

Deep Learning Models for Crowd Counting in Images

by

Mohammad Asiful Hossain

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
University of Manitoba
Winnipeg, MB, Canada

Copyright © 2019 by Mohammad Asiful Hossain

Thesis advisor

Author

Dr. Yang Wang

Mohammad Asiful Hossain

Deep Learning Models for Crowd Counting in Images

Abstract

Crowd counting on images has become a challenging task for computer vision research. Given an image of a crowded scene, our goal is to estimate the density map of this image, where each pixel value in the density map corresponds to the crowd density at the corresponding location in the image. Given the estimated density map, the final crowd count can be obtained by summing over all values in the density map. One challenge of crowd counting is the scale variation in images. In this thesis, we propose different deep learning models to solve problems regarding crowd counting. This thesis consists of three works which are disjoint but problem domain is similar which is crowd counting. We got reasonably better performance in all these works on benchmark dataset.

Contents

Abstract	ii
Table of Contents	iv
List of Figures	v
List of Tables	vii
Acknowledgments	x
Dedication	xi
Publications	xii
1 Introduction	1
1.1 Thesis Organization	3
2 Related Work	4
2.1 Crowd Counting Using Scale-Aware Attention Networks	4
2.2 One-Shot Specific Crowd Counting	6
2.3 Domain Adaptation in Crowd Counting	7
3 Crowd Counting Using Scale-Aware Attention Networks	10
3.1 Contributions	11
3.2 Our Approach	11
3.2.1 Multi-Scale Feature Extractor	12
3.2.2 Global Scale Attention	13
3.2.3 Local Scale Attention	14
3.2.4 Fusion Network	15
3.2.5 Loss Function	17
3.3 Experiments	19
3.3.1 Datasets and Setup	19
3.3.2 Experimental Results	21
3.3.3 Ablation Study	22
4 One-shot Scene Specific Crowd Counting Using Domain Adaptation	27
4.1 Contributions	28

4.2	Approach	28
4.2.1	Model Architecture	29
4.2.2	One-Shot Scene-Specific Adaptation	30
4.2.3	Model Learning	32
4.3	Experiments	36
4.3.1	Baselines	36
4.3.2	Datasets and Setup	37
4.3.3	Main Results	39
4.3.4	Cross-Dataset Testing	44
5	Domain Adaptation in Crowd Counting	45
5.1	Contributions	46
5.2	Our Approach	47
5.2.1	Network Architecture	47
5.2.2	Domain Adaptation for Crowd Counting	49
5.3	Experiments	51
5.3.1	Datasets and Setup	51
5.3.2	Baselines	54
5.3.3	Experimental Results	56
5.3.4	Analysis	57
6	Conclusion and Future Work	61
	Bibliography	63

List of Figures

1.1	Sample crowd images from different dataset [1; 2; 3; 4].	2
3.1	Overview of the proposed method. The input image is passed simultaneously through three sub-networks: multi-scale feature extractor (MFE), global scale attentions (GSA), and local scale attentions (LSA). MFE extracts feature maps in three different scales. GSA and LSA produce three global scores and three pixel-wise local attention maps, respectively. Multi-scale feature maps are then weighted by the corresponding GSA and LSA outputs. The attention-weighted features are then used as the input to the fusion network to predict the density map. Finally, the crowd count can be obtained by summing over the entries in the predicted density map.	12
3.2	Illustration of the multi-branch feature extractor (MFE) module. Here we use three branches associated with three different scales. Each branch uses different filter sizes to extract features at different scales. Using max-pooling, the output of each branch has a spatial dimension equal to one-fourth of the input image size. In this figure, we use $C(i, j)$ to denote a 2D convolution layer with filter sizes of $i \times i$ and j output feature maps. We use P to denote max-pooling.	13
3.3	Illustration of the global scale attention (GSA) module. Given an input image, this module generates three attention scores. Each score corresponds to one of the three pre-defined density labels: low-density, mid-density, and high-density.	14
3.4	Local Scale Attention Network. This module produces three feature maps of size $H/4 \times W/4$ each of which associated to a density scale. $C(i, j)$ denotes a 2D convolution with a filter size of $i \times i$ and with j output feature maps. P stands for max pooling.	15

3.5	Qualitative examples of density maps. The first two columns are original input images from the Shanghaitech partB [1] dataset and their corresponding density maps respectively. The last two columns are original input images from the Mall [2] dataset and their corresponding density maps respectively.	25
4.1	Network architecture of our backbone network based on the multi-column CNN (MCNN) [1].	29
4.2	Qualitative examples of predicted density maps. The 1st and 3rd columns contain images from the WorldExpo'10 and Trancos datasets, respectively. The 2nd and 4th columns are the corresponding density maps.	41
4.3	Qualitative examples of input images and predicted density maps in the cross-dataset testing setting. The 1st and 3rd columns are images from the Mall and the UCSD datasets, respectively, the 2nd and 4th columns are their corresponding density maps. The Trancos dataset is used for training.	42
5.1	Illustration of high-level intuition of domain adaptation. We have a source domain and a target domain. The intuition of our domain adaptation approach is to minimize the domain shift between the feature representations of images from these two domains.	47
5.2	The architecture of our backbone network based on the multi-column convolutional neural network (MCNN) [1]. This architecture consists of a parallel branch to extract features at different scales. The features from all three branches are concatenated together in the end to predict the output. The output of the network is an estimated density map of the input image. Taking an integral over the density map gives the final count. In our work, we consider the blue colored network as source and the orange network as the target. We employ maximum mean discrepancy to minimize the domain shift between the source and target feature representations.	48

List of Tables

3.1	Statistics of the three datasets used in the experiments. For each dataset, we show the following information: the image resolution, the number of images, the maximum/minimum number of people in an image, the average and total number of people annotated in the dataset.	20
3.2	Comparison of the performance of different methods on the ShanghaiTech PartB dataset [1]. [†] These results are obtained from [5].	22
3.3	Comparison of the performance of different methods on the Mall dataset [2]. [†] These results are obtained from [5].	23
3.4	Comparison of the performance of different methods on the UCF_CC_50 dataset [6].	24
3.5	Effect of GSA and LSA modules on the ShanghaiTech PartB dataset. The “base model” only contains the multi-scale feature extractor and the fusion network, i.e. without any global or local attention modules. By adding GSA or LSA, we can achieve better performance. The best performance is obtained by using both GSA and LSA together with the base model.	24
3.6	Effect of using auxiliary losses as extra supervisions on the ShanghaiTech PartB dataset. L_{DM} is the model without any auxiliary losses. By adding L_{LSA} (i.e. $L_{DM} + L_{LSA}$) or L_{GSA} (i.e. $L_{DM} + L_{GSA}$), we can improve the performance. By using both L_{LSA} and L_{GSA} as auxiliary losses (i.e. $L_{DM} + L_{LSA} + L_{GSA}$), we achieve the best performance.	26
4.1	Comparison of the performance (MAE and MSE) of our approach and the baselines on the WorldExpo’10 dataset [4]. For “ours” and “simple fine-tuning”, we consider either using the last layer or the last two layers of MCNN as the decoder.	39
4.2	Comparison of the performance (MAE and MSE) of our approach and the baselines on the Trancos dataset [3]. For “ours” and “simple fine-tuning”, we consider either using the last layer or the last two layers of MCNN as the decoder.	40

4.3	Comparison of the performance (MAE and MSE) of our approach and the baselines on the WorldExpo'10 dataset [4] using the standard split.	40
4.4	Performance in the cross-dataset testing. Here we use the WorldExpo'10 dataset [4] for training and the UCSD dataset [7] for testing.	41
4.5	Performance in the cross-dataset testing. Here we use the WorldExpo'10 dataset [4] for training and the Mall dataset [2] for testing.	43
5.1	Different domain adaptation settings in our experiment. For each setting, we show the number of labeled training images in the source domain (i.e., N_s), the number of images in the target domain (N_t), and the number of test images in the target domain. For SDA and SSDA in each setting, the number of labeled training images in the target domain (i.e., K) is $\{1, 5, 10\}$.	52
5.2	An illustration of the domain shift problem in commonly used counting datasets. When learning the model on the training images of a dataset and test the model on the test data of the same dataset, the performance is generally good (first four rows). However, when we take a model learned from the training images of one dataset and test on the test images in a different dataset, the performance drops dramatically (last six rows). This demonstrates the need for domain adaptation in crowd counting. The results are obtained using a 3d-party implementation of MCNN [1] which achieves performance similar to the original numbers (e.g. MAE 110.2 and MSE 173.2 on ShanghaiTechA, as in [1]).	54
5.3	Quantitative results for the supervised (SDA) and semi-supervised (SSDA) domain adaptation settings. We compare our models ("Ours(SDA)" and "Ours(SSDA)") against the two baseline models represented as "pre-trained" and "fine-tuning". We compare the results using MAE and MSE metrics for K -shots, where $K \in \{1, 5, 10\}$. Lower MAE/MSE means better performance. The first two rows correspond to domain adaptation between ShanghaiTechA (dense crowd) and Mall (sparse crowd) datasets where the source and target domains have different density levels. The 3rd and 4th rows show the adaptation between Mall (sparse crowd) and UCSD (sparse crowd) where two domains have similar density levels. The last two rows correspond to cross-object domain adaption, where the source domain (UCSD, Mall) is for person counting and the target domain (TRANCOS) is for vehicle counting. The 1st and 2nd best results in each setting are highlighted in red and blue, respectively.	55

5.4	Quantitative results of the unsupervised domain adaptation (UDA), where the target domain only have unlabeled training images. We compare our UDA approach to the baseline which uses the pre-trained model from the source domain. Lower MAE/MSE means better performance. Our method significantly outperforms the baseline. Note that we cannot compare with the “fine-tuning” baseline in this case, since it requires labeled images in the target domain. The best results are highlighted in red.	58
5.5	Comparison of our domain adaptation approaches with two oracle methods. The two oracle methods have access to all labeled images in the target domain. Oracle-1 is a standard supervised method trained directly in the target domain. Note that Oracle-1 does not use the data in the source domain. Oracle-2 is our SSDA approach in the extreme case where all images in the target domain are labeled. The 1st and 2nd best results in each setting are highlighted in red and blue, respectively.	59

Acknowledgments

First of all, I offer my gratitude to Almighty God who gave me the ability to accomplish my goal in form of thesis without any breaking.

I am very much thankful to Dr. Yang Wang for selecting me as one of his advisee. I have learn the actual meaning of research from his guidance. Today what I am right now with some publications, reason is the encouragement of my professor. The most interesting thing I learn for you that is "If you are going to fail at something then do it faster, so we can start something new". He is a great academician and mentor who lead me in the field of computer vision and deep learning. Without his thoughtful reviews and guidance I would hardly finish this thesis work. Thank you Prof. Yang Wang, I will always be grateful for all that I have learned from you.

I would also like to thank my committee members, Dr. Carson Kai-Sang Leung and Dr. Ekram Hossain, for their valuable suggestions as well as potential feedback in perfecting my thesis. Thanks to all of the support staff in the Department of Computer Science for their helping hand. Moreover, I am grateful to - The University of Manitoba, Faculty of Graduate Studies and The Government of Manitoba for their continuous financial support. I would like to extend my gratitude to my wonderful lab mates for their time, support and suggestions and for being so nice, kind and helpful.

Last but not the least I would like to express my hearty complement to my parents and family members for their unconditional love and encouragement to come at this stage and pursue my degree.

*This thesis is dedicated to my parents
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. **Mohammad Asiful Hossain**, Mehrdad Hosseinzadeh, Omit Chanda, Yang Wang, Crowd Counting Using Scale-Aware Attention Networks, IEEE Winter Conference on Applications of Computer Vision (WACV), 2019.
2. **Mohammad Asiful Hossain**, Omit Chanda, Mehrdad Hosseinzadeh, Yang Wang, One-Shot Scene Specific Crowd Counting, The British Machine Vision Conference 2019.(under review)
3. **Mohammad Asiful Hossain**, Mahesh Kumar K, Yang Wang, Domain Adaptation in Crowd Counting, International Conference on Computer Vision (ICCV) 2019.(under review)

Chapter 1

Introduction

We consider the problem of crowd counting in arbitrary static images. Given an arbitrary image of a crowded scene without any prior knowledge about the scene (e.g. camera position, scene layout, crowd density), our goal is to estimate the density map of the input image, where each pixel value in the density map corresponds to the crowd density at the corresponding location of the input image. The crowd count can be obtained by integrating the entire density map. In particular, we focus on the setting where the training data have dotted annotations, i.e. each object instance (e.g. people) is annotated with a single point in the image.

Crowd counting has many real-world applications, such as surveillance, public safety, traffic monitoring, urban planning [8]. The methods developed for crowd counting can also be used in counting objects in many other domains, such as counting cells or bacteria in microscopic images [1], counting animals for ecologic studies [9], counting vehicles in traffic control [10; 11; 12].

The challenges of crowd counting are manifold, including severe occlusion, per-

spective distortion, difference in head sizes, diverse crowd densities, etc. Some early work on crowd counting is based on head detection [6]. In recent years, convolutional neural networks (CNNs) have become popular in crowd counting. Most of the CNN-based approaches [4; 1; 8; 13] work by estimating a density map from the image, then obtain the crowd count based on the density map. The accuracy of crowd counting largely depends on the quality of the estimated density map.

By considering these issues, we propose three works on crowd counting. One of them is crowd counting using scale-aware attention networks (published in WACV2019) which is purely supervised learning based. The second work is one-shot scene specific crowd counting(submitted to BMVC2019) based on meta-learning. The last work is crowd counting using domain shift(submitted to ICCV2019) based on domain adaptation.



Figure 1.1: Sample crowd images from different dataset [1; 2; 3; 4].

1.1 Thesis Organization

The rest of the thesis is organized as follows: First, in Chapter 2, we review the related works and relevant literature. In this chapter, we briefly describe former approaches and techniques used for supervised crowd counting, One-shot/few-shot learning and domain shifting. In Chapter 3, we describe crowd counting using scale-aware attention networks. We show that attention networks play vital role in generating better density map which contribute to improve counting error. Moreover, we use two types of attentions, global attention and local attention. In Chapter 4, we explain one shot specific crowd counting. Here, we propose a meta training approach which transfer knowledge from one domain to another domain to perform crowd counting and vehicle counting. In Chapter 5, we demonstrate domain adaptation in crowd counting. Finally, in this work we propose domain shift among different crowd based dataset by considering supervised, semi supervised and unsupervised machine learning scenarios. we conclude this thesis in Chapter 6.

Chapter 2

Related Work

2.1 Crowd Counting Using Scale-Aware Attention Networks

Most existing crowd counting approaches work by first extracting low-level features from images, then map these features to density maps or counts using various techniques. Loy et al. [15] has categorized the existing methods into three groups: (1) detection-based methods (2) regression-based methods, and (3) density estimation based methods. In the following, we briefly review some of these methods. Interested reader may refer to [15] for a more extensive review.

Early work [16] on crowd counting uses detection-based approaches. These approaches usually apply a person or head detector on an image. Detection-based approaches often cannot handle high density crowd. To address the limitation of detection-based methods, some work [6] uses a regression-based method that directly

learns the mapping from an image patch to the count.

In recent years, convolutional neural networks (CNNs) have been popular in almost all vision tasks including crowd counting. Walach et al. [17] propose a method for learning CNN-based crowd counting model in a layer-wise fashion. Zhang et al. [18] introduce a cross-scene crowd counting method by fine-tuning a CNN model to the target scene.

One particular challenge in crowd counting is the scale variation in crowd images. Zhang et al. [1] propose a multi-column architecture (MCNN) for crowd counting. This multi-column architecture uses three branches of CNNs. Each branch works at a different scale level of the input image. The three branches are fused in the end to produce the output. Onoro-Rubio and López-Sastre [10] addressed the scale variation issue by proposing a scale-aware counting model called Hydra CNN. This model is formulated as a regression model in which the network learns the way of mapping of the image patches to their corresponding object density maps. Also, Boominathan et al. [19] tackle the issue of scale variation using a combination of deep and shallow networks.

Although these methods are proved to be robust to scale variations, they have some adverse effect that limits the size of input image during training. By reducing the training image size, they would not be capable of learning the features of original image size. To address this drawback, Sindagi et al. [20] presented a novel end-to-end cascaded CNN that jointly produces the estimated count and high quality density map. The high-level features of their network enables it to learn globally relevant and discriminative features.

Sam et al. [8] introduce a coarse to fine Switch-CNN network that chooses a branch corresponding to an estimated scale instead of fusing all branches. Sindagi et al. [13] develop a contextual pyramid CNN (CP-CNN) that combines both global and local contextual information for crowd counting. Most recently, Liu et al. [5] introduce a method called DecideNet. DecideNet has two counting models – a regression mode and a detection mode. Depending on the real density condition at a location in an image, DecideNet will learn to automatically switch between these two modes.

Inspired by the previous work, we propose a new approach for handling scale variations in crowd counting. Our proposed model uses attentions to automatically focus on a particular scale, both at the whole image level and at the local patch level. Our model is conceptually simpler than DecideNet [5], since we do not need to switch between two different counting modes.

2.2 One-Shot Specific Crowd Counting

Our work is also related to one-shot learning in computer vision. The goal of one-shot learning is to learn a new concept from one or few examples. Some early work [21] performs one-shot learning of object categories by learning appearance and geometric knowledge of objects from some object categories and transferring this knowledge to other new object categories. Vinyals et al. [22] develop a matching network for one-shot learning. Santoro et al. [23] develop a memory-based neural network for one-shot learning. Some recent work in one-shot learning uses the idea of meta-learning. Bertinetto et al. [24] propose a method for learning to predict the weights of a neural network by another neural network. Wang et al. [25] develop a

learning to learn approach for learning object categories from small data. Caelles et al [26] perform one-shot object segmentation in videos by fine-tuning a pre-trained model on the first frame in a video.

Our work is also related to domain adaptation [27] where the objective is to transform the models learned in one domain (called “source domain”) to perform well in a slightly different domain (called “target domain”). In our application, we can consider each scene in the training data as source domain. The target scene during testing can be considered as the target domain. But there are some key difference between our work and the standard domain adaptation. First of all, in the standard domain adaptation setting, there are usually only a handful of source and target domains. But in our application, we would like to be able to handle any number of scenes. Second, in traditional domain adaptation, we usually have enough data (either labeled or unlabeled) from the target domain. But in our case, we assume that we only have one single example from the target scene. Due to these differences, we cannot use off-the-shelf domain adaptation approaches in our problem.

2.3 Domain Adaptation in Crowd Counting

Few-Shot Learning: The goal of few-shot learning is to learn visual representation from a small number of examples. Some early work in few-shot learning [28] uses Bayesian approaches for object recognition. Koch et al. [29] propose a deep Siamese neural network for similarity-based ranking between inputs and later using the trained model to generalize on new data from a different distribution with a single image. Lake et al. [30] propose a generative model based on one-shot learning. Vinyals et al.

[22] propose a matching network that adopts ideas from metric learning for training a model to map from small labeled and unlabeled sets to their corresponding labels. Santoro et al. [31] propose a memory-augmented neural network for learning from a few examples. Bertinetto et al. [24] propose a network to predict the parameters to a pupil network using a few examples. Finn et al. [32] propose a model agnostic meta-learning involving both the meta and base networks together during training.

Domain Adaptation: Our work is related to domain adaptation [33; 34] where the objective is to transform the models learned in one domain (i.e., source domain) to perform well in a slightly different domain (i.e., target domain). Over the years, various domain adaptation methods have been proposed. Some previous works [35; 36] focuses on instance-based domain adaptation and others [37; 38; 39; 40] focus on features domain adaptation. The instance-based approaches primarily dealt with the reduction of the discrepancy between the domains, whereas the feature-based approaches worked towards finding a common latent space to match the domains.

Some domain adaptation approach work by enforcing the features of the two domains to be similar. Gretton et al. [41] propose a statistical test based on a distance method to measure the similarity between two distributions by mapping the distributions to Reproduce Kernel Hilbert Space (RKHS) also commonly known as Maximum Mean Discrepancy (MMD). Huang et al. [42] use MMD by re-weighting training data points to minimize the domain gap. Gong et al. [38] propose an unsupervised domain adaptation framework by making the source and target domains similar. Pan et al. [40] propose a framework to discover a feature representation space by learning transfer components across domains using MMD. Tzeng et al. [43]

introduce a CNN with an adaptation layer and an optimization function based on domain confusion loss based on MMD to learn domain invariant feature space. Long et al. [44] propose a variant of MMD with multiple kernels and with task-specific features adapted in the layers of the neural network.

Chapter 3

Crowd Counting Using

Scale-Aware Attention Networks

In recent years, attention models have shown great success in various computer vision tasks [45; 46]. Instead of extracting features from the entire image, the attention mechanism allows models to focus on the most relevant features as needed. Our model is partly inspired by some work of Chen et al. [47] in semantic segmentation. This work uses attentions to focus on the relevant scale dimension of an image for semantic segmentation. In this paper, we use a similar idea and develop a CNN architecture with scale-aware attentions for crowd density estimation and counting. The attention in our model plays a similar role to the “switch” (i.e. density classifier) in Switch-CNN [8]. Switch-CNN makes a hard decision by selecting a particular scale based on the density classifier output and only uses the features corresponding to that scale for the final prediction. The problem is that if the density classifier is not completely accurate, it might select the wrong scale and lead to incorrect density

estimation in the end. In contrast, the attention in our model acts as a “soft switch”. Instead of selecting a particular scale, we re-weight the features of a particular scale based on the attention score corresponding to that scale.

3.1 Contributions

Our contributions include:

- The contributions of our first work are three-fold. First, we introduce the attention mechanism in CNN-based crowd counting models. Although attention models have been successful in many other vision tasks, our work is the first to use attention models in crowd counting. Second, previous work usually uses attention models to focus on certain spatial locations in an image. In our work, we instead use attentions to focus on certain scales. Previous methods [1; 8; 13] in crowd counting select the scale using a learned classifier. In contrast, the attention mechanism allows our model to “softly” select the scales. Finally, we demonstrate that our proposed approach outperforms other state-of-the-art methods on several benchmark datasets.

3.2 Our Approach

Crowd densities can vary dramatically both across different images and across different spatial locations within the same image. We propose to use both global and local attention weights to capture inter-image and intra-image variations of crowd density. This allows our model to adaptively use features at appropriate scales. Our

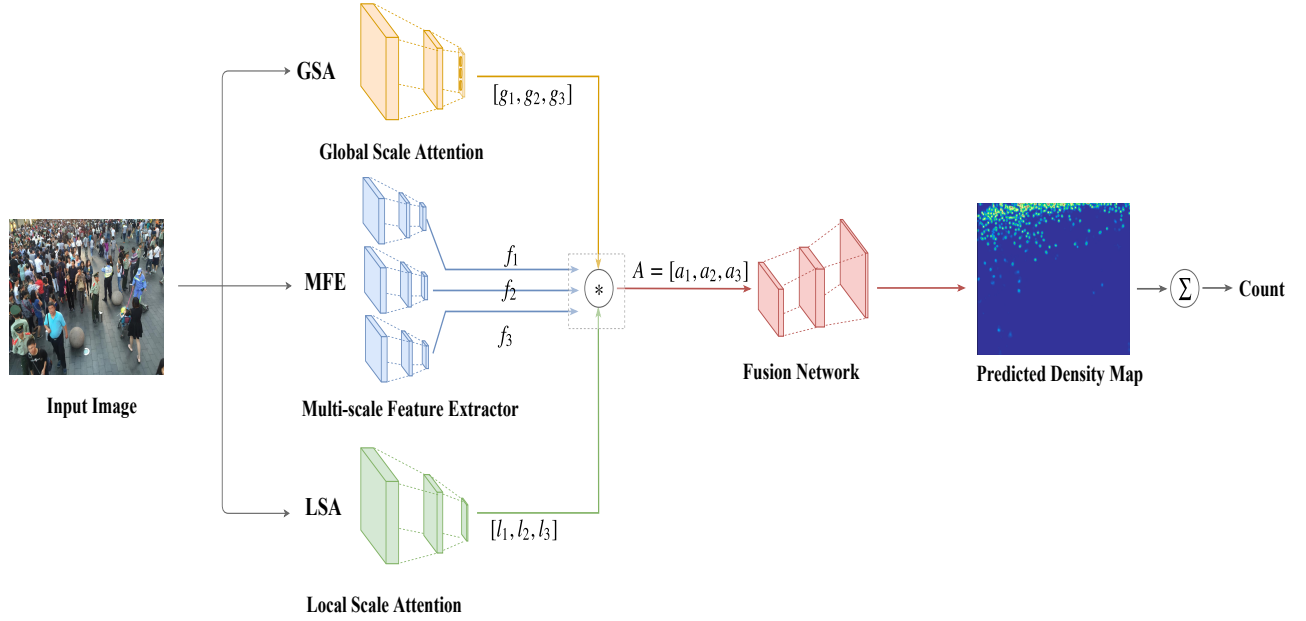


Figure 3.1: Overview of the proposed method. The input image is passed simultaneously through three sub-networks: multi-scale feature extractor (MFE), global scale attentions (GSA), and local scale attentions (LSA). MFE extracts feature maps in three different scales. GSA and LSA produce three global scores and three pixel-wise local attention maps, respectively. Multi-scale feature maps are then weighted by the corresponding GSA and LSA outputs. The attention-weighted features are then used as the input to the fusion network to predict the density map. Finally, the crowd count can be obtained by summing over the entries in the predicted density map.

proposed approach has several modules (see Fig 3.1): multi-scale feature extractor (MFE), global scale attentions (GSA), local scale attention (LSA), and the fusion network (FN) for density estimation. Following, we describe each module in details.

3.2.1 Multi-Scale Feature Extractor

The goal of this module is to extract multi-scale feature maps from an input image. Inspired by the success of using multi-branch architectures in crowd-counting [1], we use a similar multi-branch architecture to extract feature maps at different scales. The

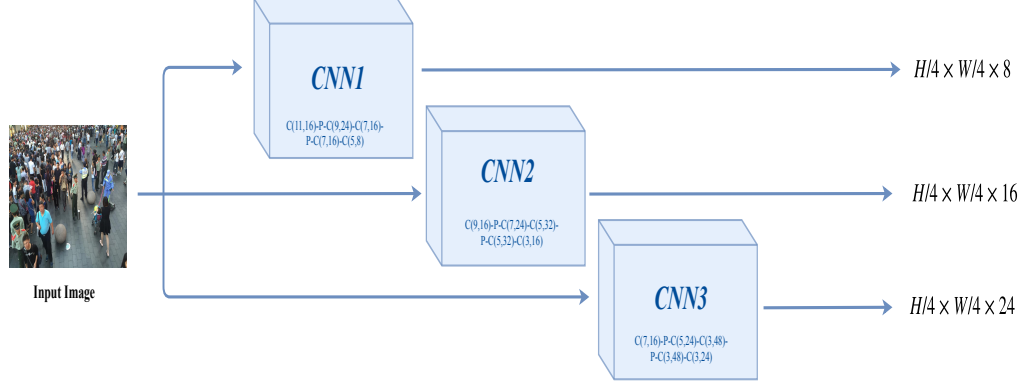


Figure 3.2: Illustration of the multi-branch feature extractor (MFE) module. Here we use three branches associated with three different scales. Each branch uses different filter sizes to extract features at different scales. Using max-pooling, the output of each branch has a spatial dimension equal to one-fourth of the input image size. In this figure, we use $C(i, j)$ to denote a 2D convolution layer with filter sizes of $i \times i$ and j output feature maps. We use P to denote max-pooling.

architecture (see Fig. 3.2) has three branches associated with three different scales. It takes an image of arbitrary size as its input. Each branch then independently processes the input at its corresponding scale level. Each branch consists of multiple blocks of convolution layers with different filter sizes. By choosing different filter sizes in these branches, we can change the receptive field in each branch and capture the features at different scales. Each branch also has several max-pooling layers. We choose the filter sizes in max-pooling so that the output feature map in each branch has a spatial dimension equal to one-fourth of the input image dimension.

3.2.2 Global Scale Attention

Previous work [8; 13; 20] has shown the benefit of leveraging global scale information in crowd counting problem. In our model, we use a global scale attention (GSA) module to capture global contextual information about how dense the image is. This

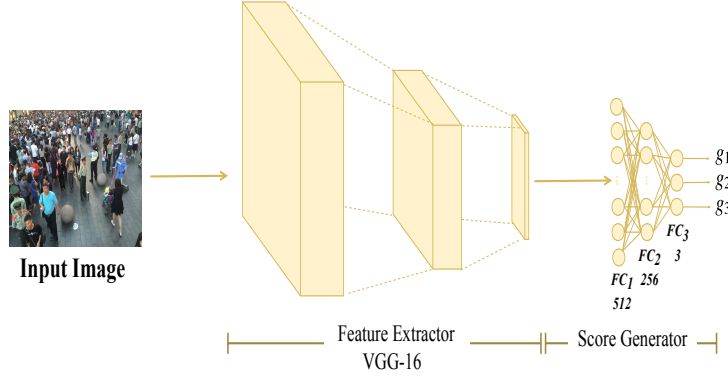


Figure 3.3: Illustration of the global scale attention (GSA) module. Given an input image, this module generates three attention scores. Each score corresponds to one of the three pre-defined density labels: low-density, mid-density, and high-density.

module takes an input image and produces three attention scores. Each score corresponds to one of the three pre-defined density label level: low-density, mid-density, and high-density. The number of density levels is equal to the number of scales in the multi-scale feature extraction module (Sec. 3.2.1). The architecture of GSA is illustrated in Fig. 3.3.

GSA outputs three scores g_i ($i = 1, 2, 3$) for each input image, representing the extent to which the input image belongs to each of the three density labels. A softmax layer is used at the end of the pipeline to normalize the scores (which can be interpreted as “*attentions*”) to sum to one.

3.2.3 Local Scale Attention

The GSA module captures the overall density level of an image. But an image may have different density levels at different locations. The global density level may not sufficient to capture the fine-grained local contextual information at different locations in an image. Inspired by [13], we incorporate a *local scale attention (LSA)* module to



Figure 3.4: Local Scale Attention Network. This module produces three feature maps of size $H/4 \times W/4$ each of which associated to a density scale. $C(i, j)$ denotes a 2D convolution with a filter size of $i \times i$ and with j output feature maps. P stands for max pooling.

capture the local scale information at different locations in an image. LSA generates pixel-wise attention maps representing the scale information at different locations. Similar to the global attention module, here we also consider three different scale levels. Unlike GSA that produces three scalar scores, LSA will produce three pixel-wise attention maps. These attention maps have the same spatial dimensions as the corresponding multi-scale feature maps (Sec. 3.2.1).

The LSA module consists of eight convolution and two max-pooling layers, followed by three fully connected layers (see Fig. 3.4). A sigmoid layer is placed at the end of the module to ensure the values in the attention maps are between 0 to 1.

3.2.4 Fusion Network

The last component of the proposed method is the fusion network (FN) which produces the final density map for an input image. This module takes the extracted feature maps from the image which are re-weighted by the global and local attention scores. The output of this component is a predicted density map. The final crowd count can be obtained by summing over all entries of the estimated density map.

Let $f_i \in \mathbb{R}^{H_i \times W_i \times D_i}$ ($i = 1, 2, 3$) denote the feature maps corresponding to the

three different scales, where $H_i \times W_i$ is the spatial dimension and D_i is the number of channels for the i^{th} feature map. We use $g_i \in \mathbb{R}$ ($i = 1, 2, 3$) and $l_i \in \mathbb{R}^{H_i \times W_i}$ ($i = 1, 2, 3$) to denote the corresponding global and local attention scores, respectively. We can use the global and local attention scores to re-weight the feature maps. Let $f_i^{h,w,d}$ denote the (h, w, d) entry of the feature map corresponding to the i -th scale. Similarly, let $l_i^{h,w}$ denote the (h, w) entry of the corresponding local attention map l_i . We use $a_i \in \mathbb{R}^{H_i \times W_i \times D_i}$ to denote the attention-weighted feature map for the i -th scale. The (h, w, d) entry $a_i^{h,w,d}$ of a_i is calculated as follows:

$$a_i^{h,w,d} = g_i \cdot l_i^{h,w} \cdot f_i^{h,w,d}, \quad \text{where } i = 1, 2, 3 \quad (3.1)$$

As shown in Fig. 3.2 and described in Sec. 3.2.1, the extracted feature maps for all three scales are of size $H/4 \times W/4$ where $H \times W$ is the spatial dimension of the input image. However, the feature maps can have different depth (i.e. number of channels) among different scales. Here the depth dimension is set to 24, 16 and 8 for the low-density, mid-density, and high-density scales, respectively.

The attention-weight feature maps for different scales are then concatenated together and fed into the fusion network (FN) module to produce the density map. The FN module consists of several convolution layers along with 2 de-convolution layers which resize the feature maps into size of $H/2 \times W/2$ and eventually $H \times W$, where $H \times W$ is the spatial dimensions of the original input image. This output of this stage is a feature map of size $H \times W \times 16$. Finally, we apply a 1×1 convolution to produce the density map (DM) of size $H \times W$ (i.e. of depth 1). To obtain the final crowd count from the 2-D density map DM , we simply sum over the entries in the

density map as follows:

$$Count = \sum_{j=1}^H \sum_{k=1}^W DM(j, k) \quad (3.2)$$

where $DM(j, k)$ is the value of at the spatial position (j, k) in the predicted density map.

3.2.5 Loss Function

In order to train the model parameters, we define an overall loss function L_{final} consisting of three losses:

$$L_{final} = L_{DM} + \lambda_g \cdot L_{GSA} + \lambda_l \cdot L_{LSA} \quad (3.3)$$

In Eq. 4.4, L_{DM} is a loss function defined on the predicted density map. This loss will encourage the model to predict density maps close to the ground-truth density maps on the training data. We also use two auxiliary losses L_{GSA} and L_{LSA} . These two losses will encourage the predicted global and local density scale attentions to be similar to the ground-truth global and local scales, respectively. The hyperparameters λ_g and λ_l are used to control the relative contributions of the two auxiliary losses. In the following, we provide details of these loss functions. To simply the notation, we focus on the definition of each loss function on one single training image. The final loss will be accumulated over all training images in the end.

Let $C \in \mathbb{R}^{H \times W}$ be the predicted density map on a training image and $C_{gt} \in \mathbb{R}^{H \times W}$ be the corresponding ground-truth density map. Here $H \times W$ is the spatial dimension of the input image. The loss L_{DM} is defined as the square of the Frobenius norm between C and C_{gt} , i.e. $L_{DM} = \frac{1}{2} \|\text{vec}(C) - \text{vec}(C_{gt})\|^2$ where $\text{vec}(\cdot)$ concatenates entries of a matrix into a vector.

The auxiliary loss L_{GSA} is used to encourage the predicted global scale attention scores to be close to the ground-truth global scale on a training image. We obtain the ground-truth global scale as follows. First, we find the maximum and minimum crowd count (denoted as $Count_{max}$ and $Count_{min}$, respectively) on the training data. We split the range $[Count_{min}, Count_{max}]$ into three bins of equal sizes. For a training image, we assign its ground-truth global scale g_{gt} ($g_{gt} \in \{1, 2, 3\}$) according to the bin that the ground-truth crowd count falls into. Let $g \in \mathbb{R}^3$ be a vector of the global attention scores (corresponding to 3 different scales) on this training image. We can consider g to be the score of classifying the image into one of the three global scales. We define L_{GSA} using the standard cross-entropy loss as $L_{GSA} = CE(g, g_{gt})$ where $CE(\cdot)$ denotes the multi-class cross-entropy loss function.

The loss L_{LSA} is used to encourage the predicted local scale at each spatial location to be consistent with the ground-truth local scale on a training image. We generate the ground-truth local scale as follows. For a pixel location in a training image, we obtain a local crowd count at this location by summing over the ground-truth density map in the 64×64 neighborhood of this location. We then find the minimum/maximum of local crowd count on training images. Similarly, we split the range to three bins and assign the ground-truth scale at a pixel location according to the bin that the local crowd count (over 64×64 neighborhood) falls into. Let $l_{gt} \in \mathbb{R}^{H \times W}$ be a matrix of ground-truth local scales of a training image, where $H \times W$ denotes the spatial dimension. Each entry l_{gt} ($l_{gt} \in \{1, 2, 3\}$) indicates the ground-truth local scale at the corresponding spatial location. Let $l \in \mathbb{R}^{H \times W \times 3}$ denote a tensor of predicted local scale attentions. We define L_{LSA} as the sum of cross-entropy

losses over all spatial locations, i.e. $L_{LSA} = \sum_{h=1}^H \sum_{w=1}^W CE(l[h, w, :], l_{gt}[h, w])$ using Matlab notations.

Empirically, we have found that the auxiliary losses help regularizing the model and improve the performance. In the experiments, we will provide ablation analysis on the impact of these auxiliary losses.

3.3 Experiments

We first introduce the datasets and our experimental setup. We then present experimental results on three benchmark datasets. Finally, we perform ablation studies to further analyze our proposed approach.

3.3.1 Datasets and Setup

Datasets: We evaluate our proposed method on three benchmark datasets: ShanghaiTech PartB [1], Mall dataset [2] and UCF_CC_50 [6]. Table 5.1 shows various statistics of these datasets. The ShanghaiTech PartB dataset contains 716 where 400 images are for training and the other 316 ones for testing. The Mall dataset [2] has 2000 frames captured from a shopping mall. The first 800 frames are used as training frames and the remaining 1200 frames are used for testing. The UCF_CC_50 dataset [6] contains a total of 50 images from web sources. Clearly, the limited number of images in these datasets raises the need for data augmentation in order to prepare the data for training a deep network. Consequently, we used data augmentation to address the issue. We followed the same data augmentation technique used in existing methods [5; 6; 1; 8; 13].

Dataset	Resolution	Images	Max	Min	Avg	Total
ShanghaiTech PartB[1]	1024×768	716	578	9	123.6	88,488
Mall[2]	320×640	2000	53	13	33	62325
UCF_CC_50[6]	varies	50	4543	94	1279.5	63,974

Table 3.1: Statistics of the three datasets used in the experiments. For each dataset, we show the following information: the image resolution, the number of images, the maximum/minimum number of people in an image, the average and total number of people annotated in the dataset.

Evaluation Metric: Following previous work [18; 1; 8; 13; 20], we use Mean Absolute Error (MAE) and Mean Square Error (MSE) as the evaluation metrics. Let N be the number of test images, $\text{Count}_{gt}^{(n)}$ be the ground truth count and $\text{Count}^{(n)}$ be the predicted count for the n -th test image. These two evaluation metrics are defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{n=1}^N |\text{Count}^{(n)} - \text{Count}_{gt}^{(n)}| \quad (3.4)$$

$$\text{MSE} = \sqrt{\frac{1}{N} \sum_{n=1}^N |\text{Count}^{(n)} - \text{Count}_{gt}^{(n)}|^2} \quad (3.5)$$

Ground-truth Density Map: On each dataset, head annotations (i.e. center of the head of a person) are provided as points. Following [5], we generate the ground-truth density map from these point annotations by applying a Gaussian kernel normalized to have a sum of one.

Training Details: We follow the data augmentation technique used in previous methods [5; 6; 1; 8; 13]. Since there are lots of parameters in our model, directly learning all parameters from scratch is challenging. In our implementation, we use

a two-phase training scheme to train our proposed model. During the first phase, we ignore the LSA module and only learn the parameters of the GSA, MFE, and FN modules. We assign each training image to one of three global scale classes (namely, low-density, mid-density, high-density) according to its ground-truth density map. We then learn the parameters of GSA, MFE, and FN modules by optimizing $L_{DM} + \lambda_g \cdot L_{GSA}$. During the second phase, we train all the modules (including LSA) together. The parameters of the GSA, MFE, and FN modules are initialized with the parameters obtained from the first phase. The parameters of the LSA module are initialized randomly from scratch.

3.3.2 Experimental Results

On the ShanghaiTech PartB dataset and the Mall dataset, we follow the standard training/testing split used in previous work [5]. On the UCF_CC_50 dataset, we follow [13] and perform 5-fold cross-validation. We apply the same data augmentation used in previous methods [5; 6; 1; 8; 13] on all datasets.

The experimental results on these three datasets are shown in Tables 3.2, Table 3.3 and Table 3.4, respectively. We also compare with existing state-of-the-art results on each of the datasets. On the ShanghaiTech PartB dataset and the Mall dataset, our proposed model significantly outperforms previous approaches in terms of both MAE and MSE. On the UCF_CC_50 dataset, our model outperforms previous approaches in terms of MAE. In terms of MSE, our model performs better than most previous approaches except for [13].

Method	ShanghaiTech Part B	
	MAE	MSE
R-FCN [†] [48]	52.35	70.12
Faster R-CNN [†] [49]	44.51	53.22
Cross-Scene [18]	32.00	49.80
MC-CNN [1]	26.40	41.30
Switching-CNN [8]	21.60	33.40
CP-CNN [13]	20.1	30.1
FCN [50]	23.76	33.12
DecideNet [5]	20.75	29.42
Ours	16.86	28.41

Table 3.2: Comparison of the performance of different methods on the ShanghaiTech PartB dataset [1]. [†]These results are obtained from [5].

3.3.3 Ablation Study

We conduct ablation study on the ShanghaiTech PartB dataset to provide further analysis of relative contributions of various components of our approach.

Impact of attention modules: First, we analyze the relative contributions of global and local attentions in our model. In Table 3.5, we show the results of removing the global or local attention modules (GSA or LSA) in our model. The base model in Table 3.5 refers the architecture with only the multi-scale feature extractor and the fusion network, i.e. without any attention modules. From the results, we can see that both GSA and LSA contribute to the final performance. Using both GSA and LSA with the base model, we achieve the best performance.

Method	Mall	
	MAE	MSE
DecideNet [5]	1.52	1.90
R-FCN [†] [48]	6.02	5.46
Faster R-CNN [†] [49]	5.91	6.60
SquareChn Detector [†] [51]	20.55	439.10
Count-Forest [52]	4.40	2.40
Exemplary-Density [†] [53]	1.82	2.74
Boosting-CNN [17]	2.01	–
Mo-CNN [54]	2.75	13.40
Weighted_VLAD [55]	2.41	9.12
Ours	1.28	1.68

Table 3.3: Comparison of the performance of different methods on the Mall dataset [2].

[†]These results are obtained from [5].

Impact of auxiliary losses: We also study the impact of using the auxiliary losses (Eq. 4.4) as extra supervisions during training. In this analysis, we use our proposed model with all the modules. But during training, we use various combinations of the three losses: L_{DM} , L_{GSA} and L_{LSA} . The results are shown in Table 3.6. We can see that both L_{GSA} and L_{LSA} help improving the performance of the learned model. By using all three losses, the best performance is achieved using all three losses during training.

Method	UCF_CC_50	
	MAE	MSE
Cross-Scene [18]	467.00	498.50
MC-CNN [1]	377.60	509.10
Switching-CNN [8]	318.10	439.20
CP-CNN [13]	295.80	320.90
Lempitsky et al. [56]	493.4	487.1
Idrees et al. [6]	419.5	541.6
Crowd-Net [19]	452.5	—
Hydra-2s [10]	333.73	425.26
Ours	271.60	391.00

Table 3.4: Comparison of the performance of different methods on the UCF_CC_50 dataset [6].

Methods	MAE	MSE
base model	27.63	46.65
base model+GSA	17.0	30.60
base model+LSA	18.07	31.87
base model+GSA+LSA (this paper)	16.86	28.41

Table 3.5: Effect of GSA and LSA modules on the ShanghaiTech PartB dataset. The “base model” only contains the multi-scale feature extractor and the fusion network, i.e. without any global or local attention modules. By adding GSA or LSA, we can achieve better performance. The best performance is obtained by using both GSA and LSA together with the base model.



Figure 3.5: Qualitative examples of density maps. The first two columns are original input images from the ShanghaiTech partB [1] dataset and their corresponding density maps respectively. The last two columns are original input images from the Mall [2] dataset and their corresponding density maps respectively.

Methods	MAE	MSE
L_{DM}	19.15	35.55
$L_{DM} + L_{LSA}$	17.02	31.49
$L_{DM} + L_{GSA}$	17.33	32.33
$L_{DM} + L_{LSA} + L_{GSA}$ (this paper)	16.86	28.41

Table 3.6: Effect of using auxiliary losses as extra supervisions on the ShanghaiTech PartB dataset. L_{DM} is the model without any auxiliary losses. By adding L_{LSA} (i.e. $L_{DM} + L_{LSA}$) or L_{GSA} (i.e. $L_{DM} + L_{GSA}$), we can improve the performance. By using both L_{LSA} and L_{GSA} as auxiliary losses (i.e. $L_{DM} + L_{LSA} + L_{GSA}$), we achieve the best performance.

Chapter 4

One-shot Scene Specific Crowd Counting Using Domain Adaptation

One popular approach for one-shot learning problem in computer vision is to use fine-tuning. We can first learn a crowd counting model using all the training data under the standard supervised setting. Given a labeled image in the target scene, we keep the model parameters in early layers of the network (called the encoder in this paper) fixed and fine-tune the last few layers in the model (called the decoder in this paper) based on this labeled image. This fine-tuning has been used for one-shot object segmentation in videos [26]. We call this approach simple fine-tuning. The limitation of this simple fine-tuning approach is that the learning of the model parameters is disconnected from the testing. In other words, the encoder parameters are learned to achieve good performance in the standard supervised setting, they are

not explicitly learned for the purpose of facilitating fine-tuning to the target scene. In this paper, we propose a novel one-shot learning approach using finetuning. Different from the simple fine-tuning, our learning algorithm is specifically designed for the goal of one-shot adaptation to a new scene.

4.1 Contributions

Our contributions include:

- The contributions of our second work are manifold. First, we introduce a new problem called the one-shot scene-specific crowd counting. This problem setting is closer to how crowd counting algorithms are used in the real world. A reliable solution to this problem will help the wide adoption of crowd counting algorithms in real-world applications. Second, we propose a novel one-shot learning algorithm for this new problem. The novelty of our approach is that it is specifically designed to facilitate fine-tuning in target scenes. Finally, since this is a new problem and there are no existing datasets specifically created for it, we have amassed two benchmark datasets for this problem by re-purposing existing crowd counting datasets. On these benchmark datasets, our proposed approach significantly outperforms several baseline methods.

4.2 Approach

We first describe the backbone architecture used in our model. Then we give details on how we fine-tune the model on a single labeled example from a target scene

to get a scene-specific model in latter section. Finally, we describe how to learn the model during training using labeled images collected from multiple scenes.

4.2.1 Model Architecture

In this paper, we use the multi-column CNN (MCNN) architecture in [1] as our backbone network, although our proposed approach is agnostic to the specific choice of network architectures. MCNN has three branches to extract features at different scales (see Fig. 4.1). Each branch processes the input image separately using filters of different sizes. In the end, the features from these branches are combined together to predict a density map that has the same spatial size of the input image. The crowd count can be obtained by summing over the entries of the predicted density map. Readers are referred to [1] for details of this network architecture.

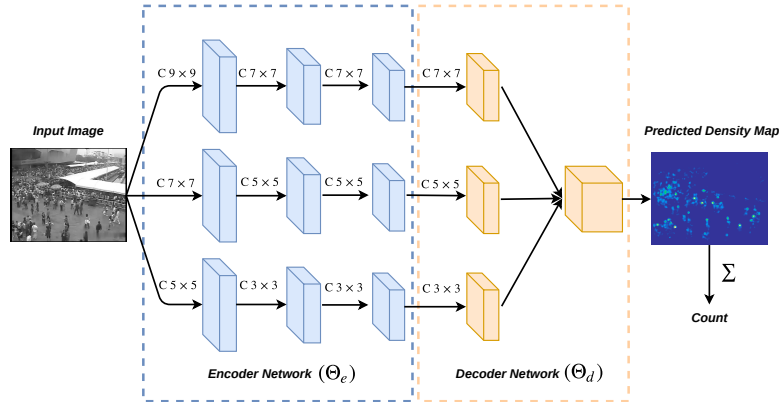


Figure 4.1: Network architecture of our backbone network based on the multi-column CNN (MCNN) [1].

With a little abuse of terminology, we refer to first part of the network architecture in Fig. 4.1 as the *encoder network* and the remaining part of the network architecture as the *decoder network*. The encoder network is used to extract features from an

image, while the decoder network is used to produce the output from the extracted features. Note that the split of encoder/decoder in the network is flexible. In our experiments, we have considered either the last layer or the last two layers of MCNN as the decoder. We consider the remaining layers as the encoder. We use Θ_e and Θ_d to denote the model parameters corresponding to the encoder and the decoder networks, respectively. The model parameters Θ of the entire MCNN are the combination of Θ_e and Θ_d , i.e. $\Theta = \{\Theta_e, \Theta_d\}$. We use $F(x; \{\Theta_e, \Theta_d\})$ to denote the output density map of MCNN for an input image x . In the standard supervised setting, the model parameters Θ are obtained by optimizing the loss function defined on a collection of training images.

4.2.2 One-Shot Scene-Specific Adaptation

During testing, we are given a collection of M images $\{x_i\}_{i=1}^M$ from the *same* scene. For example, these images might be captured by the same camera fixed at a particular location. We also have the ground-truth annotation of one “test” image for this scene. Here we abuse the terminology a bit – this image is technically not a test image since it has the ground-truth annotation. Without loss of generality, we assume that x_1 has the ground-truth annotation (i.e. density map) y_1 .

During the inference, we would like to adapt the model parameters Θ to this specific scene. A popular approach for this adaption is by fine-tuning the model parameters on (x_1, y_1) . This strategy has been successfully used in one-shot video object segmentation (OSVOS) [26]. In OSVOS, we are given a video for which the segmentation mask of the object of interest is provided on the first frame. OSVOS then

fine-tunes the learned segmentation network on the first frame to obtain a segmentation network specifically tuned to this video. Finally, the fine-tuned segmentation network is used to segment the object in the remaining frames. In our work, we use a similar idea to fine-tune a crowd counting network on one single labeled example (x_1, y_1) . As a result, the network will be adapted to this specific scene. However, since we have only one labeled example for this scene, fine-tuning the entire model parameters Θ will tend to overfit to this single example.

In our work, we choose to fix the parameters of the encoder network Θ_e and only fine-tune the parameters of the decoder network Θ_d . As mentioned previously, the split of encoder/decoder is somewhat arbitrary and depends on the application scenario. As a rule of thumb, if we only have very few labeled examples for a specific scene, we should choose a lightweight decoder with few layers. If we have more labeled examples, we can choose the decoder to have more layers.

Given an image x and its ground-truth density map y (note that y is usually represented as a matrix), we use the L_2 distance between the predicted density map $F(x; \{\Theta_e, \Theta_d\})$ and ground-truth density map y as the loss function. So the fine-tuning process is equivalent to solving the following optimization problem:

$$\Theta_d^* = \arg \min_{\Theta_d} L_d(\Theta_d) \quad (4.1)$$

where $L_d(\Theta_d) = \|\text{vec}(F(x_1; \{\Theta_e, \Theta_d\})) - \text{vec}(y_1)\|_2$ and $\text{vec}(\cdot)$ convert an input matrix to a vector. Note that the loss $L_d(\Theta_d; (x_1, y_1))$ in Eq. 4.1 is a function of Θ_d since we are only fine-tuning the decoder network. The parameters Θ_e in the encoder network are fixed during the inference and can be treated as constants. Also note that $L_d(\Theta_d)$ is defined on (x_1, y_1) which is the single labeled example for the specific scene. Fine-

tuning according to Eq. 4.1 involves computing the gradient of $L_d(\Theta_d)$ with respect to Θ_d using back-propagation and updating Θ_d using gradient descent.

After fine-tuning on the first image of the target scene, we obtain updated decoder parameters Θ_d^* . For any other test image x_i (where $i = 2, \dots, M$) from the target scene, we predict its density map as $F(x_i; \{\Theta_e, \Theta_d^*\})$. Note that the index i starts from 2 (instead of 1) since we assume that x_1 comes with its ground-truth annotation y_1 , so we do not need to make the prediction for x_1 .

In our formulation, the encoder parameters Θ_e are generic to all scenes. In contrast, the decoder parameters Θ_d are specific to a particular scene. By fine-tuning Θ_d on a labeled example from the target scene, we effectively tune our model to adapt to the target scene.

4.2.3 Model Learning

During training, we have a collection of labeled training images from T scenes. For example, each scene might correspond to a camera fixed at a particular location. For the t -th scene ($t = 1, 2, \dots, T$), we use $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^N$ to denote the training images and the corresponding ground-truth density maps from that scene. To simplify the notation, we have assumed that each scene has the same number of N training images. But our proposed algorithm can be easily generalized to the case where different scenes have different numbers of training images.

Our goal of the training phase is to learn the parameters Θ_e of the encoder network. Note that technically speaking, we do not need to learn the decoder parameters Θ_d during the training phase. If we have the encoder parameters Θ_e , the decoder pa-

parameters Θ_d will be obtained via fine-tuning (see Eq. 4.1) on a single labeled example from the target scene. Ideally we would like the learned Θ_e to have the following property. Given a target scene t , suppose we use the learned Θ_d^* in Eq. 4.1 by fine-tuning on $(x_1^{(t)}, y_1^{(t)})$, we would like MCNN with model parameters $\{\Theta_e, \Theta_d^*\}$ to perform well on the remaining training images in this scene. In other words, $F(x_i^{(t)}; \{\Theta_e, \Theta_d^*\})$ should be close to $y_i^{(t)}$ (where $i = 2, 3, \dots, N$). Note that we do not necessarily need to enforce $F(x; \{\Theta_e, \Theta_d^*\})$ to perform well if x is an image from a different scene. This is a reasonable learning objective since it mimics what happens during testing. If the learned encoder parameters Θ_e have the above property, when we fine-tune the decoder parameters Θ_d on a single labeled example in a target scene, the fine-tuned parameters will likely work well on other images from this target scene.

For a fixed Θ_e , we use $\Theta_d^{(t)}$ to denote the decoder parameters for the t -th scene obtained by fine-tuning on $(x_1^{(t)}, y_1^{(t)})$ according to Eq. 4.2:

$$\Theta_d^{(t)} = \arg \min_{\Theta_d} L_d^{(t)}(\Theta_d), \quad (4.2)$$

where $L_d^{(t)}(\Theta_d) = \|\text{vec}(F(x_1^{(t)}; \{\Theta_e, \Theta_d\})) - \text{vec}(y_1^{(t)})\|_2$. For simplicity, let us ignore the randomness (e.g. due to random initialization and local minimum) in the specific optimization algorithm used for solving Eq. 4.2 and consider that the solution of $\Theta_d^{(t)}$ in Eq. 4.2 is unique for a fixed Θ_e . Then technically speaking, $\Theta_d^{(t)}$ is a function of Θ_e . If we vary the value of Θ_e , Eq. 4.2 will give a different solution of $\Theta_d^{(t)}$. To make this relationship more explicit, we write the decoder parameters as $\Theta_d^{(t)}(\Theta_e)$ to emphasize that the decoder parameters can be treated as a function of the encoder parameters.

Now we introduce the loss function for the learning. We first focus on the loss function on the t -th scene. Note that we would like $\{\Theta_e, \Theta_d^{(t)}(\Theta_e)\}$ to perform well on

the remaining training images $\{(x_i^{(t)}, y_i^{(t)})\}_{i=2}^N$ in this scene, so a natural choice of the loss is as follows:

$$L_e^{(t)}(\Theta_e) = \sum_{i=2}^N \|\text{vec}(F(x_i^{(t)}; \{\Theta_e, \Theta_d^{(t)}(\Theta_e)\})) - \text{vec}(y_i^{(t)})\|_2 \quad (4.3)$$

Note that the loss $L_e^{(t)}(\Theta_e)$ is a function of only the encoder parameters Θ_e , since the decoder parameters $\Theta_d^{(t)}(\Theta_e)$ become deterministic once we have Θ_e .

The final loss is simply the summation of $L_e^{(t)}(\Theta_e)$ over all scenes as follows:

$$L_{final}(\Theta_e) = \sum_{t=1}^T L_e^{(t)}(\Theta_e) \quad (4.4)$$

Note that $L_{final}(\Theta_e)$ is a function of the encoder parameters Θ_e . In theory, we can just take the gradient of $L_{final}(\Theta_e)$ with respect to Θ_e and use gradient descent to optimize Eq. 4.4. However, this is difficult in practice since there is another optimization problem (see Eq. 4.2) nested in the definition of the loss $L_{final}(\Theta_e)$ in Eq. 4.4. So it is non-trivial to directly calculate the gradient of $L_{final}(\Theta_e)$ with respect to Θ_e .

Our key insight is that if we fix the decoder parameters $\Theta_d^{(t)}$ (where $t = 1, 2, \dots, T$) for all scenes, Eq. 4.4 becomes a simple function that can be easily differentiated. Based on this observation, we propose to optimize Eq. 4.4 using a coordinate descent style approach that alternates between the following two steps:

- Fix Θ_e , solve for $\Theta_d^{(t)}$ (where $t = 1, 2, \dots, T$) using Eq. 4.2 for each scene;
- Fix $\Theta_d^{(t)}$ (where $t = 1, 2, \dots, T$), solve for Θ_e by minimizing Eq. 4.4. Note that in this case, Θ_e are simply part of the parameters in MCNN and can be optimized using SGD with back-propagation.

These two steps can be repeated until convergence. In practice, we alternate between these two steps for a fixed number of iterations.

Note that our approach is different from the standard fine-tuning strategy used in other one-shot learning methods (e.g. [26]). The fine-tuning method in [26] first learns a full model (both Θ_e and Θ_d) by minimizing the empirical loss on the training data, i.e.

$$\Theta_e^*, \Theta_d^* = \arg \min_{\Theta_e, \Theta_d} \sum_{t=1}^T \sum_{i=1}^N \|\text{vec}(F(x_i^{(t)}; \{\Theta_e, \Theta_d\})) - \text{vec}(y_i^{(t)})\|_2 \quad (4.5)$$

Given a target scene, the encoder parameters Θ_e^* will be fixed. Only the decoder parameters Θ_d will be fine-tuned using Eq. 4.1. The problem of this simple fine-tuning method is that there is a disconnect between learning and testing. During learning, the loss function in Eq. 4.5 does not explicitly enforce Θ_e to be transferable to a new scene. In contrast, our proposed approach learns Θ_e in a specific way, so that when we use Θ_e to get the fine-tuned decoder parameters $\Theta_d^{(t)}$ in a target scene, the final model will perform well.

Randomization of Training Images: So far, we have assumed that the first example $(x_1^{(t)}, y_1^{(t)})$ in the t -th scene is used for fine-tuning. For testing, this is reasonable since this example corresponds to the only labeled example available in the target scene. But for training, this creates asymmetry among training data since the first training example in the t -th scene plays a different role from other training examples in this scene. In order to break this asymmetry, we randomly pick a training image from the t -th scene as $(x_1^{(t)}, y_1^{(t)})$ by shuffling the images in each scene at the beginning of every training epoch. This will make sure that all training examples in the t -th scene have the chance of being used for fine-tuning.

4.3 Experiments

In this section, we first define several baseline methods for comparative studies. We then introduce the datasets and the experiment setup. Finally, we present both quantitative and qualitative experimental results.

4.3.1 Baselines

Since this paper is the first work on one-shot scene-specific crowd counting, there is no previous work that we can directly compare with. Nevertheless, we define several of our own baseline methods as follows.

No fine-tuning: This baseline is the MCNN model [1] trained in a standard supervised setting. The model parameters are trained from all images in the training set according to Eq. 4.5. Once the training is done, the model is fixed and is applied to all test images. Note that the original MCNN [1] uses the perspective maps and ground-truth ROI regions to improve the final performance. For simplicity, we do not use these extra components.

Simple fine-tuning: This baseline first learns $\{\Theta_e, \Theta_d\}$ in the standard supervised setting using all the training images (i.e. the “no fine-tuning” baseline). For a given scene during testing, it fixes Θ_e and fine-tunes Θ_d using Eq. 4.1. Note that this baseline is equivalent to the method for one-shot object segmentation in [26].

As mentioned earlier, the split of encoder/decoder networks in our formulation is flexible. We have experimented with using either the last layer or the last two layers of MCNN as the decoder. We refer to them as “simple fine-tuning (1 layer)” and “simple

fine-tuning (2 layers)” for the “simple fine-tuning” baseline, respectively. Similarly, we have experimented with using either one layer or two layers as the decoder in our proposed models. We refer to them as “ours (1 layer)” and “ours (2 layers)”, respectively.

4.3.2 Datasets and Setup

Datasets: Most existing crowd counting datasets are not created for the problem studied in this paper. In particular, the training and testing images are often collected from the same scene in those datasets. As a result, we cannot use these datasets since our problem setting assumes that images in a dataset come from multiple scenes. To the best of knowledge, there are only two datasets with more than 100 scenes which can be used in our problem setting, namely WorldExpo’10 [4] for counting people and Trancos [3] for counting vehicles. In our experiments, we use these two datasets to create our own benchmark datasets. For cross-dataset testing we consider two more crowd counting datasets, Mall dataset [2] and UCSD dataset [7]. In section 4.3.4, we explain the significance of cross-dataset testing setup and result.

The WorldExpo’10 dataset has a total of 1132 annotated frames collected from 103 different scenes with a fixed resolution of 576×720 . We choose 50 scenes for training and the remaining 53 scenes for testing. For each scene in testing, we choose one image randomly and corresponding ground-truth label as (x_1, y_1) and perform fine-tuning on this image. The test accuracy is calculated on the remaining images in this scene.

The Trancos dataset consists of 1244 images captured from different scenes. We

choose 50 scenes for training and the remaining 20 scenes for testing. Again, for each scene in testing, we choose one image and its ground-truth label for fine-tuning.

Mall dataset contains 2000 images from a single scene with a size of 640×480 . Each image contains the headcount of the people. According to standard split for the Mall dataset first 800 frames are used for training and the remaining 1200 images are used for testing.

UCSD dataset contains 2000 labeled images from a single scene with the image size of 238×158 . The training set contains images from 600 to 1400, which contains 800 frames. The remaining 1200 frames are held-out for testing.

Ground-truth Density Map: All four datasets (WorldExpo'10, Mall, UCSD and Trancos) have ground-truth annotations in the form of points. For the WorldExpo'10, Mall and UCSD dataset, head annotations (i.e. center of the head of a person) are provided as points. For the Trancos dataset, points are the positions of x - y coordinates of the center of the vehicles. Following [1], we construct the ground-truth density map from these point annotations by applying a Gaussian kernel which is normalized to have a sum of one.

Implementation Details: The parameters $\{\Theta_e, \Theta_d\}$ learned using the “no fine-tuning” baseline are used to initialize our model. We use the Adam optimizer [57] with a learning rate of $1e-5$ and a momentum of 0.9. We run 200 epoches when optimizing Eq. 4.2 and Eq. 4.3. The number of iterations for the coordinate descent style alternating algorithm is fixed to be 1000.

Evaluation Metric: Following prior work [4; 1; 58; 13; 20], we use Mean Absolute Error (MAE) and Mean Square Error (MSE) as our evaluation metrics. Let N be the

Method	WorldExpo'10	
	MAE	MSE
no fine-tuning	13.59	16.98
simple fine-tuning (1 layer)	10.56	13.55
ours (1 layer)	9.87	12.83
simple fine-tuning (2 layers)	9.88	12.84
ours (2 layers)	9.63	12.58

Table 4.1: Comparison of the performance (MAE and MSE) of our approach and the baselines on the WorldExpo'10 dataset [4]. For “ours” and “simple fine-tuning”, we consider either using the last layer or the last two layers of MCNN as the decoder.

number of test images. We use c_i to denote the ground truth count of the i -th test image obtained by summing over the ground-truth density map. Similarly, we use $\tilde{c}_i$ to denote the corresponding predicted count obtained by summing over the predicted density map. These two evaluation metrics are defined as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^N |c_i - \tilde{c}_i|, MSE = \sqrt{\frac{1}{N} \sum_{i=1}^N |c_i - \tilde{c}_i|^2} \quad (4.6)$$

Note that in Eq. 4.6, we exclude the “test” image used for the fine-tuning in the t -th test scene, since we already know the ground-truth label of this “test” image.

4.3.3 Main Results

Table 4.1 shows the results of our method and the comparison with the baselines on the WorldExpo'10 dataset. We show results of using either the last layer or the last two layers of MCNN as the decoder. According to the results, we can see that without

Method	Trancos	
	MAE	MSE
no fine-tuning	8.00	8.08
simple fine-tuning (1 layer)	5.12	5.91
ours (1 layer)	3.92	4.78
simple fine-tuning (2 layers)	3.50	4.40
ours (2 layers)	3.38	4.26

Table 4.2: Comparison of the performance (MAE and MSE) of our approach and the baselines on the Trancos dataset [3]. For “ours” and “simple fine-tuning”, we consider either using the last layer or the last two layers of MCNN as the decoder.

Method	WorldExpo’10	
	MAE	MSE
no fine-tuning	22.62	43.70
simple fine-tuning (2 layer)	18.06	39.37
ours (2 layer)	17.60	38.65

Table 4.3: Comparison of the performance (MAE and MSE) of our approach and the baselines on the WorldExpo’10 dataset [4] using the standard split.

fine-tuning during testing, the performance is poor. Simple fine-tuning can improve the performance. Our proposed method (using either one layer or two layers as the decoder) significantly outperforms other baselines. Table 4.2 presents the results on the Trancos dataset. Again, we see a similar trend and our proposed method outperforms the baselines. We show some qualitative examples of the predicted density maps on both datasets in Fig. 4.2.

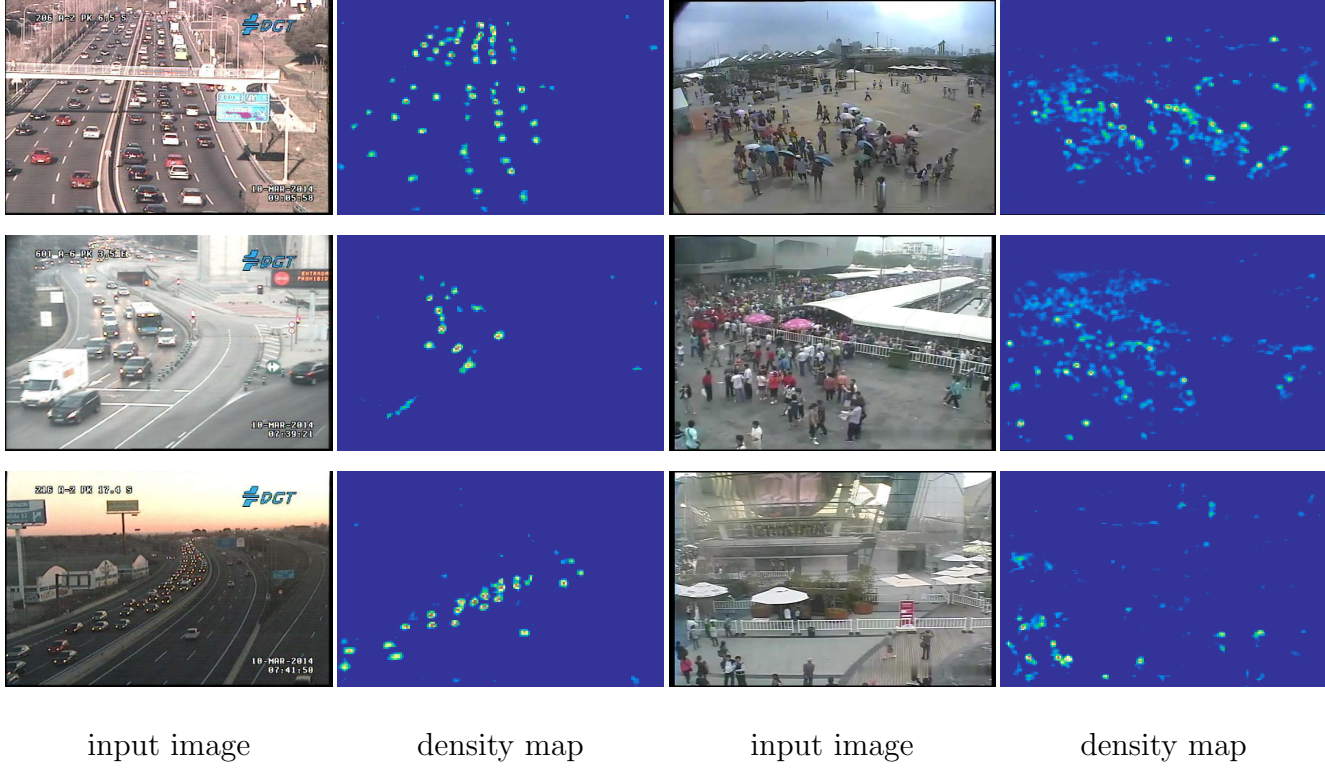


Figure 4.2: Qualitative examples of predicted density maps. The 1st and 3rd columns contain images from the WorldExpo'10 and Trancos datasets, respectively. The 2nd and 4th columns are the corresponding density maps.

Method	UCSD dataset	
	MAE	MSE
no fine-tuning	20.08	21.25
simple fine-tuning (1 layer)	3.29	4.08
ours (1 layer)	3.18	3.98
simple fine-tuning (2 layers)	7.29	9.35
ours (2 layers)	3.12	3.92

Table 4.4: Performance in the cross-dataset testing. Here we use the WorldExpo'10 dataset [4] for training and the UCSD dataset [7] for testing.

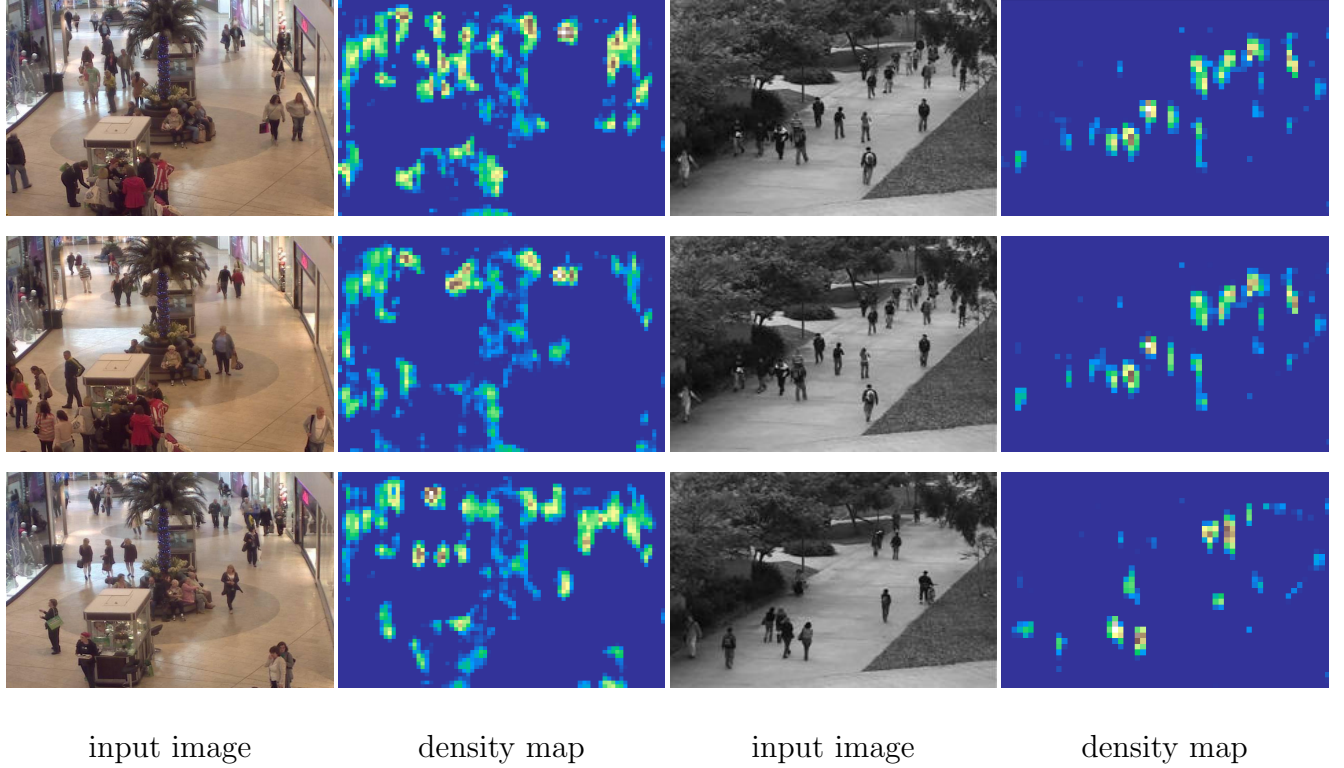


Figure 4.3: Qualitative examples of input images and predicted density maps in the cross-dataset testing setting. The 1st and 3rd columns are images from the Mall and the UCSD datasets, respectively, the 2nd and 4th columns are their corresponding density maps. The Trancos dataset is used for training.

As mentioned earlier, we have decided to use a different training/testing split in our experiments, since this paper studies a new problem and the standard split is not suitable for this problem setting. Nevertheless, we have also tried our method with 2 layers by following the standard split used in the literature as close as possible. Table 4.3 shows the results of using the standard split on WorldExpo’10. In this split, there are 5 test scenes. We can see that our method outperforms the baselines in this standard split setting as well. We like to emphasize that even in this standard split, the results are not directly comparable to previous works. Our problem setting

Method	Mall dataset	
	MAE	MSE
no fine-tuning	3.82	4.85
simple fine-tuning (1 layer)	3.58	4.56
ours (1 layer)	3.16	3.90
simple fine-tuning (2 layers)	3.28	4.10
ours (2 layers)	3.10	3.88

Table 4.5: Performance in the cross-dataset testing. Here we use the WorldExpo’10 dataset [4] for training and the Mall dataset [2] for testing.

assumes that one image in a testing scene is “labeled” and this image has to be excluded when calculating the performance numbers. In other words, there are slightly fewer number of test images used in calculating MAE/MSE in Table 4.3. Also note that MCNN in [1] uses extra components (perspective maps, ROI) to improve the final performance. Due to these differences, we do not include previously reported numbers in Table 4.3 since they are not directly comparable and can be misleading. Also note that the MAE/MSE numbers in Table 3 are in general higher than those in Table 1. This is because in the standard split, the 5 test scenes are totally different the training scenes, e.g. station vs street scenes. For the training/testing split in Table 4.1, a lot of test scenes are similar to training scenes. This domain shift explains the performance difference in Table 4.1 and Table 4.3.

Unfortunately, we could not perform the experiment on Trancos using the standard split, since images from the same scene can be splitted into both training and test sets in the standard split on Trancos. This violates the assumptions of our problem

setting which assumes that training and testing should have non-overlapping scenes.

4.3.4 Cross-Dataset Testing

In real-world applications of crowd counting, the images we will encounter during testing are likely to be completely different from the training data. To demonstrate the generalizability of our model when there is significant domain shift between training and testing, we perform an experiment using cross-dataset testing. First, we train our model on the WorldExpo'10 dataset. We then evaluate our model on the Mall dataset [2] and the UCSD dataset [7] using the standard test data splits on these datasets. We take one test image in the target dataset as for fine-tuning. For simplicity, we do not use the perspective maps and region of interest (ROI) maps. Table 4.4 and Table 4.5 show the results of this cross-dataset testing. Our method outperforms other baseline approaches.

Chapter 5

Domain Adaptation in Crowd Counting

To extend previous works, we study the new problem of domain adaptation in crowd counting and believe that the setup is closer to the real-world applications as our third work. Consider the scenario where a company tries to develop crowd counting systems to be deployed to a user. The company might set up a camera in some location to collect training images and annotate those images and consider these images to be from the *source domain*. Further, consider the images captured by the user’s camera to be from the *target domain*. A successful domain adaptation solution will enable a model to work well in the target domain by taking advantage of the available training images in the source domain. In this paper, we consider three different settings of domain adaptation: 1) *unsupervised domain adaptation*, where the target domain has a large number of unlabeled images; 2) *supervised domain adaptation*, where the target domain contains a small number of labeled images;

3) *semi-supervised domain adaptation*, where the target domain consists of a small number of labeled images and a large number of unlabeled images.

5.1 Contributions

Our contributions include:

- The contributions of our third work can be summarized as follows. First, this is the first paper on domain adaptation in crowd counting. Most existing domain adaptation work focuses on tasks such as image classification, and domain adaptation. To the best of our knowledge, there has not been any work on domain adaptation for crowd counting. The closest work to ours is [14] which learns a model for counting different object types (e.g., people, penguins, cars). Each object type has a domain-specific layer that adjusts to that object. The main goal of [14] is different from ours. Our goal is to adapt a model that works well on a target domain with limited labeled data, while [14] focuses on allowing a model to work well on several object types simultaneously. We believe crowd counting is an area more likely to benefit from domain adaptation since the test images often come from the same camera at a fixed location and angle (i.e., one domain). Moreover, we only care how well the system works in that particular domain. Second, we propose three different domain adaptation settings in crowd counting. These different settings can be used in different application scenarios. Finally, we experimentally demonstrate that our proposed approach outperforms other alternatives. In addition to standard domain adaptation where the object of interest is the same in both domains, we also demonstrate cross-object

domain adaptation. For example, the source domain involves people counting, while the target domain involves car counting.

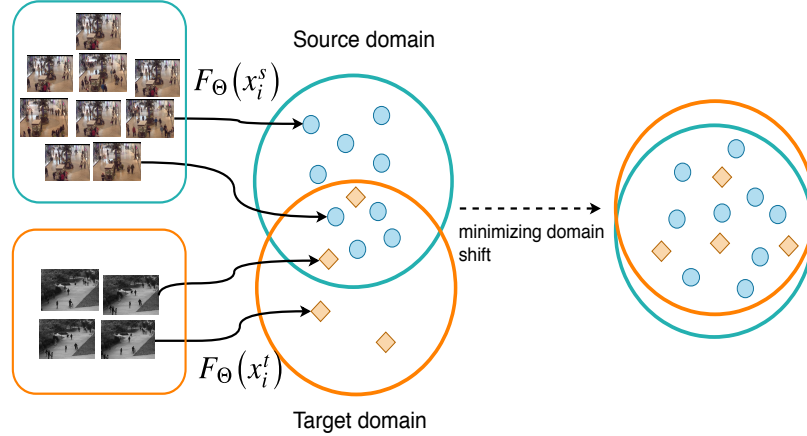


Figure 5.1: Illustration of high-level intuition of domain adaptation. We have a source domain and a target domain. The intuition of our domain adaptation approach is to minimize the domain shift between the feature representations of images from these two domains.

5.2 Our Approach

We firstly introduce the backbone network, then describe our learning mechanism for domain adaptation.

5.2.1 Network Architecture

Our approach can be used with any crowd counting network. The best backbone architecture is not our focus and is beyond the scope of this paper. For simplicity, we use the multi-column convolutional neural network (MCNN) [1] as the backbone architecture in this paper, although our approach is certainly not limited to this par-

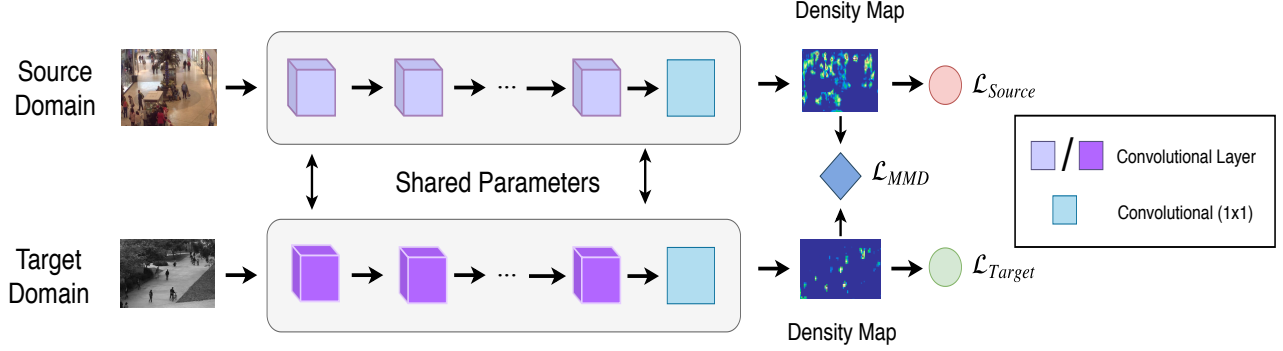


Figure 5.2: The architecture of our backbone network based on the multi-column convolutional neural network (MCNN) [1]. This architecture consists of a parallel branch to extract features at different scales. The features from all three branches are concatenated together in the end to predict the output. The output of the network is an estimated density map of the input image. Taking an integral over the density map gives the final count. In our work, we consider the blue colored network as source and the orange network as the target. We employ maximum mean discrepancy to minimize the domain shift between the source and target feature representations.

ticular choice of the backbone network. MCNN has three branches in the architecture. Each branch will extract features at a particular scale. In the end, the features from these three branches are concatenated together to form a final feature map. This feature map is then used to predict the density map of the input image. The crowd counting can be obtained by integrating over the spatial locations in the predicted density map. The overall architecture of MCNN is shown in Fig. 5.1 and readers are referred to [1] for details of this network architecture.

We use Θ to denote the model parameters of MCNN. For an input image x , we use $F_{\Theta}(x)$ to denote the final feature map before the prediction and $\Psi_{\Theta}(F_{\Theta}(x))$ to represent the predicted density map. In a supervised setting, Θ is learned by optimizing a loss function defined on labeled images.

5.2.2 Domain Adaptation for Crowd Counting

Without loss of generality, we present our domain adaptation approach in the following under the semi-supervised setting (SSDA), since the other two setups (UDA and SDA) are just special cases of SSDA.

In the SSDA setting, we learn the model parameters Θ of MCNN by optimizing the following loss functions:

$$\mathcal{L}_{SSDA}(\Theta) = \mathcal{L}_{source}(\Theta) + \mathcal{L}_{target}(\Theta) + \alpha \mathcal{L}_{mmd}(\Theta) \quad (5.1)$$

where $\mathcal{L}_{source}(\Theta)$ and $\mathcal{L}_{target}(\Theta)$ in Eq. 5.1 are loss functions defined on $\mathcal{D}_s$ and $\mathcal{D}_d^l$, respectively. The last term $\mathcal{L}_{mmd}(\Theta)$ is a loss defined for domain adaptation. The hyperparameter α controls the relative weight of the domain adaptation loss. The details of these loss functions are described in the following.

Supervised Loss: The loss $\mathcal{L}_{source}(\Theta)$ measures how well the model parameters perform on the labeled training images in the source domain $\mathcal{D}_s$. It is defined as:

$$\mathcal{L}_{source}(\Theta) = \frac{1}{N_s} \sum_{i=1}^{N_s} \|\Phi_{\Theta}(F_{\Theta}(x_i^s)) - y_i^s\|_F^2 \quad (5.2)$$

where y_i^s denotes the ground-truth density map of the x_i and $\|\cdot\|_F$ denotes the Frobenius norm.

Similarly, the loss $\mathcal{L}_{target}(\Theta)$ measures how well the model parameters perform on the small set of labeled training images in the target domain $\mathcal{D}_t^l$. It is defined as:

$$\mathcal{L}_{target}(\Theta) = \frac{1}{K} \sum_{i=1}^K \|\Phi_{\Theta}F_{\Theta}(x_i^t) - y_i^t\|_F^2 \quad (5.3)$$

Domain Adaptation Loss: We define a domain adaptation loss that measures the distance of the feature space between the source and target domains. This loss is based

on MMD [41] as used in [44; 59; 43]. MMD projects the feature of each data point to a Reproducing Kernel Hilbert Space (RKHS) to construct a mean embedding. Then the distance is computed between the constructed mean embeddings. The expression for the source feature representation is denoted by $\mathcal{F}_i^s = \Psi_\Theta(F_\Theta(x_i^s))$ (where $i \in \{1, 2, \dots, N_s\}$). Similarly, the expression for target stream is represented by $\mathcal{F}_j^t = \Psi_\Theta(F_\Theta(x_j^t))$ (where $j \in \{1, 2, \dots, N_t\}$). The calculation of MMD between the feature representations can be expressed as:

$$\mathcal{L}_{mmd}(\Theta) = \left\| \sum_{i=1}^{N_s} \frac{\chi(\mathcal{F}_i^s)}{N_s} - \sum_{j=1}^{N_t} \frac{\chi(\mathcal{F}_j^t)}{N_t} \right\|^2 \quad (5.4)$$

where $\chi(\cdot)$ denotes the mapping to RKHS. Eq. 5.4 can be further expanded to replace the inner products by kernel values based on the kernel trick and is expressed as:

$$\begin{aligned} \mathcal{L}_{mmd}(\Theta) = & \Delta(\mathcal{F}_i^s, \mathcal{F}_{i'}^s, N_s, N_s) - 2 * \Delta(\mathcal{F}_i^s, \mathcal{F}_j^t, N_s, N_t) \\ & + \Delta(\mathcal{F}_j^t, \mathcal{F}_{j'}^t, N_t, N_t), \text{ where} \end{aligned} \quad (5.5)$$

$$\Delta(\mathcal{F}_i^s, \mathcal{F}_{i'}^s, N_s, N_s) = \sum_{i, i'} \frac{k(\mathcal{F}_i^s, \mathcal{F}_{i'}^s)}{(N_s)^2} \quad (5.6)$$

$$\Delta(\mathcal{F}_i^s, \mathcal{F}_j^t, N_s, N_t) = \sum_{i, j} \frac{k(\mathcal{F}_i^s, \mathcal{F}_j^t)}{(N_s N_t)} \quad (5.7)$$

$$\Delta(\mathcal{F}_j^t, \mathcal{F}_{j'}^t, N_t, N_t) = \sum_{j, j'} \frac{k(\mathcal{F}_j^t, \mathcal{F}_{j'}^t)}{(N_t)^2} \quad (5.8)$$

Here $k(\cdot, \cdot)$ is a kernel function defined as $k(x, x') = \exp(-\|x - x'\|^2 / \sigma)$, with bandwidth parameter σ . In our work, we fix σ to be 1.

For the unsupervised domain adaptation, we ignore the loss term $\mathcal{L}_{target}(\Theta)$ since we do not have labeled images in the target domain. So the loss function in UDA can be written as:

$$\mathcal{L}_{UDA}(\Theta) = \mathcal{L}_{source}(\Theta) + \alpha \mathcal{L}_{mmd}(\Theta) \quad (5.9)$$

For the supervised case, the loss function is identical the same as Eq. 5.1. The only difference is that N_t in Eq. 5.4 will be K in this case.

5.3 Experiments

In this section, we first give details of the datasets and experiment setup in Sec. 5.3.1. In Sec. 5.3.2, we define several baseline methods for comparison, and then followed by our experimental results in Sec. 5.3.3. Also, to demonstrate the generality of our method, in addition to performing experiments where both source and target domains correspond to the same object class (i.e., people), we also perform a cross-object domain adaptation. In this setting, the source and target domains involve counting objects of different classes. For example, we consider the source domain involves counting people, while the target domain involves counting cars. The motivation of this cross object domain adaptation is that current crowd counting datasets are limited to a handful of object categories (e.g., mostly people [60; 2; 4]). However, in real-world applications, we might be interested in counting other objects in images. If we can perform well in cross-object domain adaptation, we can use the existing object counting datasets to help to count other objects in the target domain.

5.3.1 Datasets and Setup

Datasets: We use 4 datasets in the experiment, including ShanghaiTech Part_A (SHA) [1], Mall [2], UCSD [60] and TRANCOS [3]. ShanghaiTechA is a dataset containing dense crowd of people, while Mall and UCSD contain relatively sparse crowd of people. TRANCOS is a dataset of vehicle counting. We use these datasets to evaluate the

Source→Target	N_s	N_t	No. of test images
ShanghaiTechA→Mall	300	800	1200
Mall→ShanghaiTechA	100	300	182
UCSD→Mall	100	800	1200
Mall→UCSD	100	800	1200
UCSD→TRANCOS	100	403	421
Mall→TRANCOS	100	403	421

Table 5.1: Different domain adaptation settings in our experiment. For each setting, we show the number of labeled training images in the source domain (i.e., N_s), the number of images in the target domain (N_t), and the number of test images in the target domain. For SDA and SSDA in each setting, the number of labeled training images in the target domain (i.e., K) is $\{1, 5, 10\}$.

our approach when adapting between domains of different density levels, and even between different objects.

ShanghaiTech Part_A contains variable resolution of images randomly taken from Internet. Out of total 482 images, 300 images are selected for training and rest 182 are used for testing. The Mall dataset [2] consists of images captured from a surveillance camera in a mall. It is made up of 2000 images from a single scene with a size of 640×480 . Each person in an image is annotated with a point. The UCSD dataset contains 2000 labeled images from a single outdoor scene with an image size of 238×158 . TRANCOS contains 1244 labeled images capturing traffic jam from publicly available video surveillance cameras. In order to test the cross-object domain adaptation, we use the TRANCOS dataset for vehicle counting. We use the standard training/testing splits on these datasets. From the training splits, we then perform domain adaptation by using different subsets of these datasets as shown in Table 5.1. For example, the

setup “UCSD→Mall” corresponds to using the UCSD dataset as the source domain and the Mall dataset as the target domain. For simplicity, we randomly select 100 labeled images ($N_s = 100$) as the source domain. The last column in Table 5.1 shows the total number of test images in the target domain.

For the supervised (SDA) and semi-supervised (SSDA) domain adaptation, we consider the number of labeled images (i.e., K) to be $\{1, 5, 10\}$.

Generation of Ground-truth Density Maps: Similar to [1], we generate the ground-truth density maps for supervised learning from the provided point annotations (i.e., the center of the head of a person) in each of the datasets. The process involves applying a Gaussian kernel to the point annotations and normalizing the result to have a sum of one.

Evaluation Metrics: We use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the evaluation metrics. They are defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |\hat{g}_i - g_i|, \quad \text{MSE} = \frac{1}{N} \sum_{i=1}^N (\hat{g}_i - g_i)^2 \quad (5.10)$$

where $\hat{g}_i$ and g_i denote the predicted and ground-truth counts of the i -th test image, respectively. The predicted count $\hat{g}_i$ is obtained by integrating over the predicted density map.

Implementation Details: To speed up the training, we first pre-train the model only using the training images in the source domain by minimizing $\mathcal{L}_{source}(\Theta)$ defined in Eq. 5.2. We then fix the model parameters in the backbone network except for the fusion and the 1×1 conv layer. The MMD loss is defined on the output of the 1×1 conv layer. We use a learning rate of $1e-5$, a momentum of 0.9, a batch size of 1 and the Adam optimizer. Since the original MCNN code from [1] is not publicly available,

Train	Test	MAE	MSE
ShanghaiTechA	ShanghaiTechA	110	169
UCSD	UCSD	1.07	1.35
Mall	Mall	2.60	3.32
Trancos	Trancos	5.53	7.39
Mall	ShanghaiTechA	407	683
ShanghaiTechA	Mall	22.67	23.24
UCSD	Mall	24.25	25.17
Mall	UCSD	11.26	11.55
UCSD	TRANCOS	22.08	30.52
Mall	TRANCOS	21.15	26.59

Table 5.2: An illustration of the domain shift problem in commonly used counting datasets. When learning the model on the training images of a dataset and test the model on the test data of the same dataset, the performance is generally good (first four rows). However, when we take a model learned from the training images of one dataset and test on the test images in a different dataset, the performance drops dramatically (last six rows). This demonstrates the need for domain adaptation in crowd counting. The results are obtained using a 3d-party implementation of MCNN [1] which achieves performance similar to the original numbers (e.g. MAE 110.2 and MSE 173.2 on ShanghaiTechA, as in [1]).

we use a 3rd-party implementation of MCNN ¹ which gives similar performance to the performance numbers reported in [1] (see Table 5.2).

5.3.2 Baselines

There is no previous work that we can directly compare with. Nevertheless, we define several baselines by ourselves, so that we can compare and demonstrate the effectiveness of the proposed approach. These baselines are the default methods that people might currently use in practice.

¹<https://github.com/svishwa/crowdcount-mcnn>

Source→Target	Models	1-shot ($K = 1$)		5-shot ($K = 5$)		10-shot ($K = 10$)	
		MAE	MSE	MAE	MSE	MAE	MSE
ShanghaiTechA → Mall	pre-trained	22.67	23.24	22.67	23.24	22.67	23.24
	fine-tuning	5.61	7.04	5.58	6.98	5.56	6.92
	Ours (SDA)	5.36	6.65	5.34	6.62	5.42	6.72
	Ours (SSDA)	5.30	6.53	5.32	6.59	4.98	6.49
Mall → ShanghaiTechA	pre-trained	407	683	407	683	407	683
	fine-tuning	392.98	661.98	390.22	659.10	392	658.166
	Ours (SDA)	388.21	639.54	388.60	642.72	386.22	640.30
	Ours (SSDA)	384.83	641.94	382.56	639.50	382	639.88
UCSD → Mall	pre-trained	24.25	25.17	24.25	25.17	24.25	25.17
	fine-tuning	3.45	4.34	3.40	4.30	3.42	4.33
	Ours (SDA)	2.95	3.65	3.07	3.77	2.95	3.63
	Ours (SSDA)	2.93	3.62	2.92	3.60	2.92	3.64
Mall → UCSD	pre-trained	11.26	11.55	11.26	11.55	11.26	11.55
	fine-tuning	2.43	3.70	2.45	3.72	2.44	3.80
	Ours (SDA)	2.38	3.49	2.39	3.50	2.38	3.56
	Ours (SSDA)	2.36	3.57	2.35	3.48	2.35	3.55
UCSD → TRANCOS	pre-trained	22.08	30.52	22.08	30.52	22.08	30.52
	fine-tuning	19.21	24.64	19.22	24.65	19.22	24.61
	Ours (SDA)	17.85	23.14	17.07	22.49	17.06	22.34
	Ours (SSDA)	17.07	22.10	17.04	22.07	17.03	22.06
Mall → TRANCOS	pre-trained	21.15	26.59	21.15	26.59	21.15	26.59
	fine-tuning	17.58	22.97	17.59	22.99	17.58	22.97
	Ours (SDA)	17.54	23.04	17.55	22.92	17.58	22.92
	Ours (SSDA)	17.49	22.72	17.48	22.69	17.46	22.68

Table 5.3: Quantitative results for the supervised (SDA) and semi-supervised (SSDA) domain adaptation settings. We compare our models (“Ours(SDA)” and “Ours(SSDA)”) against the two baseline models represented as “pre-trained” and “fine-tuning”. We compare the results using MAE and MSE metrics for K -shots, where $K \in \{1, 5, 10\}$. Lower MAE/MSE means better performance. The first two rows correspond to domain adaptation between ShanghaiTechA (dense crowd) and Mall (sparse crowd) datasets where the source and target domains have different density levels. The 3rd and 4th rows show the adaptation between Mall (sparse crowd) and UCSD (sparse crowd) where two domains have similar density levels. The last two rows correspond to cross-object domain adaption, where the source domain (UCSD, Mall) is for person counting and the target domain (TRANCOS) is for vehicle counting. The 1st and 2nd best results in each setting are highlighted in red and blue, respectively.

Pre-trained: The baseline method ignores the data (either labeled or unlabeled) in the target domain. It uses the labeled source training images to learn the model parameters. Then it directly applies the pre-trained model to the test data in the target domain.

Fine-tuning: This baseline is the standard fine-tuning technique commonly used in deep learning. First, we train the model using the labeled images in the source domain to generate a pre-trained model. Then we fine-tune the model on the small number of labeled images in the target domain. Note that this baseline is only applicable in the SDA and SSDA settings since these two settings assume the target domain has some labeled training images. However, it does not work in the UDA setting in which there are no labeled training images in the target domain.

5.3.3 Experimental Results

Before presenting our main experimental results, we first perform an experiment as a sanity check to illustrate the domain shift problem in commonly used crowd counting datasets. This will demonstrate the need for domain adaptation in crowd counting. In Table 5.2, we show the quantitative results using training/test datasets from different combinations of various datasets. We see that the performance is generally good when both training and test data are from the same dataset. However, when we train the model using one dataset and test on another dataset, there is a significant drop (on the order of 10-20 times) in performance. This shows that there is a notable domain shift between the common datasets used in crowd counting. Without explicit domain adaptation, a model trained on one dataset (i.e. source) usually does not generalize

well to another different dataset (i.e. target).

Table 5.3 shows the results of supervised and semi-supervised domain adaptation when $K = 1, 5, 10$. We compare our approach (“Ours (SDA)” and “Ours (SSDA)”) with the two baselines (“pre-trained” and “fine-tuning”). Our proposed method significantly outperforms the baselines. In most cases, our semi-supervised method performs the best. This is reasonable since this setting makes the best use of all the training information available. Interestingly, there is no significant difference among the results when we vary the value of K from 1 to 10. One possible explanation is that the value of K is relatively small. When we increase K from 1 to 10, it is not enough to have a significant impact on the learning.

We present the results of unsupervised domain adaptation in Table 5.4. We compare our method with the baseline which directly applies the pre-trained model from the source domain to the target domain. Our method significantly outperforms this baseline. Note that we cannot compare with the “fine-tuning” baseline as it requires access to labeled training images in the target domain.

5.3.4 Analysis

We perform additional analysis to get further insight into our proposed approach. For a target domain, we define the following two “oracle” methods which have access to enough labeled training data in the target domain.

Oracle-1: This is the standard supervised learning method with fully-labeled training data in the target domain.

Oracle-2: In this oracle method, we use all training images in both source and target

Source→Target	Models	MAE	MSE
ShanghaiTechA →	pre-trained	474.81	591.73
ShanghaiTechA	Ours (UDA)	225.03	337.894
ShanghaiTechA →	pre-trained	22.67	23.24
Mall	Ours (UDA)	5.61	7.04
Mall →	pre-trained	407	683
ShanghaiTechA	Ours (UDA)	392.98	661.88
UCSD → Mall	pre-trained	24.25	25.17
	Ours (UDA)	2.93	3.65
Mall → UCSD	pre-trained	11.26	11.55
	Ours (UDA)	2.52	3.38
UCSD → TRANCOS	pre-trained	22.08	30.52
	Ours (UDA)	17.19	22.14
Mall → TRANCOS	pre-trained	21.15	26.59
	Ours (UDA)	17.90	23.36

Table 5.4: Quantitative results of the unsupervised domain adaptation (UDA), where the target domain only have unlabeled training images. We compare our UDA approach to the baseline which uses the pre-trained model from the source domain. Lower MAE/MSE means better performance. Our method significantly outperforms the baseline. Note that we cannot compare with the “fine-tuning” baseline in this case, since it requires labeled images in the target domain. The best results are highlighted in red.

domains. However, we now assume that all N_t images in the target domain are fully labeled. This oracle method can be considered as a special case of our semi-supervised setting (SSDA) in the extreme case where all images in the target domain are labeled.

We can consider the performance of these two oracle methods to be the upper bound of our approach. The difference between these two oracle methods is as follows. Oracle-1 only consider the training data in the target domain and it can overfit to this domain. Oracle-2 is trained with more training data since it uses training images from source and target domains. However, Oracle-2 is not explicitly trained to fit the

Target	Method	Source	MAE	MSE
SHA	Oracle-1	-	110	169
	Oracle-2	Mall	369.51	486.66
	Ours (UDA)	Mall	392.98	661.88
	Ours (SDA)	Mall	388.21	639.54
	Ours (SSDA)	Mall	384.83	641.94
Mall	Oracle-1	-	2.60	3.32
	Oracle-2	SHA	5.23	6.42
	Ours (UDA)	SHA	5.61	7.04
	Ours (SDA)	SHA	5.36	6.65
	Ours (SSDA)	SHA	5.30	6.53
Mall	Oracle-1	-	2.60	3.32
	Oracle-2	UCSD	2.90	3.60
	Ours (UDA)	UCSD	2.93	3.65
	Ours (SDA)	UCSD	2.95	3.65
	Ours (SSDA)	UCSD	2.93	3.62
UCSD	Oracle-1	-	1.07	1.35
	Oracle-2	Mall	2.28	3.37
	Ours (UDA)	Mall	2.52	3.38
	Ours (SDA)	Mall	2.38	3.49
	Ours (SSDA)	Mall	2.36	3.57
TRANCOS	Oracle-1	-	4.67	6.61
	Oracle-2	UCSD	16.98	21.05
	Ours (UDA)	UCSD	17.19	22.14
	Ours (SDA)	UCSD	17.85	23.14
	Ours (SSDA)	UCSD	17.07	22.10
	Oracle-2	Mall	17.43	22.65
	Ours (UDA)	Mall	17.90	23.36
	Ours (SDA)	Mall	17.54	23.04
	Ours (SSDA)	Mall	17.49	22.72

Table 5.5: Comparison of our domain adaptation approaches with two oracle methods. The two oracle methods have access to all labeled images in the target domain. Oracle-1 is a standard supervised method trained directly in the target domain. Note that Oracle-1 does not use the data in the source domain. Oracle-2 is our SSDA approach in the extreme case where all images in the target domain are labeled. The 1st and 2nd best results in each setting are highlighted in red and blue, respectively.

target domain.

In Table 5.5, we compare our domain adaptation approach to these oracle methods. Note that in the case of SDA and SSDA, we have used the result from the 1-shot ($K = 1$) setting. We can make several interesting observations. First, Oracle-1 consistently outperforms Oracle-2 on all datasets. This suggests that if we have enough labeled data in the target domain and if we only care about the model performance in the target domain, we should directly use standard supervised method trained in the target domain. The impact of domain adaptation is more pronounced when the labeled data are scarce in the target domain. Second, the difference between Oracle-2 and our approach is not significant. This means if we use the proposed domain adaptation method, the number of labeled images in the target domain does not significantly impact the performance of the model. This is in fact good news for practical applications, since this means that annotating a small number of images in the target domain is enough to achieve good performance comparable to that of having more images.

Chapter 6

Conclusion and Future Work

We have proposed three pieces of work on crowd counting in images. First, we have presented scale-aware attention networks for crowd counting in images. The novelty of our work is that we use the attention mechanism to softly select the appropriate scales at both global and local levels. Compared with DecideNet [5] that learns to switch between two different models (detection vs density estimation), our model is much simpler since it only has one model (density estimation). Our experimental results demonstrate that our proposed model outperforms other state-of-the-art approaches for crowd counting. Secondly, we have introduced a new problem called the one-shot scene-specific crowd counting. This new problem setting is motivated by the disconnect between current research in crowd counting algorithms and how these algorithms are deployed in real-world application. We introduce a novel algorithm for this problem. Our experimental results show that the proposed approach outperforms other baseline methods. Finally, we have proposed the first work on deep learning based domain adaptation in crowd counting. Our method can operate in supervised,

unsupervised, and semi-supervised settings. We have demonstrated experimentally that our approach outperforms other alternative approaches. In addition to the standard domain adaptation where the object of interest is the same in both source and target domains, we have shown that our method is also applicable for cross-object domain adaptation.

Bibliography

- [1] 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*, 2016. [v](#), [vi](#), [vii](#), [viii](#), [1](#), [2](#), [5](#), [11](#), [12](#), [19](#), [20](#), [21](#), [22](#), [24](#), [25](#), [29](#), [36](#), [38](#), [43](#), [47](#), [48](#), [51](#), [53](#), [54](#)
- [2] K. Chen, C. C. Loy, S. Gong, and T. Xiang, “Feature mining for localised crowd counting,” in *British Machine Vision Conference*, 2012. [v](#), [vi](#), [vii](#), [viii](#), [2](#), [19](#), [20](#), [23](#), [25](#), [37](#), [43](#), [44](#), [51](#), [52](#)
- [3] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Onoro-Rubio, “Extremely overlapping vehicle counting,” in *Iberian Conference on Pattern Recognition and Image Analysis*, 2015. [v](#), [vii](#), [2](#), [37](#), [40](#), [51](#)
- [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*, 2015. [v](#), [vii](#), [viii](#), [2](#), [37](#), [38](#), [39](#), [40](#), [41](#), [43](#), [51](#)
- [5] J. Liu, C. Gao, D. Meng, and A. G. Hauptmann, “Decidenet: Counting varying density crowds through attention guided detection and density estimation,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. [vii](#), [6](#), [19](#), [20](#), [21](#), [22](#), [23](#), [61](#)

-
- [6] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, “Multi-source multi-scale counting in extremely dense crowd images,” in *IEEE Conferece on Computer Vision and Pattern Recognition*, 2013. [vii](#), [2](#), [4](#), [19](#), [20](#), [21](#), [24](#)
 - [7] 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*, 2008. [viii](#), [37](#), [41](#), [44](#)
 - [8] D. B. Sam, S. Surya, and R. V. Babu, “Switching convolutional neural networks for crowd counting,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. [1](#), [2](#), [6](#), [10](#), [11](#), [13](#), [19](#), [20](#), [21](#), [22](#), [24](#)
 - [9] C. Arteta, V. Lempitsky, and A. Zisserman, “Counting in the wild,” in *European Conference on Computer Vision*. Springer, 2016. [1](#)
 - [10] D. Onoro-Rubio and R. J. Lopez-Sastre, “Towards perspective-free object counting with deep learning,” in *European Conference on Computer Vision*, 2016. [1](#), [5](#), [24](#)
 - [11] H. Zhang, Z. Kyaw, S.-F. Chang, and T.-S. Chua, “Visual translation embedding network for visual relation detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. [1](#)
 - [12] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, “FCN-rLSTM: Deep spatial-temporal neural networks for vehicle counting in city cameras,” in *IEEE International Conference on Computer Vision*, 2017. [1](#)
 - [13] V. A. Sindagi and V. M. Patel, “Generating high-quality crowd density maps using contextual pyramid CNNs,” in *IEEE International Conference on Computer Vision*, 2017. [2](#), [6](#), [11](#), [13](#), [14](#), [19](#), [20](#), [21](#), [22](#), [24](#), [38](#)

- [14] M. Marsden, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor, "People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 46
- [15] 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*. Springer, 2013. 4
- [16] I. S. Topkaya, H. Erdogan, and F. Porikli, "Counting people by clustering person detector outputs," in *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*. IEEE, 2014. 4
- [17] E. Walach and L. Wolf, "Learning to count with cnn boosting," in *European Conference on Computer Vision*, 2016. 5, 23
- [18] 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*, 2015. 5, 20, 22, 24
- [19] L. Boominathan, S. S. Kruthiventi, and R. V. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016. 5, 24
- [20] 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*, 2017. 5, 13, 20, 38
- [21] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 6

-
- [22] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, “Matching networks for one shot learning,” in *Advances in Neural Information Processing Systems*, 2016. 6, 8
 - [23] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “One-shot learning with memory-augmented neural networks,” in *Advances in Neural Information Processing Systems*, 2016. 6
 - [24] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, “Learning feed-forward one-shot learners,” in *Advances in Neural Information Processing Systems*, 2016. 6, 8
 - [25] Y.-X. Wang and M. Hebert, “Learning to learn: Model regression networks for easy small sample learning,” in *European Conference on Computer Vision*, 2016. 6
 - [26] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, “One-shot video object segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7, 27, 30, 35, 36
 - [27] G. Csurka, “Domain adaptation for visual applications: A comprehensive survey,” 2017, arXiv:1702.05374. 7
 - [28] L. Fei-Fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 7
 - [29] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *International Conference on Machine Learning*, 2015. 7
 - [30] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, “Human-level concept learning through probabilistic program induction,” *Science*, 2015. 7

- [31] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, “Meta-learning with memory-augmented neural networks,” in *International Conference on Machine Learning*, 2016. 8
- [32] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International Conference on Machine Learning*, 2017. 8
- [33] G. Csurka, “Domain adaptation in computer vision applications: A comprehensive survey,” *arXiv preprint arXiv:1702.05374*, 2017. 8
- [34] M. Wang and W. Deng, “Deep visual domain adaptation: A survey,” *Neurocomputing*, 2018. 8
- [35] L. Bruzzone and M. Marconcini, “Domain adaptation problems: A dasvm classification technique and a circular validation strategy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 8
- [36] W.-S. Chu, F. De la Torre, and J. F. Cohn, “Selective transfer machine for personalized facial action unit detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 8
- [37] M. Gheisari and M. S. Baghshah, “Unsupervised domain adaptation via representation learning and adaptive classifier learning,” *Neurocomputing*, 2015. 8
- [38] B. Gong, K. Grauman, and F. Sha, “Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation,” in *International Conference on Machine Learning*, 2013. 8
- [39] S. Pachori, A. Deshpande, and S. Raman, “Hashing in the zero shot framework via domain adaptation,” *Neurocomputing*, 2018. 8

-
- [40] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis,” *IEEE Transactions on Neural Networks*, 2011. 8
 - [41] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, “A kernel method for the two-sample-problem,” in *Advances in Neural Information Processing Systems*, 2007. 8, 50
 - [42] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, “Correcting sample selection bias by unlabeled data,” in *Advances in Neural Information Processing Systems*, 2007. 8
 - [43] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *arXiv preprint arXiv:1412.3474*, 2014. 8, 50
 - [44] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *International Conference on Machine Learning*, 2015. 9, 50
 - [45] V. Mnih, N. Heess, A. Graves *et al.*, “Recurrent models of visual attention,” in *Advances in Neural Information Processing Systems*, 2014. 10
 - [46] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International Conference on Machine Learning*, 2015. 10
 - [47] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 10

- [48] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *Advances in neural Information Processing Systems*, 2016. 22, 23
- [49] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems*, 2015. 22, 23
- [50] M. Marsden, K. McGuiness, S. Little, and N. E. O’Connor, “Fully convolutional crowd counting on highly congested scenes,” *arXiv preprint arXiv:1612.00220*, 2016. 22
- [51] R. Benenson, M. Omran, J. Hosang, and B. Schiele, “Ten years of pedestrian detection, what have we learned?” in *European Conference on Computer Vision*. Springer, 2014. 23
- [52] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 23
- [53] Y. Wang and Y. Zou, “Fast visual object counting via example-based density estimation,” in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016. 23
- [54] S. Kumagai, K. Hotta, and T. Kurita, “Mixture of counting cnns: Adaptive integration of cnns specialized to specific appearance for crowd counting,” *arXiv preprint arXiv:1703.09393*, 2017. 23
- [55] B. Sheng, C. Shen, G. Lin, J. Li, W. Yang, and C. Sun, “Crowd counting via weighted vlad on dense attribute feature maps,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2016. 23

-
- [56] V. Lempitsky and A. Zisserman, “Learning to count objects in images,” in *Advances in Neural Information Processing Systems*, 2010. 24
 - [57] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015. 38
 - [58] 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*, 2017. 38
 - [59] A. Rozantsev, M. Salzmann, and P. Fua, “Beyond sharing weights for deep domain adaptation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 50
 - [60] 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*, 2008. 51