	5.62 ± 0.99	0.166 ± 0.033
	Ours w/o ROI	4.32 ± 0.74	5.57 ± 0.98	0.15 ± 0.022	3.82 ± 0.39	4.87 ± 0.58	0.14 ± 0.012
	Ours w/ ROI	**3.08 ± 0.13**	**4.16 ± 0.23**	**0.12 ± 0.005**	**3.41 ± 0.26**	**4.22 ± 0.36**	**0.12 ± 0.007**

Table 3.5: The overall results for adaptation on WorldExpo'10 [4] test set, Mall [6] and UCSD [5] with $K = 1$ and $K = 5$ train images. We compare with other optimization based meta-learning approaches "Reptile" [7] and "Meta-LSTM" [8].

shown in Table 3.5. Nichol *et al.* [7] propose an optimization based meta-learning approach similar to [12] using gradient descent. However, [7] differs from [12] in that it does not consider the second-order derivative in the meta-optimization. As a result, its performance is slightly lower as reported in Table 3.5 although it converges faster. In case of [8], the meta-learner is a LSTM based model unlike in [12] and [7], because of the similarity between the gradient update in backpropagation and the cell-state update in LSTM. The drawback of [8] is the large number of trainable parameters and different architectures for the learner and meta-learner. In general, the MAML-based meta learning that our approach uses outperforms other optimization-based meta-learning approaches. In Table 3.5, we highlight only the average score across 5 scenes in WorldExpo due to page limit. The training scheme for [7, 8] is similar to our approach based on the same backbone network [3] as described in the implementation details (Sec 3.3.1). For the hyperparameters setting, please refer to [7, 8].

Chapter 4

AdaCrowd: Unlabeled Scene Adaptive Crowd Counting

4.1 Problem Setup

In this section, we first review several standard setups for crowd counting that have been studied in the literature. These setups have various limitations in real-world deployment. This 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 $f_\theta : x \to y$ that maps an image x to a density map y. The model parameters θ are learned from labeled training images. There has been lots of previous work on designing powerful models (e.g., [3, 4, 9, 10, 11, 20, 34, 38, 39, 40, 41, 42, 43, 44]).

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 adaption 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 [45, 46] 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 [1, 2] 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 [1, 2] 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 thesis, we push the limit on data requirements and overcome gradient updates at test time by proposing the following setup.

Unlabeled Scene Adaptation (our approach): 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 [1, 2], 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.

4.2 Our Approach

In Fig. 4.1, 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.

In the following, we first describe the crowd counting network with guided batch normalization in Sec. 4.2.1. We give details of the guiding network in Sec. 4.2.3. Finally, we describe the learning and inference of our model in Sec. 4.3.

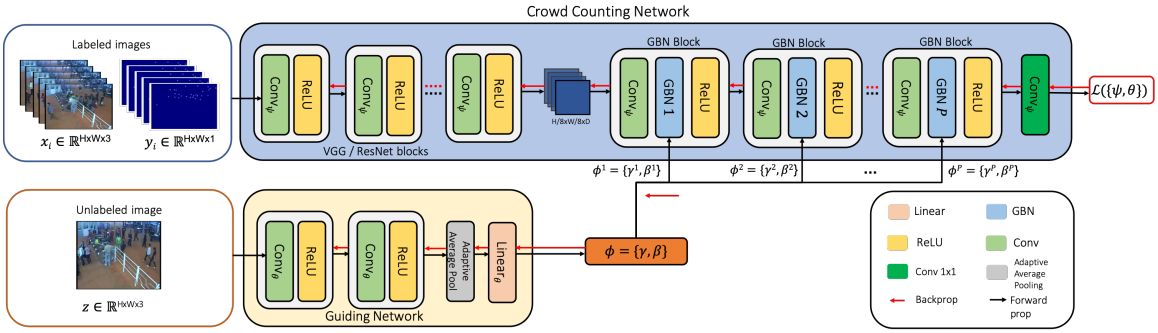


Figure 4.1: Illustration of our AdaCrowd framwork. (Top row) The crowd counting network takes an input image and estimates the density map. The crowd counting network has P number of GBN blocks with the proposed guided batch normalization (GBN) layers. The parameters $\phi = \{\gamma, \beta\}$ for all the GBN layers are predicted from another neural network called the guiding network. (Bottom row) The guiding network takes the unlabeled image (z) from the training scene and produces GBN parameters specifically adapted to that scene. Through these scene adapted GBN parameters, our model achieves adaptation to different scenes.

4.2.1 Crowd Counting Network with 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 [47], 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 γ and β over the examples $\{x_i : i = 1, 2, ..., B\}$ in the mini-batch, in

GBN, we directly predict them using the guiding net (see Sec. 4.2.3). The overall computation in GBN can be summarized as follows:

$$\hat{x}_i[h,w,d] = GBN(x_i[h,w,d]; \gamma_d, \beta_d), \text{ where} \tag{4.1a}$$

$$GBN = \gamma_d \cdot \big((x_i[h,w,d] - \mu_d)/\sigma_d\big) + \beta_d \tag{4.1b}$$

$$\forall \;\; i \in \{1..B\}, h \in \{1..H\}, w \in \{1..W\}, d \in \{1..D\} \tag{4.1c}$$

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 $x_i[h,w,d]$. The mean μ_d and standard deviation σ_d are calculated from the activation in channel d:

$$\mu_d = \frac{\sum_{i=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W} x_i[h,w,d]}{B \times H \times W} \tag{4.2a}$$

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^{B}\sum_{h=1}^{H}\sum_{w=1}^{W} \big(x_i[h,w,d] - \mu_d\big)^2}{B \times H \times W} + \epsilon} \tag{4.2b}$$

where $\mu_d, \sigma_d \in \mathbb{R}$ and ϵ is used for numerical stability. In Eq. 4.1, 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 γ and β 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 γ_d^p, β_d^p.

The proposed GBN layer is related to conditional batch normalization (CBN) [48]. 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

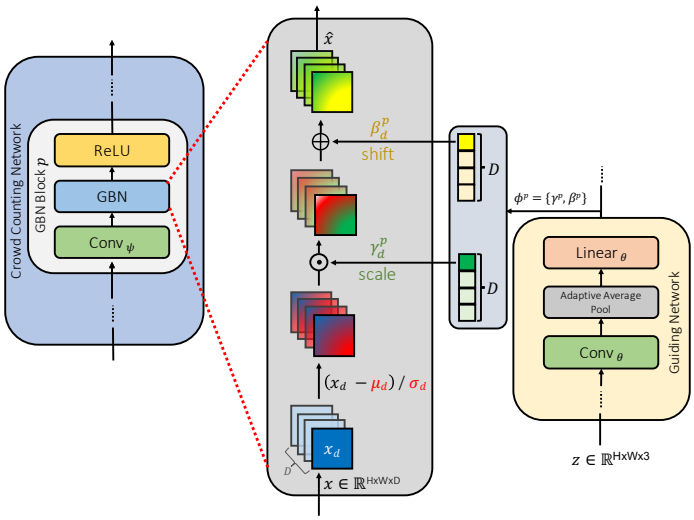


Figure 4.2: Overview of a GBN block in the `AdaCrowd` framework. For instance, given an input x to the GBN layer in GBN Block p, we first normalize x along the channel dimensions in the GBN layer. We then use the affine transformation parameters γ_d^p and β_d^p corresponding to the p-th GBN block to uniformly scale and shift the activation features to generate the output $\hat{x}$. The affine parameters are generated by the guiding network based on the scene-specific unlabeled data z.

information from spatial input data such as images. (2) CBN depends on the learned weights for γ and β obtained from BN layer in a pre-trained network for initialization. However, in GBN we directly predict γ and β 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 γ and β by small margin, whereas we completely replace the values for γ and β 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. 4.2.

4.2.2 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.

To simplify the notation, we use ϕ to denote the concatenation of all parameters in GBN layers (i.e. γ and β from all GBN layers) in the crowd counting network. We use ψ 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:

$$\hat{y} = f(x; \{\phi, \psi\}) \tag{4.3}$$

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

Note that ψ will be the same for different scenes, while ϕ will change across scenes since ϕ is the output of the guiding network (see Sec. 4.2.3). By predicting ϕ specifically to a scene, we achieve scene adaptive crowd counting.

4.2.3 Guiding Network

The goal of the guiding network is to predict the GBN parameters ϕ 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 ϕ as:

$$\phi = g(z; \theta) \tag{4.4}$$

where $g(\cdot; \theta)$ denotes a function parameterized by θ. In this thesis, $g(\cdot; \theta)$ 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 ($\phi = \frac{1}{K} \sum_{i=1}^{K} \phi_i$) the predicted ϕ_i over K inputs.

Discussion: The idea of using adjustable GBN parameters for adaptation is inspired by the work in image translation [49]. Similar to [49], 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 layers (which are spatially invariant), the parameter ϕ 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 ψ and are shared across different scenes.

4.3 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 ϕ will be directly predicted from the guiding network (see Eq. 4.4). 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 ϕ obtained via Eq. 4.4 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 $\mathcal{S}^t$, 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. 4.4 to compute the GBN parameters ϕ from the guiding network parameterized by θ. 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 $\mathcal{S}^t$ as follows:

$$\mathcal{L}(\{\psi,\theta\};\mathcal{S}^t) = \sum_{i=2}^{N} ||f(x_i;\{\phi,\psi\}) - y_i||^2 \tag{4.5a}$$

$$= \sum_{i=2}^{N} ||f(x_i;\{g(z;\theta),\psi\}) - y_i||^2 \tag{4.5b}$$

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. 4.5 across all scenes during training, i.e.:

$$\min_{\{\psi,\theta\}} \sum_t \mathcal{L}(\{\psi,\theta\};\mathcal{S}^t) \tag{4.6}$$

In Eq. 4.6, 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 $\{\mathcal{S}^t\}$
Initialize the model parameters $\{\psi,\theta\}$;
while *not done* **do**
  for *each scene* $\mathcal{S}^t$ **do**
    Sample one image to obtain z;
    Compute GBN parameters ϕ using Eq. 4.4;
    Evaluate $\bigtriangledown_{\{\psi,\theta\}}\mathcal{L}(\{\psi,\theta\};\mathcal{S}^t)$ in Eq. 4.5 ;
  end
  Update $\{\psi,\theta\} \leftarrow \sum_t \bigtriangledown_{\{\psi,\theta\}}\mathcal{L}(\{\psi,\theta\};\mathcal{S}^t)$;
end

Let $\{\psi^*,\theta^*\}$ be the learned model parameters that minimize Eq. 4.6. 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:

$$\hat{y} = f(x; \{\phi^*, \psi^*\}), \text{ where } \phi^* = g(z_{new}; \theta^*) \tag{4.7}$$

4.4 Experiments

In this section, we first describe the datasets and the experiment setup in Sec. 4.4.1. We then introduce several baseline methods used for comparisons in Sec. 4.4.2. We present experimental results in Sec. 4.4.3.

4.4.1 Datasets and Setup

Datasets: We experiment with the following datasets.

- *WorldExpo'10* [4]: 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 $\times$ 672.

- *Mall* [6]: The Mall dataset consists of 2000 frames captured from a surveillance camera in a mall. The frame size is 640 $\times$ 480. The training and test sets consist of 1200 and 800 images, respectively.

- *PETS* [51]: The PETS dataset is a multi-view dataset of crowd scene from 8 views. We use the first 3 views following [52] and similarly we consider se-

quences 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 [51]. The original image size is 576×768. In our experiments, we resize the images to 288×384 following [52].

- *FDST* [53]: 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 [53], we resize all the frames to 640×360 in our experiments.

Our problem setup requires data from multiple scenes during the training phase. To the best of our knowledge, WorldExpo'10 [4] is the only crowd counting dataset with large number 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 [33] or ResNet [50]) networks. The learning rate for all the experiments is set to $1e$-5. We use Adam [37] 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 [3], [4] to evaluate the performance of our model, namely mean absolute error (MAE) and root mean squre error (RMSE). Smaller MAE/RMSE

indicates better performance.

4.4.2 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 scenes from different scenes without adaptation.

- *CSRNet* [3]: The first sub-network in CSRNet is based on VGG [33] and consists of layers till *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* [3]: This baseline is based on CSRNet. We add batch normalization layers between every *conv* and *relu* layer in the second sub-network to generate the density map for the input.

- *ResNet FCN* [20]: This baseline is based on the ResNet101 [50] 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* [20]: 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* [20]: This baseline uses ResNet101 [50] 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 [20].

- *ResNet SFCN w/ Batch Normalization* [20]: This baseline is based on ResNet SFCN. We add batch normalization layers in the density map generator like in other BN based baselines.

We also 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.

Method	MAE	RMSE
CSRNet [3]	19.56	28.34
CSRNet w/ BN [3]	18.57	29.91
Ours w/ CSRNet	**17.32** ±0.1	**27.03** ±0.05
FCN [50]	23.07	34.16
FCN w/ BN [50]	21.65	33.3
Ours w/ FCN	**20.91** ±0.3	**29.61** ±0.2
SFCN [20]	25.47	35.91
SFCN w/ BN [20]	23.84	35.52
Ours w/ SFCN	**14.56** ±0.4	**22.75** ±0.2

Table 4.1: Quantitative results for training and testing on WorldExpo'10. We report results using different backbone architectures. "Ours" correspond to using one unlabeled image z. The results for our approach show the mean and standard deviation (%) over 5 random trials. We show the best results in **bold**.

Method	WorldExpo → Mall		WorldExpo → PETS		WorldExpo → FDST	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
CSRNet [3]	9.94	10.41	17.99	19.80	12.74	13.09
CSRNet w/ BN [3]	8.72	9.92	18.63	20.49	7.30	7.81
Ours w/ CSRNet	**4.0** ±0.08	**5.0** ±0.09	**17.43** ±0.04	**19.70** ±0.05	**7.14** ±0.3	**7.77** ±0.2
FCN [50]	9.03	9.6	20.38	22.67	7.14	7.85
FCN w/ BN [50]	9.63	10.32	19.59	21.61	8.89	9.23
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
SFCN [20]	15.17	15.53	19.62	21.55	8.72	8.94
SFCN w/ BN [20]	10.8	11.39	19.94	22.26	6.60	6.96
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

Table 4.2: Quantitative results for the cross-dataset testing for one unlabeled image. We train WorldExpo'10 and test on Mall, PETS, and FDST. We show results of different backbone networks and report mean and standard deviation (%) of our models over 5 random trials. We show the results in **bold**.

4.4.3 Experimental Results

Quantitative Results: In Table 4.1, we show the average results of training with 103 scenes and testing on 5 different scenes on WorldExpo'10. In Table 4.2, we show the results of training on WorldExpo'10 and testing on other datasets (Mall, PETS and FDST). As other datasets (Mall, PETS and FDST) have same scenes in both training and testing, we use unlabeled image from the train set when sampling z and 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. We report the mean score with the standard deviation (%) in Table 4.1 and 4.2. In all the cases, our `AdaCrowd` significantly outperforms other baselines.

Qualitative Results: In Fig. 4.3, we present qualitative examples from different datasets. We visualize the predicted density map and the ground-truth count.

Ablation Analysis: We perform additional analysis on the proposed `AdaCrowd`

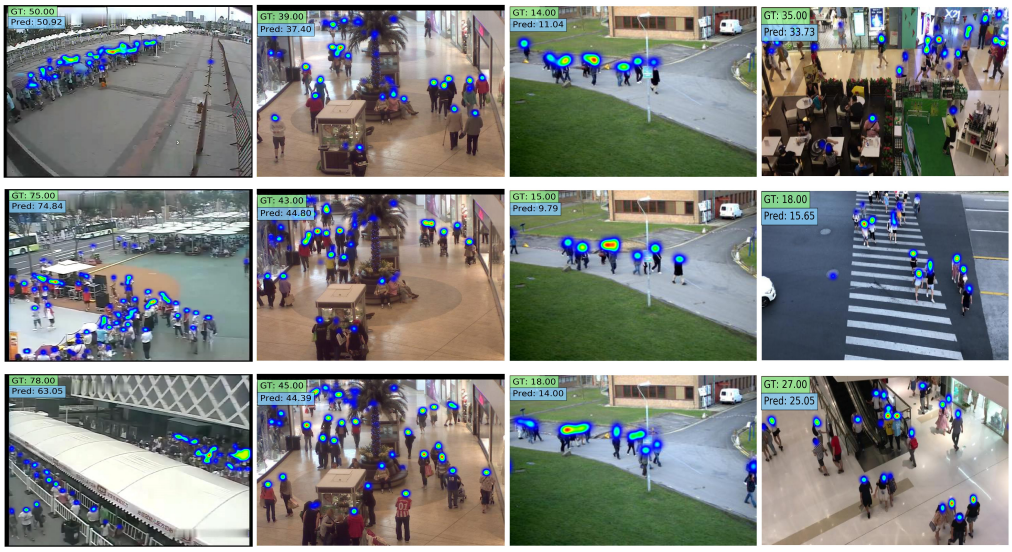


Figure 4.3: Qualitative results of our approach on different datasets. We visualize the density maps and show both ground-truth and predicted count at the top-left corner of each image.

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

Table 4.3: 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.

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 4.3. 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. 4.4, we present the analysis on how the performance varies when we vary

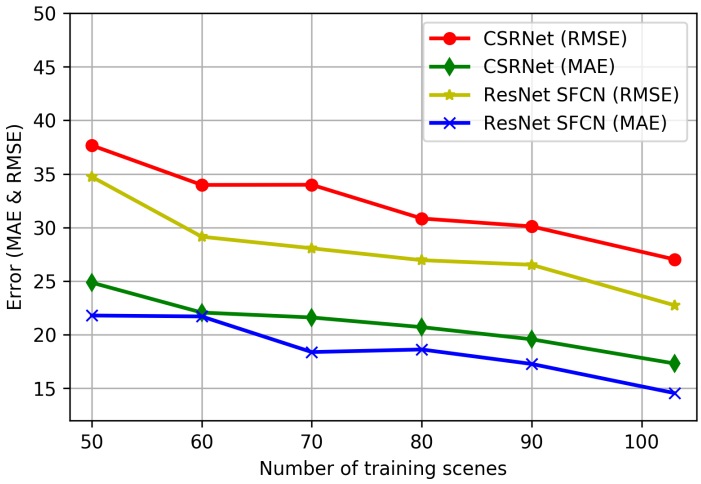


Figure 4.4: We present the ablation study on the relation between network performance and number of training scenes on WorldExpo'10 dataset with CSRNet and ResNet SFCN architectures.

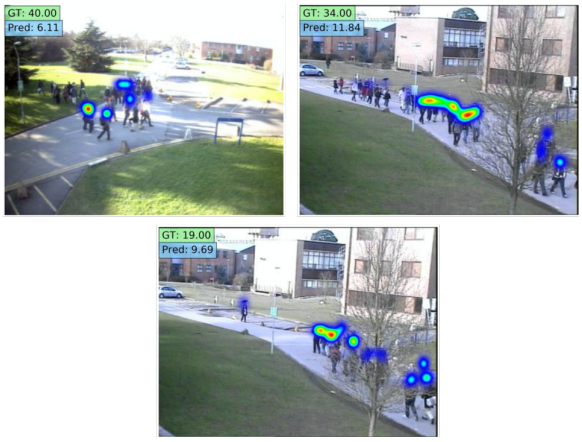


Figure 4.5: We provide an overview of some failure cases caused by drastic changes in the target scene images due to illumination, occlusion or image quality. Nevertheless, our approach still performs better than alternative methods in these cases.

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. 4.5, 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 these challenging cases.

Chapter 5

Conclusion and Future Work

In this thesis, we have presented effective deep learning approaches for scene-specific adaptation for crowd counting. More specifically, we have proposed two adaptation methods to handle both labeled and unlabeled scene-specific data. First, we have addressed the problem of few-shot scene adaptation for crowd counting. We have proposed a meta-learning inspired approach to address the learning mechanism for few-shot scenario. Our proposed approach learns the model parameters in a way that facilitates fast adaptation to new target scenes. Second, 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. 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.

In terms of future direction, we would like to explore the aspects of synthetic to real-world scene adaptation and multi-task learning by considering additional tasks to improve the primary crowd counting performance.

Bibliography

[1] M. A. Hossain, M. K. K. Reddy, M. Hosseinzadeh, O. Chanda, and Y. Wang, "One-shot scene-specific crowd counting," in *British Machine Vision Conference (BMVC)*, 2019.

[2] M. K. K. Reddy, M. 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.

[3] Y. Li, X. Zhang, and D. Chen, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] 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.

[5] 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.

[6] 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.

[7] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

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

[9] 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)*, 2017.

[10] V. A. Sindagi and V. M. Patel, "Generating high-quality crowd density maps using contextual pyramid CNNs," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[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] 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.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[14] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.

[15] A. B. Chan and N. Vasconcelos, "Bayesian poisson regression for crowd counting," in *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[16] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[17] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[18] V.-Q. Pham, T. Kozakaya, O. Yamaguchi, and R. Okada, "Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[19] C. C. Loy, K. Chen, S. Gong, and T. Xiang, "Crowd counting and profiling: Methodology and evaluation," in *Modeling, Simulation and Visual Analysis of Crowds*, 2013.

[20] 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.

[21] D. Kang, D. Dhar, and A. Chan, "Incorporating side information by adaptive convolution," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[22] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2006.

[23] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum, "One-shot learning by inverting a compositional causal process," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[24] Z. Luo, Y. Zou, J. Hoffman, and L. F. Fei-Fei, "Label efficient learning of transferable representations acrosss domains and tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[25] S. Bengio, Y. Bengio, J. Cloutier, and J. Gecsei, "On the optimization of a synaptic learning rule," in *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, 1992, pp. 6–8.

[26] J. Schmidhuber, "Evolutionary principles in self-referential learning," *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[27] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *International Conference on Machine Learning (ICML)*, 2015.

[28] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[29] 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.

[30] 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.

[31] T. Munkhdalai and H. Yu, "Meta networks," in *International Conference on Machine Learning (ICML)*, 2017.

[32] 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.

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

[34] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Advances in Neural Information Processing Systems Workshop (NeurIPSW)*, 2017.

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

[38] 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.

[39] D. Kang and A. Chan, "Crowd counting by adaptively fusing predictions from an image pyramid," in *British Machine Vision Conference (BMVC)*, 2018.

[40] 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.

[41] D. B. Sam, N. N. Sajjan, H. Maurya, and R. V. Babu, "Almost unsupervised learning for dense crowd counting," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[42] 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.

[43] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *The European Conference on Computer Vision (ECCV)*. Springer, 2016.

[44] J. Wan and A. Chan, "Adaptive density map generation for crowd counting," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.