Scalable Human Identification with Deep Learning

XIAO, Tong

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Doctor of Philosophy

in

Electronic Engineering

The Chinese University of Hong Kong

August 2017

Abstract

Human identification, which aims at finding a target person of interest from a gallery of digital photos, is one of the fundamental problems in computer vision. It is a key component in many real-world applications including smart phone apps, self-driving cars, home security systems, and intelligent surveillance cameras.

Thanks to the development of deep learning research and large-scale well annotated datasets, deep neural networks are now capable of recognizing thousands of object categories. However, human identification is still challenging because it lacks a dataset large enough to supervise the model training. Moreover, existing small datasets usually have their own image biases, which makes it hard to learn a single model that generalizes over all these domains. Meanwhile, most of the existing research simplified the problem setting, which leaves a gap between research approaches and practical applications.

In this dissertation we address these challenges from three aspects to make human identification scalable to real-world data and applications. First, we propose a semi-supervised deep learning framework that uses noisy-labeled rather than well annotated data. We collect a large-scale clothing dataset with noisy annotations, from which we can learn good representations for clothes that help recognize human. Second, we develop a joint single task learning algorithm and a domain guided dropout technique to learn a single model from multiple human identification datasets with domain biases. It enables us to collectively use the data contributed by different people in the community. At last, we focus on the more realistic problem setting that finds a target person in whole scene images. We develop a unified framework that combines person detection and identification, as well as a loss function that trains the identification model effectively.

摘要

人物識別是計算機視覺領域非常重要的一類基礎問題，其主要研究的是如何從一個圖片或者視頻集中找出一個目標人物。人物識別廣泛應用於實際場景，是很多產品的重要組件，例如各類手機相冊應用、自動駕駛係統、家庭安全防範係統、以及智能監控攝像頭等等。

依托於深度學習技術的發展和大型帶標註數據集的出現，深度神經網絡目前已經可以在照片中識別上千種物體。但是人物識別仍然是一個很有挑戰性的問題，因為該領域缺乏一個足夠大規模的、帶標註的數據集以訓練有效的深度神經網絡。與此同時，已有的小規模數據集通常收集自不同的場景，其圖片各有偏頗，導緻很難訓練一個模型使其在所有的數據集上同時有效。另外，已有的研究工作多數集中於較為簡單的問題設置，和真實的應用場景存在較大偏差，使得很難將研究直接用於實際產品。

因此，在本文中我們將從三個角度解決這些問題，從而使人物識別技術擴展到真實的數據和應用上麵。首先，我們提出使用帶噪聲、易收集的數據來代替精標註、難收集的數據來訓練模型。我們還從互聯網上收集了一個大規模的、含有錯誤標註的衣物數據集，並用我們提出了的深度學習模型從此類帶噪聲的數據中學習有用的服裝特征，從而幫助人物識別。其次，我們針對多個各有偏頗的數據集，提出了一個聯合此類數據訓練的深度神經網絡的方法，從而使其可以有效地同時利用這些不同研究機構和個人收集的數據。最后，我們著手解決一個更貼近實際應用的問題設置，即從完整的場景圖片中找出並識別目標人物。我們設計了一個統一的深度神經網絡，同時進行人物檢測和識別，並且提出了一個新的目標函數用以更加高效地訓練整個網絡。

Acknowledgments

It is my greatest luck and honor to have worked with many inspiring people during my Ph.D. career. First and foremost, I would like to express my sincere appreciation and gratitude to my advisor Xiaogang Wang, who is a great mentor leading me to the research field of computer vision and deep learning. He set an example of being creative, rigorous, and hard working. Without his help and guidance I would hardly finish this dissertation.

I enjoyed a wonderful postgraduate study at The Chinese University of Hong Kong, where I met and worked with fellow professors and students: Kai Kang, Xiao Chu, Jing Shao, Lu Sheng, Xingyu Zeng, Shuai Yi, Wei Li, Rui Zhao, Shuang Li, Yantao Shen, Hongyang Li, Wei Yang, Yikang Li, Zhuoyi Zhao, Yuanjun Xiong, Ruohui Wang, Hongsheng Li, Wanli Ouyang, Dahua Lin, Chen Change (Cavan) Loy, and Xiaoou Tang. It is my fortunate to learn a lot from all these collaborators.

I really appreciate Tian Xia, Yi Yang, and Chang Huang, for offering me the great opportunity to glance at deep learning in industry, which encourages me to solve the problems that really matter.

I would also like to thank Cheng Tai, Zhirong Wu, Jiaxin Mao, Chiheng Xu, and Xudong Yang. These talented people enlarged my vision and enlightened my career as a student.

Last but not least, my deepest thanks go to my parents and Yahui Chen, who always support and encourage me to pursue my dream.

Declaration

I hereby declare that this dissertation is composed by myself and all the contents has not been submitted to this or any other universities for a degree. The materials of some chapters have been published in the following conference proceedings or journals.

- Wei Li, Rui Zhao, **Tong Xiao**, and Xiaogang Wang, "DeepReID: Deep Filter Pairing Neural Network for Person Re-Identification." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2014.

- **Tong Xiao**, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang, "Learning from Massive Noisy Labeled Data for Image Classification." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2015.

- **Tong Xiao**, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang, "Learning Deep Feature Representations with Domain Guided Dropout for Person Re-identification." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2016.

- Cheng Tai, **Tong Xiao**, Yi Zhang, Xiaogang Wang, and Weinan E, "Convolutional Neural Networks with Low-rank Regularization." In *Proceedings of the International Conference on Learning Representations* (***ICLR***), 2016.

- Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, **Tong Xiao**, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang, "T-CNN: Tubelets with convolutional neural networks for object detection from videos." *IEEE Transactions on Circuits and System for Video Technology* (***TCSVT***), to be published.

- Xingyu Zeng, Wanli Ouyang, Junjie Yan, Hongsheng Li, **Tong Xiao**, Kun Wang, Yu Liu, Yucong Zhou, Bin Yang, Zhe Wang, Hui Zhou, and Xiaogang Wang, "Crafting GBD-Net for Object Detection" (***TPAMI***), to be published.

- **Tong Xiao**, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang, "Joint Detection and Identification Feature Learning for Person Search." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2017.

- Shuang Li, **Tong Xiao**, Hongsheng Li, Bolei Zhou, Dayu Yue, and Xiaogang Wang, "Person Search with Natural Language Description." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2017.

- Kai Kang, Hongsheng Li, **Tong Xiao**, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang, "Object Detection in Videos with Tubelet Proposal Networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (***CVPR***), 2017.

- Shuang Li, **Tong Xiao**, Hongsheng Li, Wei Yang, and Xiaogang Wang, "Identity-Aware Textual-Visual Matching with Latent Co-attention." In *Proceedings of the IEEE International Conference on Computer Vision* (***ICCV***), 2017.

- Yantao Shen, **Tong Xiao**, Hongsheng Li, Shuai Yi, Xiaogang Wang, "Learning Deep Neural Networks for Vehicle Re-ID with Visual-spatio-temporal Path Proposals." In *Proceedings of the IEEE International Conference on Computer Vision* (***ICCV***), 2017.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine that in the near future, people are able to travel with self-driving cars, which recognize who you are, automatically navigating to your destination, and pay attention to other cars and pedestrians along the way. Vision, as the most important perception for both human beings and artificial intelligence, is the key to interpret the surrounding environment and help finish all these tasks correctly.

Among the broad spectrum of topics, *human identification*, which aims at finding a person of interest from a gallery of digital photos or videos, is one of the most fundamental problems in computer vision. Human identification is closely related to real applications, for example, it enables smart phones to group together photos that belong to the same person, it helps home security systems recognize housebreakers, and it also assists police to locate and track criminals from massive surveillance videos.

Given a pair of person photos, although human can easily tell whether they are of the same identity or not, this task is extremely difficult to computers. Human can instantly figure out high-level attributes, such as gender and age, and then gradually compare their finer details, for example, the clothes colors, materials, styles, and the accessories. However, what computers percept from a digital photo is just an array of integers representing the color illumination at each pixel. Computers need to extract patterns from these numbers and understand their high-level semantic meanings, which are also called *features* or *feature representations* in computer vision. The challenge here is to define which features are useful for human identification, and what are their unique patterns.

Thanks to the rapid development of deep learning algorithms, researchers have proposed many effective approaches to tackle these challenges. Different from traditional methods [2–6] that manually define a fixed set of rules to extract features, deep learning

methods [7–10] automatically learn from data a parameterized function that directly transforms an input image into features. Instead of trying to define good features, deep learning solves the challenge by optimizing the feature extraction function for an objective in a data-driven manner. We have witnessed rapid progress of teaching computers to classify an image into 1000 categories—the accuracy increases from 71.8% to 96.4% in five years! The success of deep learning methods in image classification have laid a great foundation for solving our human identification problem.

However, human identification poses more challenges than image classification. First, it lacks enough supervised data for learning good features for human. Image classification often relies on large-scale datasets [11] with manual annotations. But most of their images consist of animals, plants, and man-made objects, which may not be relevant to human identification. Second, most of the existing datasets for human identification are collected by different research institutes. Their data each has its own biases in human races, clothing styles, and camera settings, which requires the designed model to be able to handle such domain discrepancies. Third, image classification often assumes that a major object is in the center of an image. But practical human identification, especially in surveillance applications, often requires to find a target person in whole scene images. It currently lacks a uniform framework for such practical applications.

In this dissertation we address these three challenges respectively, in order to make human identification methods scalable to large-scale real-world applications.

In Chapter 2 we cover some basic concepts about deep learning, including the mathematical formulation, backpropagation, learning objectives, and popular neural network structures for computer vision.

In Chapter 3 we introduce the background for human identification, reviewing some related work, and develop a roadmap for the research in this field.

In Chapter 4 we address the first challenge by leveraging the need of supervised data to noisy-labeled semi-supervised data. We propose a probabilistic model to describe how the label noise is generated for web images. The proposed probabilistic model is then integrated into a deep neural network. We develop an end-to-end optimization algorithm for the network to learn features directly from noisy-labeled images.

In Chapter 5 we tackle the domain discrepancy problem raised above. A joint

single task learning framework is developed and a domain guided dropout technique is proposed. Together they enable the learning of a single deep neural network with data from multiple biased domains.

In Chapter 6 we develop a new framework for finding a target person in whole scene images. It uses a *convolutional neural network (CNN)* that jointly detects pedestrians inside an image and extracts their identity features. A novel loss function is proposed to train the CNN effectively without whistles and bells.

Finally, we conclude this dissertation in Chapter 7.

Chapter 2

Deep Learning Basics

This chapter introduces some basic concepts about deep learning. Deep learning covers a large spectrum of applications. Categorized by the type of input data, deep learning models can be roughly classified into two classes — *feedforward neural network* for non-sequential data, such as static images, and *recurrent neural network* for sequential data, such as video, audio, or text in natural language. Deep learning also belongs to general machine learning, therefore multiple frameworks have been developed to adopt different levels of supervisory signals, including supervised, semi-supervised, unsupervised, and reinforcement learning. In this chapter we focus on the supervised learning of feedforward neural networks, which is also mainly used in this dissertation.

For supervised learning, a training dataset can be denoted by $\mathcal{D}_{train} = \left\{(x^{(i)}, t^{(i)})\right\}_N$, where x and t are input and target tensor, respectively, and N is the number of samples. A feedforward neural network is a parameterized function $f_\theta(x)$ that maps an input x to some output tensor y. The learning objective is to find the best parameter θ^* that minimizes a predefined loss function $L(y, t)$, formally

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L\left(f_\theta(x^{(i)}), t^{(i)}\right). \tag{2.1}$$

Section 2.1 will detail on how to parameterize $f_\theta(\cdot)$. Section 2.2 will introduce several commonly used loss functions $L(\cdot, \cdot)$. The optimization process of finding θ^* will be elaborated in Section 2.3.

2.1 Feedforward Neural Network

A typical feedforward neural network is a sequence of linear and non-linear operators, as depicted in Figure 2.1. These linear and non-linear operators are often encapsulated

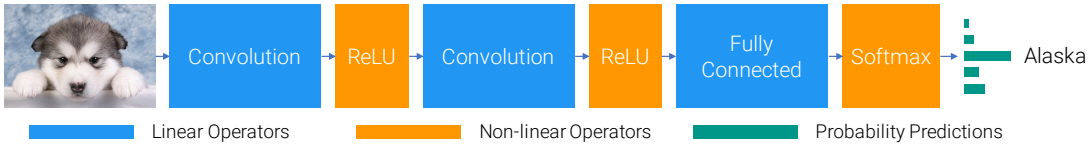


Figure 2.1: *Feedforward neural network consisting of sequential linear and non-linear operators*

into *layers* when designing neural networks. The output of the previous layer is served as the input of the next layer. Below we will introduce some commonly used operators.

2.1.1 Linear Operators

One of the basic linear operators is matrix-vector multiplication, namely the *fully connected (FC)* layer in neural network. Formally, the output of the FC layer is

$$y = Wx + b, \tag{2.2}$$

where $x \in \mathbb{R}^d$ is an input vector, $W \in \mathbb{R}^{m \times d}$ is a parameter dubbed *weight* matrix, and $b \in \mathbb{R}^m$ is a parameter dubbed *bias* vector. FC layers are often used as linear classifiers at then end of a neural network.

While each output dimension y_i, also called *unit*, is computed from all of the input dimensions, sometimes people wish to constrain the size of this input range, which forms another linear operator — *convolution*. Take as an example of 3-D input image $x \in \mathbb{R}^{C \times H \times W}$ with C channels and spatial size $H \times W$, a convolution layer is parameterized by M convolution kernels each has a weight $w \in \mathbb{R}^{C \times (2K+1) \times (2K+1)}$ and a bias b, where $2K + 1$ is called the kernel size (here we assume the kernel size is odd for simplicity). Applying each convolution kernel w to x results in an output *feature map* $y \in \mathbb{R}^{H \times W}$, where

$$y(i, j) = b + \sum_{c=1}^{C} \sum_{p=-K}^{K} \sum_{q=-K}^{K} w(c, p, q) x(c, i + p, j + q). \tag{2.3}$$

This process is demonstrated in Figure 2.2. Each output unit can be seen as the inner-product between the kernel weight and the input content inside a local region, which represents their matching similarities. The convolution kernel is shared across all the output units on the feature map. Thus it serves as a pattern *detector* that finds certain pattern on the input image.

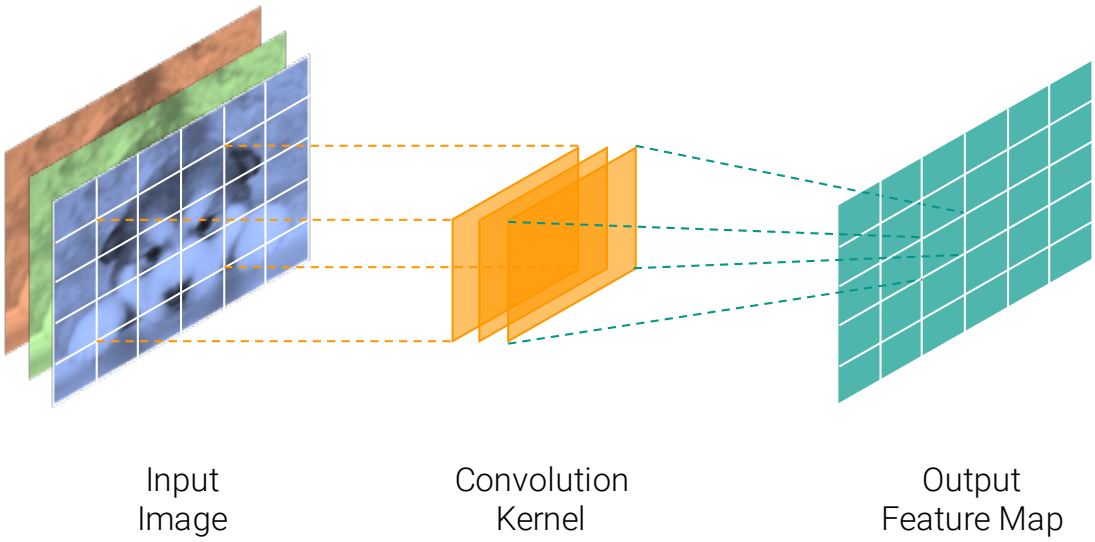


Figure 2.2: *Convolution operator on a 3-channel input image*

2.1.2 Non-linear Operators

Performing linear operations consecutively is useless, since they can be replaced by a single linear operation. For example, applying two FC layers

$$W_2(W_1x + b_1) + b_2 = (W_2W_1)x + (W_2b_1 + b_2) \tag{2.4}$$

is equivalent to a single FC layer with parameters $(W_2W_1, W_2b_1+b_2)$. In order to enrich the function family that neural networks can represent, people often insert non-linear operators between every two linear operators.

One of the most widely used non-linear operators is *rectified linear unit (ReLU)*, mathematically it is a piece-wise linear transformation applying to each input unit independently

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{when } x \geq 0, \\ 0 & \text{when } x < 0. \end{cases} \tag{2.5}$$

Another common non-linear operator is *softmax*, which maps a vector $x \in \mathbb{R}^d$ to a multinomial distribution

$$y_i = \mathrm{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}. \tag{2.6}$$

Note that $y_i > 0$ and $\sum_{i=1}^{d} y_i = 1$. It is often used at the end of a neural network for image classification, which casts the network output to a categorical distribution over predefined classes.

2.1.3 Normalizations

Normalizations form another important bunch of operators. They keep the output of each layer in a certain range of values, and ease the parameter optimization process. One of the most effective normalization operators is *Batch Normalization (BN)* [12]. Given N input samples each denoted by a vector $x^{(i)} \in \mathbb{R}^d, i = 1, \ldots, N$, BN computes the mean and variance of each dimension independently across all the samples

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)}, \tag{2.7}$$

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} (x_j^{(i)} - \mu_j)^2. \tag{2.8}$$

Then it normalizes the input to be zero mean and unit variance by

$$\hat{x}_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}, \tag{2.9}$$

where ϵ is a small positive number that avoids zero division. At last, BN applies a linear transformation to $\hat{x}$ by

$$y_j^{(i)} = \alpha_j \hat{x}_j^{(i)} + \beta_j, \tag{2.10}$$

where $\alpha, \beta \in \mathbb{R}^d$ are learnable parameters.

2.1.4 Residual Module

Many popular *convolutional neural network (CNNs)* have been proposed [7–9] to tackle the image classification problem. One simple yet effective variant worth mention here is the residual module [10]. The output of this module is

$$y = f_\theta(x) + x, \tag{2.11}$$

where the parameterized function $f_\theta(\cdot)$ learns some additive residual to the input. This residual module can at least keep the input unchanged, thus it will not be worse when

stacking more such layers. We refer the readers to [10, 13] for more detailed benefits.

2.2 Loss Functions

Discriminative learning has two types of objectives — *classification* and *regression*. Take as an example of predicting a person's age according to his/her photo, classification predicts a categorical class, such as child, teen, middle age, or elder, while regression directly predicts the number, such as 26.5-year old.

For classification, the target is denoted by an one-hot vector $t \in \mathbb{R}^d$ over d predefined classes, where $t_i = 1$ if the target is the class-i and all the other dimensions are zero. The neural network usually outputs a multinomial random variable $y \in \mathbb{R}^d$ as Eq. (2.6), and the *cross entropy* loss function is often used to indicate the discrepancy between random variables t and y:

$$\text{cross entropy}(t, y) = -\sum_{i=1}^{d} t_i \log y_i. \tag{2.12}$$

For regression, both the target and the network output can be denoted by arbitrary vectors $t, y \in \mathbb{R}^d$. Different distance metrics could be used as the loss function, and the *squared L2* distance is one the most widely used

$$\text{squared}_{L_2}(t, y) = ||t - y||_2^2 = \sum_{i=1}^{d} (t_i - y_i)^2. \tag{2.13}$$

In order to improve the robustness to the outliers, Girshick *et al.* [14] proposed an alternative *smooth L1* loss

$$\text{smooth}_{L_1}(t, y) = \sum_{i=1}^{d} \begin{cases} 0.5(t_i - y_i)^2 & \text{if } |t_i - y_i| < 1, \\ |t_i - y_i| - 0.5 & \text{otherwise.} \end{cases} \tag{2.14}$$

2.3 Backpropagation

Due to the highly non-linearity nature of the parametric form of a deep neural network, the objective function Eq. (2.1) usually does not have a closed-form global optimum. Therefore, we often seek for a local optimum instead by solving Eq. (2.1) iteratively using *stochastic gradient descent (SGD)*. During each iteration, we first randomly choose a subset, which is called a *minibatch* $\mathcal{D}_{minibatch} = \left\{\left(x^{(i)}, t^{(i)}\right)\right\}_M$, from the whole training

set. Then we compute the gradient of the loss function w.r.t. the parameters

$$\nabla_\theta L = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial L\left(f_\theta(x^{(i)}), t^{(i)}\right)}{\partial \theta}, \tag{2.15}$$

and update the parameters with a step along the direction of steepest descent

$$\theta \leftarrow \theta - \gamma \nabla_\theta L, \tag{2.16}$$

where γ is a scalar *learning rate*.

Backpropgation (BP), proposed by Rumelhart [15] and LeCun [16], is an efficient algorithm that computes the partial gradient in Eq. (2.15) efficiently, based on the *chain rule* of compound functions.

2.3.1 On a Chain of Layers

Let's first take a look at a simple case, where a neural network just stacks a sequence of layers as depicted in Figure 2.1. In such case, we can represent the whole network by a chain of compound functions

$$f(x_0) = f_D(f_{D-1}(\cdots f_2(f_1(x_0))\cdots)), \tag{2.17}$$

where x_0 is the input of the network, D dubbed *depth* is the number of layers, and $f_i(\cdot), i = 1, \ldots, D$ is the operator of the i-th layer, which takes x_{i-1} as input and outputs x_i. $f_i(\cdot)$ is parameterized by θ_i, if we vectorize all the x_i and θ_i, according to the chain rule,

$$\nabla_{\theta_i} L = \left(\frac{\partial x_i}{\partial \theta_i}\right)^T \nabla_{x_i} L, \tag{2.18}$$

where $\partial x_i / \partial \theta_i$ is informally the *Jacobian* matrix of x_i w.r.t. to θ_i. Meanwhile,

$$\nabla_{x_i} L = J_{i+1}^T \cdots J_{D-1}^T J_D^T \nabla_{x_D} L = J_{i+1}^T \nabla_{x_{i+1}} L, \tag{2.19}$$

where J_i is the Jacobian matrix of x_i w.r.t. x_{i-1}, which can be seen as a function of x_i and θ_i.

BP utilizes this recursive property, it first computes the outputs $x_1, \cdots, x_D$ from the first layer to the last layer and saves all their values in temporary buffers. Then it computes $\nabla_{x_D} L$ and goes back from the last layer to the first layer. At layer-i,

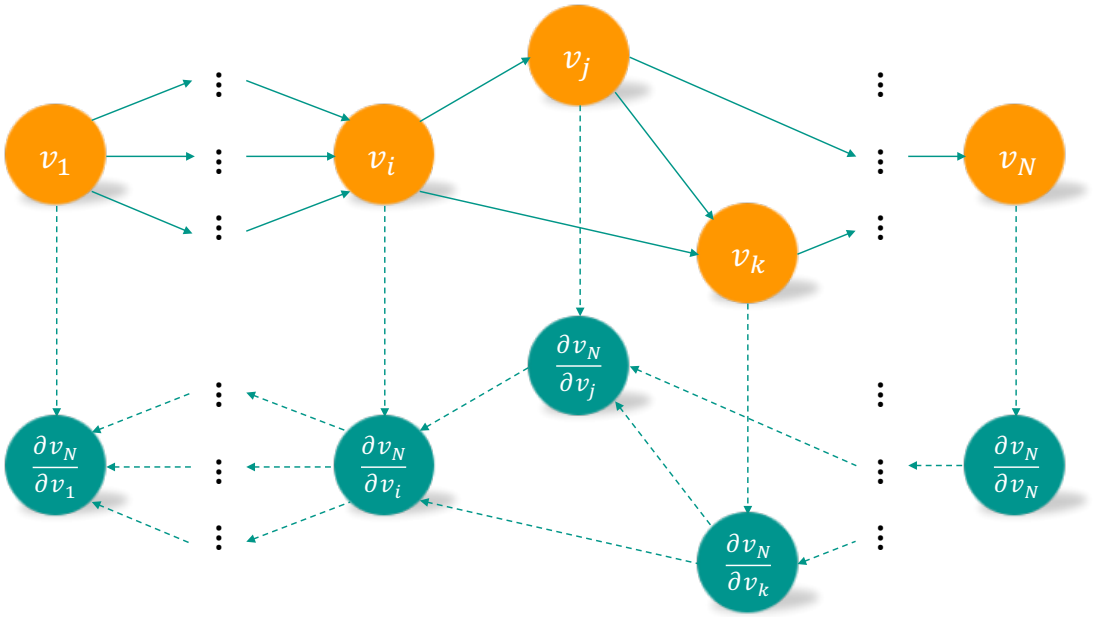


Figure 2.3: *Demonstration of a directed acyclic computational graph*

it computes both the gradient w.r.t. the parameters according to Eq. (2.18) and the gradient w.r.t. the layer inputs according to Eq. (2.19). The Jacobian matrices can be efficiently computed, since the outputs are all buffered.

2.3.2 On General Computational Graphs

The above analysis can be generalized to a more complicated *computational graph*, as long as it can be represented by a *directed acyclic graph (DAG)*. A DAG is defined by $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ is a set of nodes each representing a tensor (could be either an output or a parameter), and $E = \{(v_i, v_j)\}_M$ is a set of edges each indicating v_j relies on v_i as one of its inputs. A demonstration is shown in Figure 2.3.

Suppose all the nodes $v_1, \ldots, v_N$ are vectorized and *topologically sorted*, *i.e.*, $(v_i, v_j) \notin E, \forall i < j$. For arbitrary $i < k$, the gradients of node v_k w.r.t. v_i can be computed by

$$\frac{\partial v_k}{\partial v_i} = \sum_{j|(v_i,v_j)\in E} \left(\frac{\partial v_j}{\partial v_i}\right)^T \frac{\partial v_k}{\partial v_j}, \tag{2.20}$$

where $\partial v_j / \partial v_i$ is informally the Jacobian matrix of v_j w.r.t. v_i.

In neural networks, v_N (and maybe some other nodes that have no outgoing edges) is typically the loss. Thus to compute the gradient of loss w.r.t. the parameter nodes,

we can similarly perform a two-pass computation. In the forward pass (solid arrows in Figure 2.3), we sequentially compute the values from v_1 to v_N and store all their values in temporary buffers. And then in the backward pass (dashed arrows in Figure 2.3), we compute the gradient of v_N w.r.t. to each node v_i based on Eq. (2.20) sequentially from v_N to v_1. Whenever a forward or backward node is computed, we can virtually remove all its incoming edges. This would make some nodes no longer have outgoing edges and we can safely reuse their data buffer during implementation to save memory.

Chapter 3

Human Identification Background

This chapter introduces the background of human identification. Because of its rich values in real applications, human identification receives much research attentions in computer vision, which leads to various subtopics. In Section 3.1, we introduce different problem settings and evaluation metrics for human identification, as well as current research trend towards more realistic application scenarios. In Section 3.2, we develop a technical roadmap for human identification, and review some related work along this roadmap.

3.1 Problem Settings

Human identification aims at finding a target person in a gallery of photos. It can be categorized into three levels regarding different types of input photos — faces only, persons in photo albums, and pedestrians captured by surveillance cameras, as compared in Figure 3.1.

Facial images are the easiest case among the three, as human faces are more rigid-like. Slight deformation only appears when the person has expressions. Thus computers can easily align different facial images and compare among them, which is well studied in [17–20]. However, in many cases human faces cannot be clearly captured, for example, there are many people profiles in our daily photo albums. It is much harder and requires utilizing multiple cues, including facial profile, hair styles, body shapes, and clothes to recognize a person [21].

Apart from these two types of human photos captured by normal digital cameras, surveillance cameras capture another type of people — pedestrians. Due to the low resolution image with various weather and lighting conditions, as well as various human poses, recognizing pedestrians is very challenging. Researchers have named this problem

(a) Frontal Faces (b) People in Photo Album (c) Pedestrians

Figure 3.1: *Human identification with different types of input photos*

setting *Person Re-Identification (re-ID)* [22]. In this dissertation we also mainly focus on this setting.

3.1.1 Evaluation Metrics

Despite the difference of image types, all these problem settings can be abstracted as follows. Given a photo of the person of interest, which is often called the *query* or *probe*, and a collection of images, which is called the *gallery*, computers are required to rank the gallery images according to their similarity with the query photo.

Cumulative Matching Characteristics (CMC) curves are the most popular evaluation metrics for such problem settings. Consider a simple single-gallery-shot setting, where each gallery identity has only one instance. For each query, an algorithm should rank all the gallery samples according to their distances to the query from small to large, and the CMC top-k accuracy is

$$\text{cmc}_k = \begin{cases} 1 & \text{if the query identity is contained in the top-}k\text{ ranked gallery samples,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

which is a shifted step function. The final CMC curve is computed by averaging the

shifted step functions over all the queries.

While the single-gallery-shot CMC is well defined, it does not have a common agreement when it comes to the multi-gallery-shot setting, where each gallery identity could have multiple instances. Authors of different datasets have defined different criterion. For example, CUHK03 [23] and Market-1501 [24] calculate the CMC curves and CMC top-k accuracy quite differently. To be specific,

- CUHK03: Query and gallery sets are from different camera views. For each query, they randomly sample one instance for each gallery identity, and compute a CMC curve in the single-gallery-shot setting. The random sampling is repeated for multiple times and the expected CMC curve is reported.

- Market-1501: Query and gallery sets could have same camera views, but for each individual query identity, his/her gallery samples from the same camera are excluded. They do not randomly sample only one instance for each gallery identity. This means the query will always match the "easiest" positive sample in the gallery while does not care other harder positive samples when computing CMC.

3.1.2 Recent Advances

Apart from the classical settings introduced above, researchers have recently extended the human identification problem from various aspects. For example, traditional person re-identification requires both query and gallery images to be manually cropped, while [25–27] relaxed such constraints and considered detection with re-ID. Zheng *et al.* [28] explored person re-identification from videos rather than static images. Sometimes people do not have photo for the query person, for example, a criminal is only seen by a witness, in such case we can only rely on the testimony. Therefore, Li *et al.* [29] proposed to replace the query image with text descriptions to search a person among gallery photos, which enriches the application scenarios of human identification.

3.2 Technical Roadmap

After introducing the problem settings for human identification, this section develops a technical roadmap for solving the problem, which reviews some existing approaches

and enlightens future research.

Human identification should be addressed from three levels. At first, we only consider the information from a single image, and try to learn good features to represent the person's identity. Secondly, given two feature maps from a pair of images, we should develop methods to compare these two feature maps, and predict whether they belong to the same identity. At last, we could explore the structure of the whole gallery set, determining how the representations should be constructed for this set, and rank the gallery samples according to the query sample. We elaborate on these three aspects in Section 3.2.1, 3.2.2, and 3.2.3.

By interlacing human identification with other research fields we could advance the problem beyond the traditional image-based setting. For example, we may use natural language descriptions for querying a person instead of using the photo. Moreover, we could synthesize training data by exploiting generative models and rendering 3D models, which leverages the need of constructing large-scale supervised datasets. We introduce some recent advances towards these directions in Section 3.2.4 and 3.2.5.

3.2.1 Identity Feature Learning

The first fundamental question is how to learn discriminative feature representations for a single person image. Given an input image x, the objective is to learn a feature vector $f(x)$ such that the distance between the same identity $||f(x) - f(x^+)||_2$ (where x and x^+ are images of the same identity) is small enough, while the distance among different people $||f(x) - f(x^-)||_2$ (where x and x^- are images of different people) is large enough.

Researchers have proposed various loss functions to learn such features. For example, Sun *et al.* [17,18] proposed to train the neural network to classify the identities with softmax cross entropy loss introduced in Section 2.2. Although it is indeed a surrogate loss function for feature learning, it works reasonably well in practice. Later on triplet loss is proposed [19] that learns the features directly for ranking rather than classification. And Parkhi *et al.* [20] combined both loss functions together in a multi-staged training process.

A disadvantage of triplet loss is that it can only compare one pair of samples at each time, which is inefficient and highly relies on good sampling strategies. Therefore,

in Section 6.4 we proposed a novel loss function dubbed *Online Instance Matching (OIM)* that compares the features among all identities at once. OIM loss absorbs the advantages from softmax and triplet loss but leaves their disadvantages behind. Several other recent work [30–32] also explored similar ideas.

3.2.2 Feature Comparison

After getting discriminative feature representations for each single image, the next question is how to compare a pair of image features. Prior to deep learning, many research focused on metric learning [6, 33–37], which learns a distance function $d(f(x_1), f(x_2))$ that replaces the conventional L2-distance. However when deep learning emerges, learning simple linear metrics can be casted to learning underlying features, which makes metric learning less significant.

Comparing a pair of deep learning feature maps is nontrivial, since the feature maps contain spatial information and the two feature maps are usually not well aligned, for example, the region of frontal face in one image could be just background in the other image. Li *et al.* [23] proposed a siamese deep learning architecture that computes the correlation between the strips on two feature maps, and estimates the similarity according to the correlation. Ahmed *et al.* [38] further changed the correlation from strips to local rectangle regions, which significantly improved the performance. This feature maps matching problem is also closely related to the pixel correspondence problem for *optical flow* estimation [39, 40]. One may get inspired from these work to design better architectures for capturing geometric transformations.

3.2.3 Structure of Feature Set

The last question is how to rank a set of gallery images according to the query. Answering this question requires modeling the structure of the gallery feature set. For example, by observing many gallery photos one can understand what the *salient* features are for distinguishing different people [41, 42]. Several other work focused on the ranking algorithm itself. Zhong *et al.* [43] proposed a re-ranking algorithm by exploiting the prior knowledge that each identity has at least two instances in gallery set. Liu *et al.* [44] designed an algorithm cooperate with human labelers to refine a initial ranking result.

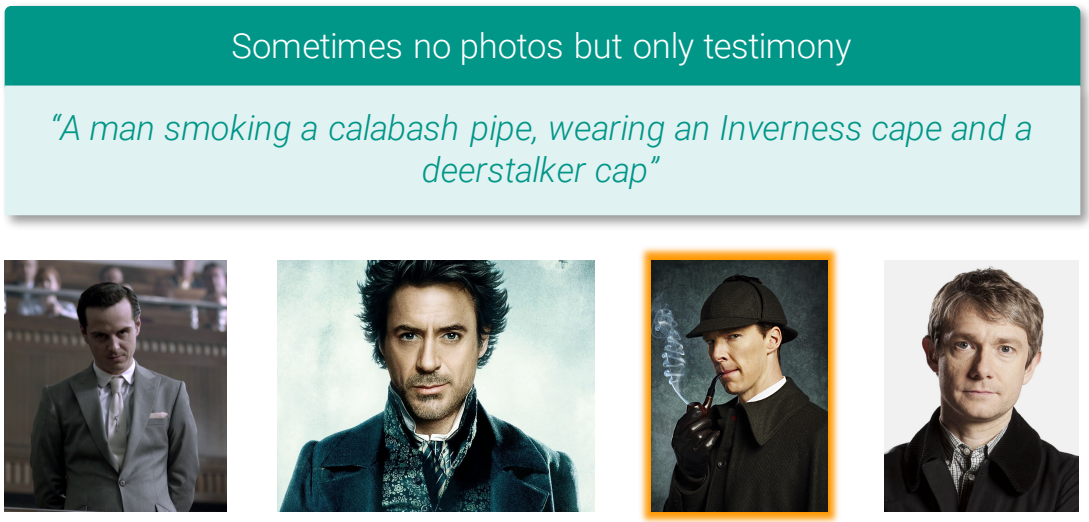


Figure 3.2: *Human identification with natural language descriptions as query*

3.2.4 Data Modalities

In some real-world scenarios, we do not have photos for the query person, for example when catching a criminal, the police sometimes only have testimony from a witness, as demonstrated in Figure 3.2. Studying different data modalities for human identification thus becomes very important for pushing it towards real applications. Li *et al.* [29] introduced the person search problem with natural language description. They proposed a CNN-RNN architecture with attention mechanisms to solve the problem, and collected a dataset to validate their approach. It was further improved in [45] by exploiting the identity information to help feature learning.

The relationship between image modality and language modality has been actively studied in recent years from different aspects, including image captioning [46–49], visual question answering (VQA) [50–53], and text-based image retrieval [54]. For human-identification, one intriguing property of using language modality is that it enriches the information we have. Traditionally, we only know whether each pair of people matches or not. With natural language descriptions, we can further know why they match and what the evidences are. These additional information could make the learned deep neural networks more discriminative and generalize better.

3.2.5 Generative Models and Rendering 3D Models

Learning well-performed deep neural networks currently requires large-scale supervised datasets, but collecting the data could be tedious and time consuming. To cope with this problem, recent work [55, 56] exploited computer graphics technologies that render 3D models to high-quality life-like 2D images. They showed that combining small amount of real data and numerous synthetic data could significantly improve the performance of deep neural networks for scene parsing.

One problem of using synthetic images is that when they are too different from real images, it could be meaningless to learn deep neural networks with them. To reduce the gap of the discrepancy between synthetic data and real data, Shrivastava *et al.* [57] exploit the *Generative Adversarial Networks (GANs)* to transfer the synthetic images to real images. In the person re-identification literature, Zheng *et al.* [58] use GAN to directly generate the person images, which serve as training data for training deep neural networks.

Chapter 4

Learning Features from Noisy Labels

Large-scale supervised datasets are crucial to train CNNs for various computer vision problems. However, obtaining a massive amount of well-labeled data is usually very expensive and time consuming. In this chapter, we develop a general framework to train CNNs with only a limited number of clean labels and millions of easily obtained noisy labels. We model the relationships between images, class labels and label noises with a probabilistic graphical model and further integrate it into an end-to-end deep learning system. To demonstrate the effectiveness of our approach, we collect a large-scale real-world clothing classification dataset with both noisy and clean labels. Experiments on this dataset indicate that our approach can better correct the noisy labels and improves the performance of trained CNNs.

The contribution of this chapter is three-fold. First, we study the cause of noisy labels in real-world data and describe it with a novel probabilistic model. Second, we integrate the model into a deep learning framework and explore different training strategies to make the CNNs learn from better supervisions. Finally, we collect a large-scale clothing dataset with both noisy and clean labels, which will be released for academic use.

4.1 Related Work

Deep learning with large-scale supervised training dataset has recently shown very impressive improvement on multiple image recognition challenges including image classification [7], attribute learning [59], and scene classification [60]. While state-of-the-art results have been continuously reported [8, 9, 61], all these methods require reliable annotations from millions of images [11] which are often expensive and time-consuming to obtain, preventing deep models from being quickly trained on new image recognition

problems. Thus it is necessary to develop new efficient labeling and training frameworks for deep learning.

One possible solution is to automatically collect a large amount of annotations from the Internet web images [62] (*i.e.* extracting tags from the surrounding texts or keywords from search engines) and directly use them as ground truth to train deep models. Unfortunately, these labels are extremely unreliable due to various types of noise (*e.g.* labeling mistakes from annotators or computing errors from extraction algorithms).

Learning with noisy labeled training data has been extensively studied in the machine learning and computer vision literature. For most of the related work including the effect of label noises, taxonomy of label noises, robust algorithms and noise cleaning algorithms for learning with noisy data, we refer to [63] for a comprehensive review.

Direct learning with noisy labels: Many studies have shown that label noises can adversely impact the classification accuracy of induced classifiers [64–66]. To better handle label noise, some approaches rely on training classifiers with label noise-robust algorithms [67, 68]. However, Bartlett *et al.* [69] prove that most of the loss functions are not completely robust to label noise. Experiments in [70] show that the classifiers inferred by label noise-robust algorithms are still affected by label noise. These methods seem to be adequate only when label noise can be safely managed by overfitting avoidance [63]. On the other hand, some label noise cleansing methods were proposed to remove or correct mislabeled instances [71–73], but these approaches were difficult in distinguishing informative hard examples from harmful mislabeled ones. Thus they might remove too many instances and the overcleansing could reduce the performances of classifiers [74].

Noise modeling with deep learning: Various methods have been proposed to handle label noise in different problem settings, but there are very few works about deep learning from noisy labels [1, 75, 76]. Mnih and Hinton [75] built a simple noise model for aerial images but only considered binary classification. Larsen *et al.* [76] assumed label noises are independent from true class labels which is a simple and special case. Sukhbaatar *et al.* [1] generalized from them by considering multi-class classification and modeling class dependent noise, but they assumed the noise was conditionally independent with the image content, ignoring the hardness of labeling images of different confusing levels. Our work can be viewed as a generalization of [1, 77]

and our model is flexible enough to not only class dependent but also image dependent noise.

Although annotating all the data is costly, it is often easy to obtain a small amount of clean labels, which enables to use some existing semi-supervised learning or transfer learning methods as introduced below.

Semi-supervised learning: Apart from direct learning with label noise, some semi-supervised learning algorithms were developed to utilize weakly labeled or even unlabeled data. The Label Propagation method [78] explicitly used ground truths of well labeled data to classify unlabeled samples. However, it suffered from computing pairwise distance, which has quadratic complexity with the number of samples thus cannot be applied on large-scale datasets. Weston *et al.* [79] proposed to embed a pairwise loss in the middle layer of a deep neural network, which benefits the learning of discriminative features. But they needed extra information about whether a pair of unlabeled images belong to the same class, which cannot be obtained in our problem.

Transfer learning: The success of CNNs lies in their capability of learning rich and hierarchical image features. However, the model parameters cannot be properly learned when training data is not enough. Researchers proposed to conquer this problem by first initializing CNN parameters with a model pretrained on a larger yet related dataset, and then finetuning it on the smaller dataset of specific task [7, 80–82]. Nevertheless, this transfer learning scheme could be suboptimal when the two tasks are just loosely related. In our case of clothing classification, we find that training a CNN from scratch with limited clean labels and massive noisy labels is better than finetuning it only on the clean labels.

4.2 Method Overview

Our goal is to build an end-to-end deep learning system that is capable of training with both limited clean labels and massive noisy labels more effectively. Figure 4.1 shows the framework of our approach. We collect $1,000,000$ clothing images from online shopping websites. Each image is automatically assigned with a noisy label according to the keywords in its surrounding text. We manually refine $72,409$ image labels, which constitute a clean sub-dataset. All the data are then used to train CNNs, while the major challenge is to identify and correct wrong labels during the training process.

Figure 4.1: *Overview of our approach. Labels of web images often suffer from different types of noise. A label noise model is proposed to detect and correct the wrong labels. The corrected labels are used to train underlying CNNs.*

To cope with this challenge, we extend CNNs with a novel probabilistic model, which infers the true labels and uses them to supervise the training of the network. Our work is inspired by [1], which modifies a CNN by inserting a linear layer on top of the softmax layer to map clean labels to noisy labels. However, [1] assumes noisy labels are conditionally independent of input images given clean labels. However, when examining our collected dataset, we find that this assumption is too strong to fit the real-world data well. For example, in Figure 4.2, all the images should belong to "Hoodie". The top five are correct while the bottom five are either mislabeled as "Windbreaker" or "Jacket". Since different sellers have their own bias on different categories, they may provide wrong keywords for similar clothes. We observe these visual patterns and

Figure 4.2: *Mislabeled images often share similar visual patterns.*

hypothesize that they are important to estimate how likely an image is mislabeled. Based on these observations, we further introduce two types of label noise:

- **Confusing noise** makes the noisy label reasonably wrong. It usually occurs when the image content is confusing (*e.g.*, the samples with "?" in Figure 4.1).
- **Pure random noise** makes the noisy label totally wrong. It is often caused by either the mismatch between an image and its surrounding text, or the false conversion from the text to label (*e.g.*, the samples with "×" in Figure 4.1).

Our proposed probabilistic model captures the relations among images, noisy labels, ground truth labels, and noise types, where the latter two are treated as latent variables. We use the Expectation-Maximization (EM) algorithm to solve the problem and integrate it into the training process of CNNs. Experiments on our real-world clothing dataset indicate that our model can better detect and correct the noisy labels.

4.3 Label Noise Model

We target on learning a classifier from a set of images with noisy labels. To be specific, we have a noisy labeled dataset $\mathcal{D}_\eta = \left\{\left(\mathbf{x}^{(1)}, \tilde{y}^{(1)}\right), \ldots, \left(\mathbf{x}^{(N)}, \tilde{y}^{(N)}\right)\right\}$ with n-th image $\mathbf{x}^{(n)}$ and its corresponding noisy label $\tilde{y}^{(n)} \in \{1, \ldots, L\}$, where L is the number of classes. We describe how the noisy label is generated by using a probabilistic graphical model shown in Figure 4.3.

Despite the observed image $\mathbf{x}$ and the noisy label $\tilde{\mathbf{y}}$, we exploit two discrete latent

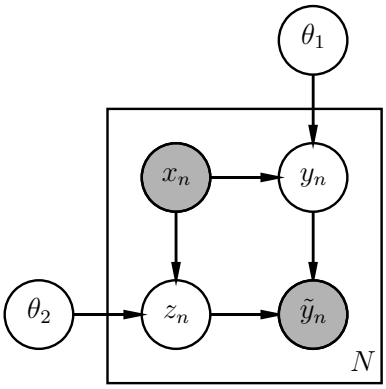


Figure 4.3: *Probabilistic graphical model of label noise*

variables — $\mathbf{y}$ and $\mathbf{z}$ — to represent the true label and the label noise type, respectively. Both $\tilde{\mathbf{y}}$ and $\mathbf{y}$ are L-dimensional binary random variables in 1-of-L fashion, *i.e.*, only one element is equal to 1 while others are all 0.

The label noise type $\mathbf{z}$ is an 1-of-3 binary random variable. It is associated with three semantic meanings:

1. The label is noise free, *i.e.*, $\tilde{\mathbf{y}}$ should be equal to $\mathbf{y}$.

2. The label suffers from a pure random noise, *i.e.*, $\tilde{\mathbf{y}}$ can take any possible value other than $\mathbf{y}$.

3. The label suffers from a confusing noise, *i.e.*, $\tilde{\mathbf{y}}$ can take several values that are confusing with $\mathbf{y}$.

Following this assignment rule, we define the conditional probability of the noisy label as

$$p(\tilde{\mathbf{y}}|\mathbf{y},\mathbf{z}) = \begin{cases} \tilde{\mathbf{y}}^T\mathbf{I}\mathbf{y} & \text{if } \mathbf{z}_1 = 1 \\ \frac{1}{L-1}\tilde{\mathbf{y}}^T(\mathbf{U}-\mathbf{I})\mathbf{y} & \text{if } \mathbf{z}_2 = 1 \\ \tilde{\mathbf{y}}^T\mathbf{C}\mathbf{y} & \text{if } \mathbf{z}_3 = 1, \end{cases} \tag{4.1}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{U}$ is the unit matrix (all the elements are ones), $\mathbf{C}$ is a sparse stochastic matrix with $\text{tr}(\mathbf{C}) = 0$ and $\mathbf{C}_{ij}$ denotes the confusion probability between classes i and j. Then we can derive from Figure 4.3 the joint distribution of

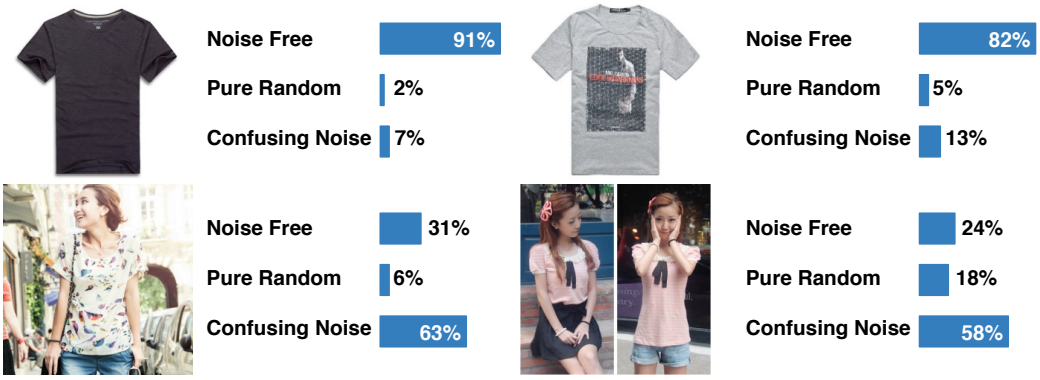


Figure 4.4: *Predicting noise types of four different "T-shirt" images. The top two can be recognized with little ambiguity, while the bottom two are easily confusing with the class "Chiffon". Image content can affect the possibility of it to be mislabeled.*

$\tilde{\mathbf{y}}, \mathbf{y}$ and $\mathbf{z}$ conditioning on $\mathbf{x}$,

$$p(\tilde{\mathbf{y}}, \mathbf{y}, \mathbf{z}|\mathbf{x}) = p(\tilde{\mathbf{y}}|\mathbf{y}, \mathbf{z})p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{x}). \tag{4.2}$$

While the class label probability distribution $p(\mathbf{y}|\mathbf{x})$ is comprehensible, the semantic meaning of $p(\mathbf{z}|\mathbf{x})$ needs extra clarification: it represents how confusing the image content is. Specific to our clothing classification problem, $p(\mathbf{z}|\mathbf{x})$ can be affected by different factors, including background clutter, image resolution, the style and material of the clothes. Some examples are shown in Figure 4.4.

To illustrate the relations between noisy and true labels, we derive their conditional probability from Eq. 4.2,

$$p(\tilde{\mathbf{y}}|\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{z}} p(\tilde{\mathbf{y}}, \mathbf{z}|\mathbf{y}, \mathbf{x}) = \sum_{\mathbf{z}} p(\tilde{\mathbf{y}}|\mathbf{y}, \mathbf{z})p(\mathbf{z}|\mathbf{x}), \tag{4.3}$$

which can be interpreted as a mixture model. Given an input image $\mathbf{x}$, the conditional probability $p(\mathbf{z}|\mathbf{x})$ can be seen as the prior of each mixture component. This makes a key difference between our work and [1], where they assume $\tilde{\mathbf{y}}$ is conditionally independent with $\mathbf{x}$ if $\mathbf{y}$ is given. All the images share a same noise model in [1], while in our approach each data sample has its own.

4.3.1 Learning the Parameters

We exploit two CNNs to model $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ separately. Denote the parameter set of each CNN by θ_1 and θ_2. Our goal is to find the optimal $\theta = \theta_1 \cup \theta_2$ that maximizes

the incomplete log-likelihood $\log p(\tilde{\mathbf{y}}|\mathbf{x};\theta)$. The EM algorithm is used to iteratively solve this problem.

For any probability distribution $q(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x})$, we can derive a lower bound of the incomplete log-likelihood,

$$\begin{aligned}\log p(\tilde{\mathbf{y}}|\mathbf{x};\theta) &= \log \sum_{\mathbf{y},\mathbf{z}} p(\tilde{\mathbf{y}},\mathbf{y},\mathbf{z}|\mathbf{x};\theta) \\ &\geq \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x}) \log \frac{p(\tilde{\mathbf{y}},\mathbf{y},\mathbf{z}|\mathbf{x};\theta)}{q(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x})}.\end{aligned} \tag{4.4}$$

E-Step The difference between $\log p(\tilde{\mathbf{y}}|\mathbf{x};\theta)$ and its lower bound is the Kullback-Leibler divergence $\mathrm{KL}\left(q(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x})||p(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta)\right)$, which is equal to zero if and only if $q(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x}) = p(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta)$. Therefore, in each iteration t, we first compute the posterior of latent variables using the current parameters $\theta^{(t)}$,

$$\begin{aligned}p(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta^{(t)}) &= \frac{p(\tilde{\mathbf{y}},\mathbf{y},\mathbf{z}|\mathbf{x};\theta^{(t)})}{p(\tilde{\mathbf{y}}|\mathbf{x};\theta^{(t)})} \\ &= \frac{p(\tilde{\mathbf{y}}|\mathbf{y},\mathbf{z};\theta^{(t)})p(\mathbf{y}|\mathbf{x};\theta^{(t)})p(\mathbf{z}|\mathbf{x};\theta^{(t)})}{\sum_{\mathbf{y}',\mathbf{z}'} p(\tilde{\mathbf{y}}|\mathbf{y}',\mathbf{z}';\theta^{(t)})p(\mathbf{y}'|\mathbf{x};\theta^{(t)})p(\mathbf{z}'|\mathbf{x};\theta^{(t)})}.\end{aligned} \tag{4.5}$$

Then the expected complete log-likelihood can be written as

$$Q(\theta;\theta^{(t)}) = \sum_{\mathbf{y},\mathbf{z}} p(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta^{(t)}) \log p(\tilde{\mathbf{y}},\mathbf{y},\mathbf{z}|\mathbf{x};\theta). \tag{4.6}$$

M-Step We exploit two CNNs to model the probability $p(\mathbf{y}|\mathbf{x};\theta_1)$ and $p(\mathbf{z}|\mathbf{x};\theta_2)$, respectively. Thus the gradient of Q w.r.t. θ can be decoupled into two parts:

$$\begin{aligned}\frac{\partial Q}{\partial \theta} &= \sum_{\mathbf{y},\mathbf{z}} p(\mathbf{y},\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta^{(t)}) \frac{\partial}{\partial \theta} \log p(\tilde{\mathbf{y}},\mathbf{y},\mathbf{z}|\mathbf{x};\theta) \\ &= \sum_{\mathbf{y}} p(\mathbf{y}|\tilde{\mathbf{y}},\mathbf{x};\theta^{(t)}) \frac{\partial}{\partial \theta_1} \log p(\mathbf{y}|\mathbf{x};\theta_1) + \\ &\qquad \sum_{\mathbf{z}} p(\mathbf{z}|\tilde{\mathbf{y}},\mathbf{x};\theta^{(t)}) \frac{\partial}{\partial \theta_2} \log p(\mathbf{z}|\mathbf{x};\theta_2).\end{aligned} \tag{4.7}$$

The M-Step above is equivalent to minimizing the cross entropy between the estimated ground truth distribution and the prediction of the classifier.

4.3.2 Estimating Matrix C

Notice that we do not set parameters to the conditional probability $p(\tilde{\mathbf{y}}|\mathbf{y}, \mathbf{z})$ in Eq. (4.1) and keep it unchanged during the learning process. Because without other regularizations, learning all the three parts in Eq. (4.2) could lead to trivial solutions. For example, the network will always predict $\mathbf{y}_1 = 1$, $\mathbf{z}_3 = 1$, and the matrix $\mathbf{C}$ is learned to make $\mathbf{C}_{1i} = 1$ for all i. To avoid such degeneration, we estimate $\mathbf{C}$ on a relatively small dataset $\mathcal{D}_c = \{(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{y}})\}_N$, where we have N images with both clean and noisy labels. As prior information about $\mathbf{z}$ is not available, we solve the following optimization problem:

$$\max_{\mathbf{C}, \mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}} \sum_{i=1}^{N} \log p(\tilde{\mathbf{y}}^{(i)}|\mathbf{y}^{(i)}, \mathbf{z}^{(i)}). \tag{4.8}$$

Obviously, sample i contributes nothing to the optimal $\mathbf{C}^*$ if $\mathbf{y}^{(i)}$ and $\tilde{\mathbf{y}}^{(i)}$ are equal. So that we discard those samples and reinterpret the problem in another form by exploiting Eq. (4.1):

$$\begin{aligned} \max_{\mathbf{C}, \mathbf{t}} \quad & E = \sum_{i=1}^{N'} \log \alpha^{\mathbf{t}_i} + \log(\tilde{\mathbf{y}}^{(i)T} \mathbf{C} \mathbf{y}^{(i)})^{1-\mathbf{t}_i}, \\ \text{subject to} \quad & \mathbf{C} \text{ is a stochastic matrix of size } L \times L, \\ & \mathbf{t} \in \{0, 1\}^{N'}, \end{aligned} \tag{4.9}$$

where $\alpha = \frac{1}{L-1}$ and N' is the number of remaining samples. The semantic meaning of the above formulation is that we need to assign each $(\mathbf{y}, \tilde{\mathbf{y}})$ pair the optimal noise type, while finding the optimal $\mathbf{C}$ simultaneously.

Next, we will show that the problem can be solved by a simple yet efficient algorithm in $O(N' + L^2)$ time complexity. Denote the optimal solution by $\mathbf{C}^*$ and $\mathbf{t}^*$

Theorem 4.1. $\mathbf{C}^*_{ij} \neq 0 \Rightarrow \mathbf{C}^*_{ij} > \alpha, \forall i, j \in \{1, \ldots, L\}$.

Proof. Suppose there exists some i, j such that $0 < \mathbf{C}^*_{ij} \leq \alpha$. Then we conduct the following operations. First, we set $\mathbf{C}^*_{ij} = 0$ while adding its original value to other elements in column j. Second, for all the samples n where $\tilde{\mathbf{y}}^{(n)}_i = 1$ and $\mathbf{y}^{(n)}_j = 1$, we set $\mathbf{t}_n$ to 1. The resulting E will get increased, which leads to a contradiction. □

Theorem 4.2. $(\tilde{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) = (\tilde{\mathbf{y}}^{(j)}, \mathbf{y}^{(j)}) \Rightarrow \mathbf{t}^*_i = \mathbf{t}^*_j, \forall i, j \in \{1, \ldots, N'\}$.

Proof. Suppose $\tilde{\mathbf{y}}^{(i)}_k = \tilde{\mathbf{y}}^{(j)}_k = 1$ and $\mathbf{y}^{(i)}_l = \mathbf{y}^{(j)}_l = 1$ but $\mathbf{t}^*_i \neq \mathbf{t}^*_j$. From Theorem 1 we know that elements in $\mathbf{C}^*$ is either 0 or greater than α. If $\mathbf{C}^*_{kl} = 0$, we can set

$\mathbf{t}_i^* = \mathbf{t}_j^* = 1$, otherwise we can set $\mathbf{t}_i^* = \mathbf{t}_j^* = 0$. In either case the resulting E will get increased, which leads to a contradiction. □

Theorem 4.3. $\tilde{\mathbf{y}}^{(i)T}\mathbf{C}^*\mathbf{y}^{(i)} > \alpha \Leftrightarrow \mathbf{t}_i^* = 0$ *and* $\tilde{\mathbf{y}}^{(i)T}\mathbf{C}^*\mathbf{y}^{(i)} = 0 \Leftrightarrow \mathbf{t}_i^* = 1, \forall i \in \{1, \ldots, N'\}$.

Proof. The first part is straightforward. For the second part, $\mathbf{t}_i^* = 1$ implies $\tilde{\mathbf{y}}^{(i)T}\mathbf{C}^*\mathbf{y}^{(i)} \leq \alpha$. By using Theorem 1 we know that $\tilde{\mathbf{y}}^{(i)T}\mathbf{C}^*\mathbf{y}^{(i)} = 0$. □

Notice that if the true label is class i while the noisy label is class j, then it can only affect the value of $\mathbf{C}_{ij}$. Thus each column of $\mathbf{C}$ can be optimized separately. Theorem 1 further shows that samples with same pair of $(\tilde{\mathbf{y}}, \mathbf{y})$ share a same noise type. Then what really matters is the frequencies of all the $L \times L$ pairs of $(\tilde{\mathbf{y}}, \mathbf{y})$. Considering a particular column $\mathbf{c}$, suppose there are M samples affecting this column. We can count the frequencies of noisy label class 1 to L as $m_1, \ldots, m_L$ and might as well assume $m_1 \geq m_2 \geq \cdots \geq m_L$. The problem is then converted to

$$\begin{aligned} \max_{\mathbf{c},\mathbf{t}} \quad & E = \sum_{k=1}^{L} m_k \left(\log \alpha^{\mathbf{t}_k} + \log \mathbf{c}_k^{1-\mathbf{t}_k} \right), \\ \text{subject to} \quad & \mathbf{c} \in [0,1]^L, \sum_{k=1}^{L} \mathbf{c}_k = 1, \\ & \mathbf{t} \in \{0,1\}^L. \end{aligned} \tag{4.10}$$

Due to the rearrangement inequality, we can prove that in the optimal solution,

$$\max(\alpha, \mathbf{c}_1^*) \geq \max(\alpha, \mathbf{c}_2^*) \geq \cdots \geq \max(\alpha, \mathbf{c}_L^*). \tag{4.11}$$

Then by using Theorem 3, there must exist a $k^* \in \{1, \ldots, L\}$ such that

$$\begin{aligned} & \mathbf{t}_i^* = 0, i = 1, \ldots, k^*, \\ & \mathbf{t}_i^* = 1, i = k^* + 1, \ldots, L. \end{aligned} \tag{4.12}$$

This also implies that only the first k^* elements of $\mathbf{c}^*$ have nonzero values (greater than α actually). Furthermore, if k^* is known, finding the optimal $\mathbf{c}^*$ is to solve the following

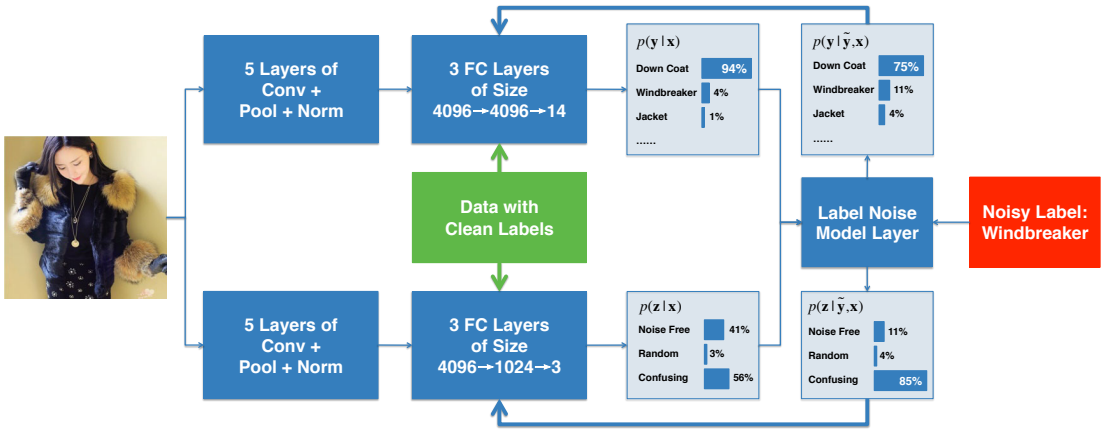


Figure 4.5: *System diagram of our method. Two CNNs are used to predict the class label* $p(\mathbf{y}|\mathbf{x})$ *and the noise type* $p(\mathbf{z}|\mathbf{x})$*, respectively. The label noise model layer uses both these predictions and the given noisy label to estimate a posterior distribution of the true label, which is then used to supervise the training of CNNs. Data with clean labels are also mixed in to prevent the models from drifting away.*

problem:

$$\begin{aligned} \max_{\mathbf{c}} \quad & E = \sum_{k=1}^{k^*} m_k \log \mathbf{c}_k, \\ \text{subject to} \quad & \mathbf{c} \in [0,1]^L, \sum_{k=1}^{k^*} \mathbf{c}_k = 1, \end{aligned} \tag{4.13}$$

whose solution is

$$\begin{aligned} \mathbf{c}_i^* &= \frac{m_i}{\sum_{k=1}^{k^*} m_k}, i = 1, \ldots, k^*, \\ \mathbf{c}_i^* &= 0, i = k^* + 1, \ldots, L. \end{aligned} \tag{4.14}$$

The above analysis leads to a simple algorithm. We enumerate k^* from 1 to L. For each k^*, $\mathbf{t}^*$ and $\mathbf{c}^*$ are computed by using Eq. (4.12) and (4.14), respectively. Then we evaluate the objective function E and record the best solution.

4.4 Deep Learning from Noisy Labels

We integrate the proposed label noise model into a deep learning framework. As demonstrated in Figure 4.5, we predict the probability $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ by using two independent CNNs. Moreover, we append a label noise model layer at the end, which takes as input the CNNs' prediction scores and the observed noisy label. Stochastic Gradient Ascent with backpropagation technique is used to approximately optimize the whole network. In each forward pass, the label noise model layer computes the posterior of

latent variables according to Eq. (4.5). While in the backward pass, it computes the gradients according to Eq. (4.7).

Directly training the whole network with random initialization is impractical, because the posterior computation could be totally wrong. Therefore, we need to pretrain each CNN component with strongly supervised data. Images and their ground truth labels in the dataset $\mathcal{D}_c$ are used to train the CNN that predicts $p(\mathbf{y}|\mathbf{x})$. On the other hand, the optimal solutions of $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$ in Eq. (4.8) are used to train the CNN that predicts $p(\mathbf{z}|\mathbf{x})$.

After both CNN components are properly pretrained, we can start to train the whole network with massive noisy labeled data. However, some practical issues need further discussion. First, if we merely use noisy labels, we will lose precious knowledge that we have gained before and the model could be drifted. Therefore, we need to mix the data with clean labels into our training set, which is depicted in Figure 4.5 as the extra supervisions for the two CNNs. Then each CNN receives two kinds of gradients, one is from the clean labels and the other is from the noisy labels. We denote them by Δ_c and Δ_n, respectively. A potential problem is that $|\Delta_c| \ll |\Delta_n|$, because clean data is much less than noisy data. To deal with this problem, we bootstrap the clean data $\mathcal{D}_c$ to half amount of the noisy data $\mathcal{D}_\eta$. In our experiments, we find that the performance of the classifier drops significantly without upsampling, but it is not sensitive with the upsampling ratio as long as the number of clean and noisy samples are in the same order.

Our proposed method has the ability to figure out the true label given the image and its noisy label. From the perspective of information, our model predicts from two kinds of clues: what are the true labels for other similar images; and how likely the image is mislabeled. The Label Propagation method [78] explicitly uses the first kind of information, while we implicitly capture it with a discriminative deep model. On the other hand, the second kind of information correlates the image content with the label noise, which can help distinguish between hard samples and mislabeled ones.

4.5 Experiments

4.5.1 Dataset

Since there is no publicly available dataset that has both clean and noisy labels, to test our method under real-world scenario, we build a large-scale clothing dataset by crawling images from several online shopping websites. More than a million images are collected together with their surrounding texts provided by the sellers. These surrounding texts usually contain rich information about the products, which can be further converted to visual tags. Specific to our task of clothing classification, we define 14 class labels: T-shirt, Shirt, Knitwear, Chiffon, Sweater, Hoodie, Windbreaker, Jacket, Down Coat, Suit, Shawl, Dress, Vest, and Underwear. We assign an image a noisy label if we find its surrounding text contains only the keywords of that label, otherwise we discard the image to reduce ambiguity.

After that we manually refine the noisy labels of a small portion of all the images and split them into training ($\mathcal{D}_c$), validation and test sets. On the other hand, the remaining samples construct the noisy labeled training dataset ($\mathcal{D}_\eta$). A crucial step here is to remove from $\mathcal{D}_c$ and $\mathcal{D}_\eta$ the images that are near duplicate with any image in the validation or test set, which ensures the reliability of our test protocol. Finally, the size of training datasets are $|\mathcal{D}_c| = 47,570$ and $|\mathcal{D}_\eta| = 10^6$, while validation and test set have $14,313$ and $10,526$ images, respectively.

The confusion matrix between clean and noisy labels is presented in Figure 4.6. We can see that the overall accuracy is 61.54%, and some pairs of classes are very confusing with each other (*e.g.* Knitwear and Sweater), which means that the noisy labels are not so reliable.

4.5.2 Evaluation on the Collected Dataset

We validate our method through a series of experiments conducted on the collected dataset. Our implementation is based on Caffe [83], and the `bvlc_reference_caffenet`[1] is chosen as the baseline model, which approximates AlexNet [7]. Besides, we reimplement two other approaches. One is a semi-supervised learning method called Pseudo-Label [84], which exploits classifier's prediction as ground truth for unlabeled data. The other one is the Bottom-Up method introduced in [1], where the relation between

[1] http://caffe.berkeleyvision.org/model_zoo.html

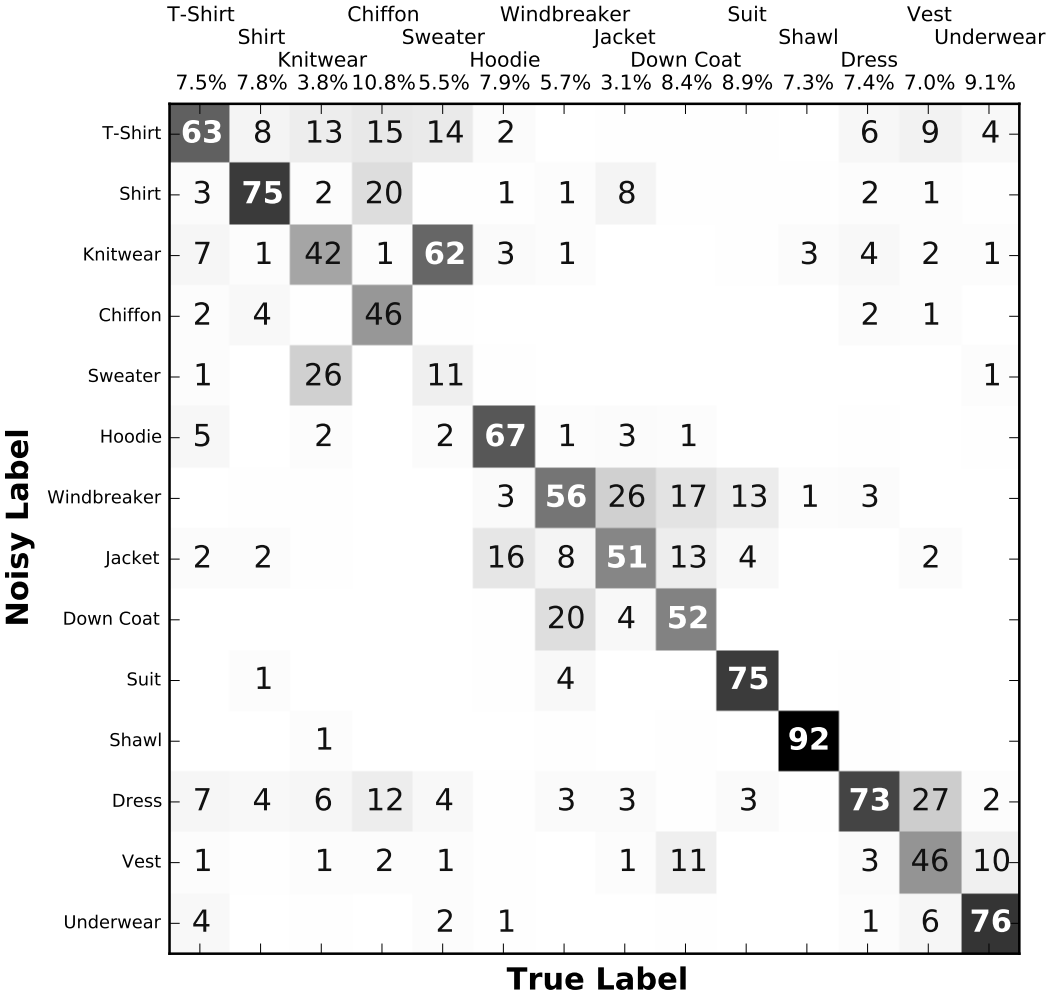


Figure 4.6: *Confusion matrix between clean and noisy labels. We hide extremely small grid numbers for better demonstration. Frequency of each true label is listed at the top of each column. The overall accuracy is* 61.54%*, which indicates that the noisy labels are not reliable.*

noisy labels and clean labels are built by a confusion matrix Q. In the experiments, we directly use the true Q as shown in Figure 4.6 instead of estimating its values.

We list all the experiment settings in Table 4.1. Different methods require different training data. We use only the clean data $\mathcal{D}_c$ to get the baselines under strong supervisions. On the other hand, when all the data are used, we upsample the clean data as discussed in Section 4.4. Meanwhile, the noisy labels of $\mathcal{D}_\eta$ are treated as true labels for AlexNet, and are discarded for Pseudo-Label.

In general, we use a mini-batch size of 256. The learning rate is initialized to be

#	Method	Training Data	Initialization	Test Accuracy
1	AlexNet	$\mathcal{D}_c$	random	64.54%
2	AlexNet	$\mathcal{D}_c$	ImageNet	72.63%
3	AlexNet	$\mathcal{D}_c$ and $\mathcal{D}_\eta$ as GT	random	74.03%
4	AlexNet	$\mathcal{D}_c$ and $\mathcal{D}_\eta$ as GT	ImaegNet	75.13%
5	AlexNet	$\mathcal{D}_c$ and $\mathcal{D}_\eta$ as GT	model #2	75.30%
6	Pseudo-Label [84]	$\mathcal{D}_c$ and $\mathcal{D}_\eta$ as unlabeled	model #2	73.04%
7	Bottom-Up [1]	$\mathcal{D}_c$ and $\mathcal{D}_\eta$	model #2	76.22%
8	Ours	$\mathcal{D}_c$ and $\mathcal{D}_\eta$	model #2	**78.24%**

Table 4.1: *Experimental results on the clothing classification dataset. $\mathcal{D}_c$ contains 47,570 clean labels while $\mathcal{D}_\eta$ contains 10^6 noisy labels.*

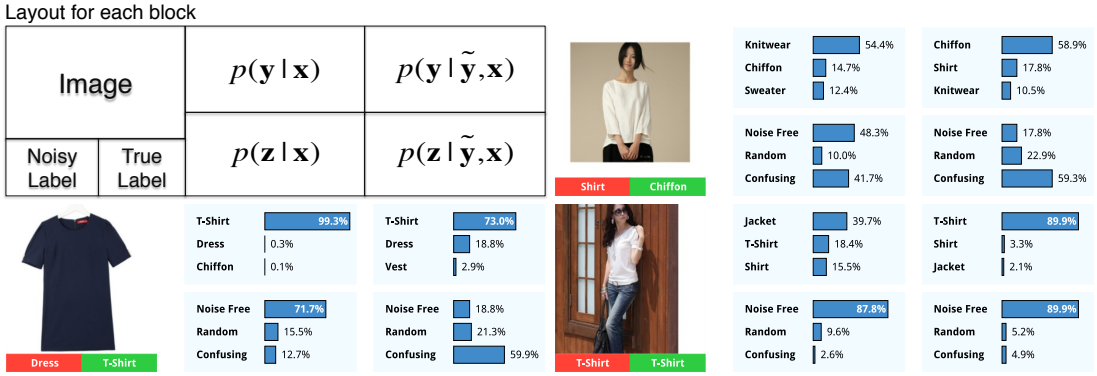


Figure 4.7: *Examples of handling noisy labels. The information layout for each block is illustrated on the top-left. $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ are predictions of the true label and noise type based on image content. After observing the noisy label, our model infers the posterior distributions $p(\mathbf{y}|\mathbf{y},\mathbf{x})$ and $p(\mathbf{z}|\mathbf{y},\mathbf{x})$, then replaces the $\mathbf{y}$ and $\mathbf{z}$ with them as supervisions to the CNNs.*

0.001 and is divided by 10 after every 50,000 iterations. We keep training each model until convergence. Classification accuracies on the test set are presented in Table 4.1.

We first study the effect of transfer learning and massive noisy labeled data. From row #1 we can see that training a CNN from scratch with only small amount of clean data can result in bad performance. To deal with this problem, finetuning from an ImageNet pretrained model can significantly improve the accuracy, as shown in row #2. However, by comparing row #2 and #3, we find that training with random initialization on additional massive noisy labeled data is better than finetuning only on the clean data, which demonstrates the power of using large-scale yet easily obtained noisy labeled data. The accuracy can be further improved if we finetune the model either from an ImageNet pretrained one or model #2. The latter one has slightly

Noise Level	CIFAR10-quick	[1]	Ours
30%	65.57%	69.73%	69.81%
40%	62.38%	66.66%	66.76%
50%	57.36%	63.39%	63.00%

Table 4.2: *Accuracies on CIFAR-10 with synthetic label noises. The label noises generated here only depend on the true labels but not the image content, which exactly match the assumption of [1] but are unfavored to our model.*

better performance thus is used to initialize the remaining methods.

Next, from row #6 we can see that semi-supervised learning methods may not be a good choice when massive noisy labeled data are available. Although model #6 achieves marginally better result than its base model, it is significantly inferior to model #5, which indicates that simply discarding all the noisy labels cannot fully utilize these information.

Finally, row #7 and #8 show the effect of modeling the label noise. Model #7 is only 0.9% better than the baseline model #5, while our method gains improvement of 2.9%. This result does credit to our image dependent label noise model, which fits better to the noisy labeled data crawled from the Internet.

4.5.3 Evaluation on CIFAR-10 with Synthetic Noises

We also conduct synthetic experiments on CIFAR-10 following the settings of [1]. We first randomly generate a confusion matrix Q between clean labels and noisy labels, and then corrupt the training labels according to it. Based on Caffe's CIFAR10-quick model, we compare the [1] (Bottom Up with true Q) with our model under different noise levels. The test accuracies are reported in Table 4.2.

It should be noticed that [1] assumed the distribution of noisy labels only depends on classes, while we assume it also depends on image content. Label noises generated in the synthetic experiments exactly match their assumption but are unfavored to our model. Thus the noise type predictions in our model could be less informative. Nevertheless, our model achieves comparable performances with [1].

4.5.4 Effect of Noise Estimation

In order to understand how our model handles noisy labels, we first show several examples in Figure 4.7. We find that given a noisy label, our model exploits its current

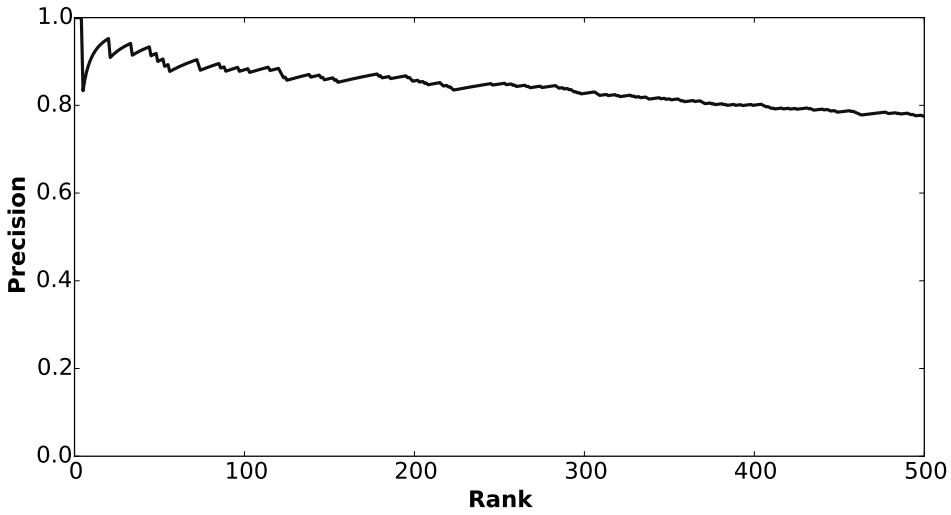


Figure 4.8: *Rank-precision curve of label noise predictions. We rank the validation images from low to high according to their "noise free" probabilities. For the precision calculation, we consider a candidate image as true positive if its clean label mismatches its original noisy label, and our model predicts it as not "noisy free".*

knowledge to estimate the probability distribution of the true label and then replaces the noisy one with it as supervision. Another interesting observation is that our model can still figure out the correct label if the prediction of the class label $p(\mathbf{y}|\mathbf{x})$ or the noise type $p(\mathbf{z}|\mathbf{x})$ goes wrong. These two latent variables compensate with each other to help the system distinguish between hard samples and noisy labeled ones.

Next we demonstrate the effectiveness of learning to predict the label noise type. Notice that if an image has low probability of "noise free" (*i.e.*, $p(\mathbf{z}_1 = 1|\mathbf{x})$ is small), then our model will believe it is likely to be mislabeled. In order to check the reliability of these predictions, we estimate $p(\mathbf{z}_1 = 1|\mathbf{x})$ on the validation set and sort the images accordingly in ascending order. For the precision calculation, we consider a candidate image as true positive if its clean label mismatches its original noisy label, and our model predicts it as not "noisy free". The rank-precision curve is plotted in Figure 4.8. It shows that our model can identify mislabeled samples based on their image content, which verifies the observation that mislabeled images often share similar visual patterns.

4.6 Conclusions

In this chapter, we address the problem of training a deep learning classifier with a massive amount of noisy labeled training data and a small amount of clean annotations which are generally easy to collect. To utilize both limited clean labels and massive noisy labels, we propose a novel probabilistic framework to describe how noisy labels are produced. We introduce two latent variables, the true label and the noise type, to bridge the gap between an input image and its noisy label. We observe that the label noises not only depend on the ambiguity between classes, but could follow similar visual patterns as well. We build the dependency of the noise type w.r.t. images, and infer the ground truth label with the EM algorithm. We integrate the probabilistic model into a deep learning framework and demonstrate the effectiveness of our method on a large-scale real-world clothing classification dataset.

Chapter 5

Multi-Domain Person Re-identification

Learning generic and robust feature representations with data from multiple domains for the same problem is of great value, especially for the problems that have multiple datasets but none of them are large enough to provide abundant data variations. In this chapter, we present a pipeline for learning deep feature representations from multiple domains with CNNs. When training a CNN with data from all the domains, some neurons learn representations shared across several domains, while some others are effective only for a specific one. Based on this important observation, we propose a Domain Guided Dropout algorithm to improve the feature learning procedure. Experiments show the effectiveness of our pipeline and the proposed algorithm. Our methods on the person re-identification problem outperform state-of-the-art methods on multiple datasets by large margins.

The contribution of this chapter is three-fold. First, we present a pipeline for learning generic feature representations from multiple domains that perform well on all of them. This enables us to learn better features from multiple datasets for the same problem. Second, we propose Domain Guided Dropout to discard useless neurons for each domain, which improves the performance of the CNN. At last, our method outperforms state-of-the-arts on multiple person re-identification datasets by large margins. We observe that learning feature representations by utilizing data from multiple datasets improve the performance significantly, and the largest gain is 46% on the PRID dataset. Extensive experiments validate our proposed method and the internal mechanism of the method is studied in details.

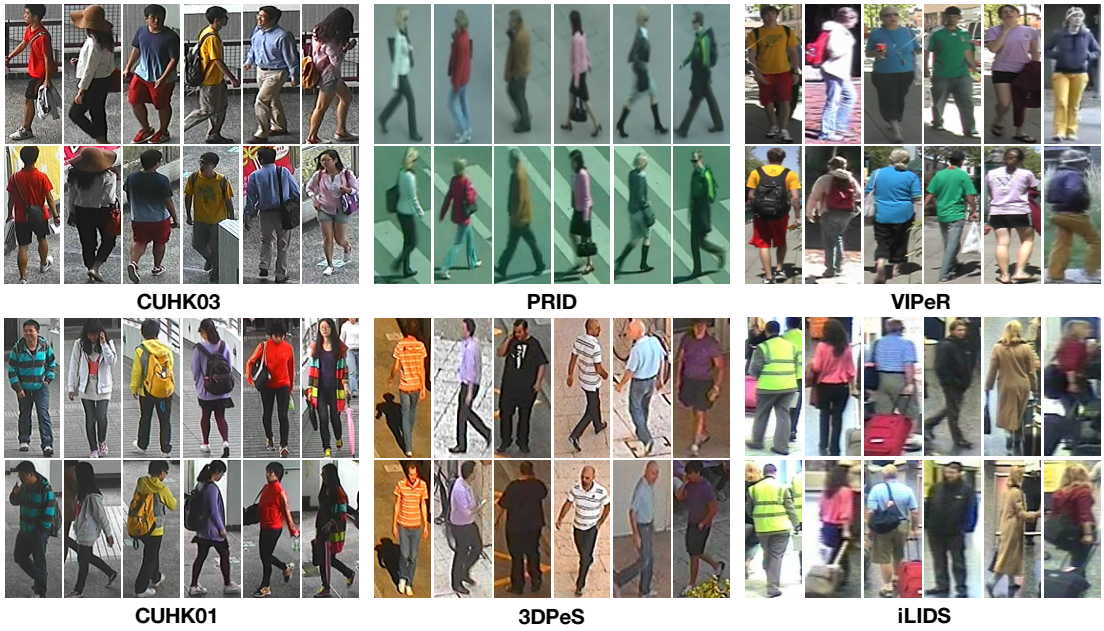


Figure 5.1: *Examples of multiple person re-identification datasets. Each dataset has its own bias. Our goal is to learn generic feature representations that are effective on all of them simultaneously.*

5.1 Related Work

In computer vision, a *domain* often refers to a dataset where samples follow the same underlying data distribution. It is common that multiple datasets with different data distributions are proposed to target the same or similar problems. Multi-domain learning aims to solve the problem with datasets across different domains simultaneously by using all the data they provide. As deep learning arises in the recent years, learning good feature representations achieves great success in many research fields and real-world applications. The success of deep learning is driven by the emergence of large-scale training data, which makes multi-domain learning an interesting problem. Many studies [80–82] have shown that fine-tuning a deep model pretrained on a large-scale dataset (*e.g.* ImageNet [11]) is effective for other related domains and tasks. However, in many specific areas, there is no such large-scale dataset for learning robust and generic feature representations. Nonetheless, different research groups have proposed many smaller datasets. It is necessary to develop an effective algorithm that jointly utilizes all of them to learn generic feature representations.

Another interesting aspect of multi-domain learning is that it enriches the data variety because of the domain discrepancies. Limited by various conditions, data collected by a research group might only include certain types of variations. Take the person re-identification [23,85] problem as an example, pedestrian images are usually captured in different scenes (*e.g.*, campus, markets, and streets), as shown in Figure 5.1. Images in CUHK01 [86] and CUHK03 [23] are captured on campus, where many students wear backpacks. PRID [87] contains pedestrians in street views, where crosswalks appear frequently in the dataset. Images in VIPeR [88] suffer from significant resolution changes across different camera views. Each of such datasets is biased and contains only a subset of possible data variations, which is not sufficient for learning generic feature representations. Combining them together can diversify the training data, thus makes the learned features more robust.

In recent years, training deep neural networks with multiple domains has been explored. Feature representations learned by Convolutional Neural Networks have shown their effectiveness in a wide range of visual recognition tasks [7,89–93]. Long *et al.* [94] incorporated the multiple kernel variant of Maximum Mean Discrepancy (MMD) objective for regularizing the training of neural networks. Ganin *et al.* [95] proposed to reduce the distribution mismatch between the source and target domains by reversing the gradients of the domain classification loss, which is also utilized by [96] with a soft-label matching loss to transfer task information. Most of these methods aim at finding a common feature space that is domain invariant. However, our approach allows the representation to have disjoint components that are domain specific, while also learning a shared representation.

As deep neural networks usually contain millions of parameters, it is of great importance to reduce the parameter space by adding regularizations to the weights. The quality of the regularization method would significantly affect both the discriminative power and generalization ability of the trained networks. Dropout [97] is one of the most widely used regularization method in training deep neural networks, which significantly improves the performance of the deep model [7]. During the network training process, Dropout randomly sets neuron responses to zero with a probability of 0.5. Thus a training batch updates only a subset of all the neurons at each time, which avoids co-adaptation of the learned feature representations.

While the standard Dropout algorithm treats all the neurons equally with a fixed probability, Ba *et al.* [98] proposed an adaptive dropout scheme by learning a binary belief network to predict the dropout probability for each neuron. In practice, they use the response of each neuron to compute the dropout probability for itself. Our approach significantly differs from this method, as we propose to train a CNN from multiple domains, and utilize the domain information to guide the dropout procedure.

For concrete demonstration, we target the person re-identification problem, but the method itself would be generalized to other problems with datasets of multiple domains. Existing Re-ID methods mainly address the problem from two aspects: finding more powerful feature representations and learning better metrics. Zhao *et al.* [41, 99, 100] proposed to combine SIFT features with color histogram as features. In deep learning literature, Li *et al.* [23] and Ahmed *et al.* [38] designed CNN models specifically to the Re-ID task and achieved good performance on large-scale datasets. They trained the network with pairs of pedestrian images and adopted the verification loss function. Ding *et al.* [101] utilized triplet samples for training features that maximize relative distance between the pair of same person and the pair of different people in the triplets. Apart from the feature learning methods, a large number of metric learning algorithms [33–36, 102, 103] have also been proposed to solve the Re-ID problem from a complementary perspective. Some recent works addressed the problem of mismatch between traditional Re-ID and real application scenarios. Liao *et al.* [104] proposed a database for open-set Re-ID. Zheng *et al.* [24] treated Re-ID as an image search problem and introduced a large-scale dataset. Xu *et al.* [25] raised the problem of searching a person inside whole images rather than cropped bounding boxes.

5.2 Method Overview

In this chapter, we present a framework for learning generic feature representations from multiple domains that are effective on all of them simultaneously. Our proposed pipeline consists of several stages, as shown in Figure 5.2. Since learning features from a large-scale classification dataset is proved to be effective [17], we first mix all the domains together and train a Convolutional Neural Network (CNN) to recognize person identities (IDs), which is formulated as a *Joint Single Task Learning* problem.

The CNN model we designed consists of several BN-Inception [9, 12] modules, and

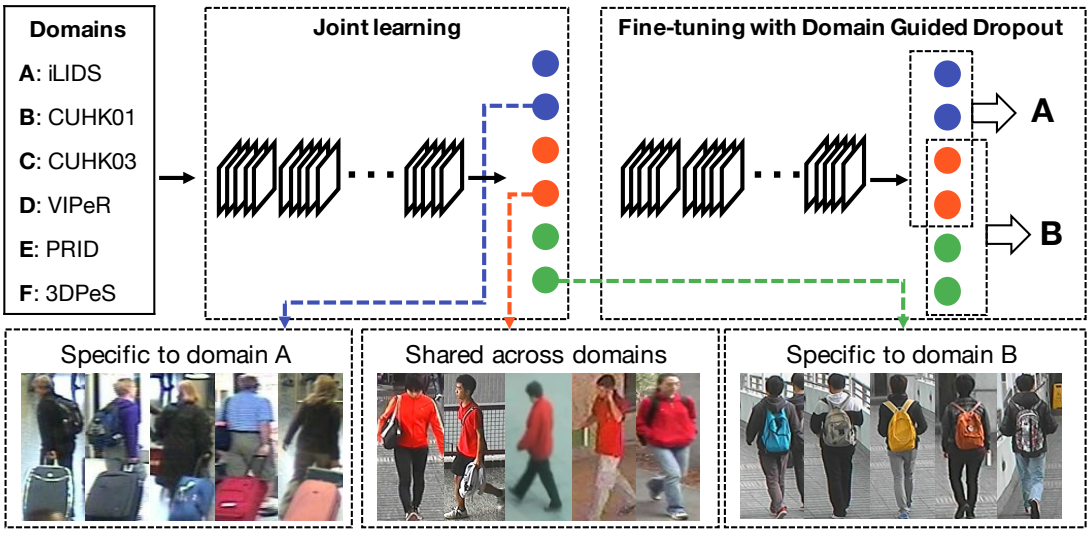


Figure 5.2: *Overview of our framework. For the person re-identification problem, we first train a CNN jointly on all six domains. Then we analyze the effectiveness of each neuron on each domain. For example, some may capture the luggages that only appear in domain A, while some others may capture the red clothes shared across different domains. We propose a Domain Guided Dropout algorithm to discard useless neurons for each domain during the training process, which drives the CNN to learn better feature representations on all the domains simultaneously.*

its capacity well fits to the scale of the mixed dataset. This carefully designed CNN model provides us a fairly strong baseline, but the simple joint learning scheme does not take full advantages of the variations of multiple domains.

We observed an interesting phenomenon for the units of the fully connected layers. Neurons that are effective for one domain could be useless for another domain because of the presence of domain biases. For example, in Figure 5.1 only the i-LIDS dataset contains pedestrians with luggages, thus the neurons that capture luggage features are of no use when recognizing people from other domains.

Based on this observation, we propose *Domain Guided Dropout* — a simple yet effective method of muting non-related neurons for each domain. Different from the standard Dropout [97], which treats all the neurons equally, our method assigns each neuron a specific dropout rate for each domain according to its effectiveness on that domain. To be specific, for each domain, we perform the forward pass on all its samples and compute for each neuron its average impact on the identification loss function. Then we replace the standard Dropout layer with a proposed deterministic Domain Guided Dropout layer, and continue to train the CNN model for several more epochs.

With the guidance of which neurons being effective for each domain, the CNN learns more discriminative features for all of them. At last, if we want to obtain feature representations for a specific domain, the CNN could be further fine-tuned on it, with a proposed stochastic Domain Guided Dropout to improve the performance.

In the following sections, we detail these stages, and compare our design choices with other alternatives.

5.3 Joint Single Task Learning

Although our framework itself is not limited to any specific scope, we target the person re-identification problem for concrete demonstration. This section formulates the problem, defines the objective functions, and introduces the network architecture.

5.3.1 Problem Formulation

The problem can be formulated as follows. Suppose we have D domains, each of which consists of N_i images of M_i different people. Let $\{(x_i^{(j)}, y_i^{(j)})_{j=1}^{N_i}\}_{i=1}^{D}$ denote all training samples, where $x_i^{(j)}$ is the j-th image of the i-th domain, and $y_i^{(j)} \in \{1, 2, \ldots, M_i\}$ is the identity of the corresponding person. Our goal is to learn a generic feature extractor $g(\cdot)$ that has similar outputs for images of the same person and dissimilar outputs for different people. During the test phase, given a probe pedestrian image and a set of gallery images, we use $g(\cdot)$ to extract features from all of them, and rank the gallery images according to their Euclidean distances to the probe image in the feature space. For the training phase, there are several frameworks that use pairwise [23,38] or triplet [105] inputs for learning feature embeddings. In our approach, we train a CNN to recognize the identity of each person, which is also adopted in the face verification work [17].

5.3.2 Objectives Functions

When mixing all the D domains together, a straightforward solution is to employ a multi-task objective function, *i.e.*, learning D softmax classifiers $f_1, f_2, \ldots, f_D$ and a shared features extractor g that minimize

$$\arg\min_{f_1, f_2, \ldots, f_D, g} \sum_{i=1}^{D} \sum_{j=1}^{N_i} \mathcal{L}\left(f_i(g(x_i^{(j)})), y_i^{(j)}\right), \tag{5.1}$$

where $\mathcal{L}$ is the softmax loss function that equals to the cross-entropy between the predicted probability vector and the ground truth.

However, since different person re-identification datasets usually have totally different identities, it is also safe to merge all $M = \sum_{i=1}^{D} M_i$ people together and relabel them with new IDs $y' \in \{1, 2, \ldots, M\}$. For the merged dataset, we can define a single-task objective function, *i.e.*, learning one softmax classifier f and the features extractor g that minimize

$$\arg\min_{f,g} \sum_{i=1}^{D} \sum_{j=1}^{N_i} \mathcal{L}\left((f \circ g)(x_i^{(j)}), {y'}_i^{(j)}\right). \tag{5.2}$$

Compared with the multi-task formulation, this single-task learning scheme forces the network to simultaneously distinguish people from all domains. The feature representations capture two types of information: domain biases (*e.g.*, background clutter, lighting, *etc.*) as well as person appearance and attributes. If the data distributions of two domains differ a lot, it would be easy to separate the persons of the two domains by observing only the domain biases. However, when these biases are not significant enough, the network is required to learn discriminative person-related features to make the decisions. Thus the single-task objective fits better to our setting and is chosen for this work.

5.3.3 Network Architecture

Since pedestrian images are usually quite small and are not of square-shapes, it is not appropriate to directly use the ImageNet pretrained CNN models, which are trained with object images of high resolution and abundant details. Thus we propose to design a network structure that well fits our problem scale. Inspired by [8, 12], we build a CNN with three preceding 3×3 convolutional layers followed by six Inception modules and two fully connected layers. Detailed structures are listed in Table 5.1. The Batch Normalization (BN) layers are employed before each ReLU layer, which accelerate the convergence process and avoid manually tweaking the initialization of weights and biases. For training the CNN from scratch, we randomly dropout 50% neurons of the fc7 layer. The initial learning rate is set to 0.1 and is decreased by 4% for every 4 epochs until it reaches 0.0005. The learning rate is then fixed at this value for a few more epochs until convergence.

name	kernel size/ stride	output size	#1×1	#3×3 reduce	#3×3	double #3×3 reduce	double #3×3	pool proj
input		$3 \times 144 \times 56$						
conv1	$3 \times 3/1$	$32 \times 144 \times 56$						
conv2	$3 \times 3/1$	$32 \times 144 \times 56$						
conv3	$3 \times 3/1$	$32 \times 144 \times 56$						
pool3	$2 \times 2/2$	$32 \times 72 \times 28$						
icp (4a)		$256 \times 72 \times 28$	32	32	32	32	32	avg 32
icp (4b)	stride 2	$384 \times 72 \times 28$	32	32	32	32	32	max pass
icp (5a)		$512 \times 36 \times 14$	64	64	64	64	64	avg 64
icp (5b)	stride 2	$768 \times 36 \times 14$	64	64	64	64	64	max pass
icp (6a)		$1024 \times 36 \times 14$	128	128	128	128	128	avg 128
icp (6b)	stride 2	$1536 \times 36 \times 14$	128	128	128	128	128	max pass
fc7		256						
fc8		M						

Table 5.1: *The structure of our proposed CNN for person re-identification*

5.4 Domain Guided Dropout

Given the CNN model pretrained by using the mixed dataset, we identify for each domain which neurons are effective. For each domain sample, we define the impact of a particular neuron on this sample as the gain of the loss function when we remove the neuron. Specifically, let $g(x) \in \mathbb{R}^d$ denote the d-dimensional CNN feature vector of an image x. The impact score of the i-th ($i \in \{1, 2, \ldots, d\}$) neuron on this image sample is defined as

$$s_i = \mathcal{L}(g(x)_{\setminus i}) - \mathcal{L}(g(x)), \tag{5.3}$$

where $g(x)_{\setminus i}$ is the feature vector after we setting the i-th neuron response to zero. For each domain $\mathcal{D}$, we then take the expectation of s_i over all its samples to obtain the averaged impact score $\bar{s}_i = \mathrm{E}_{x \in \mathcal{D}}[s_i]$. We visualize the neuron impact scores between several pairs of domains in Figure 5.3. It clearly shows that the two sets of impact scores have little correlation, indicating that the effective neurons for different domains are not the same.

A naive computation of all the impact values requires $O(d|\mathcal{D}|)$ network forward

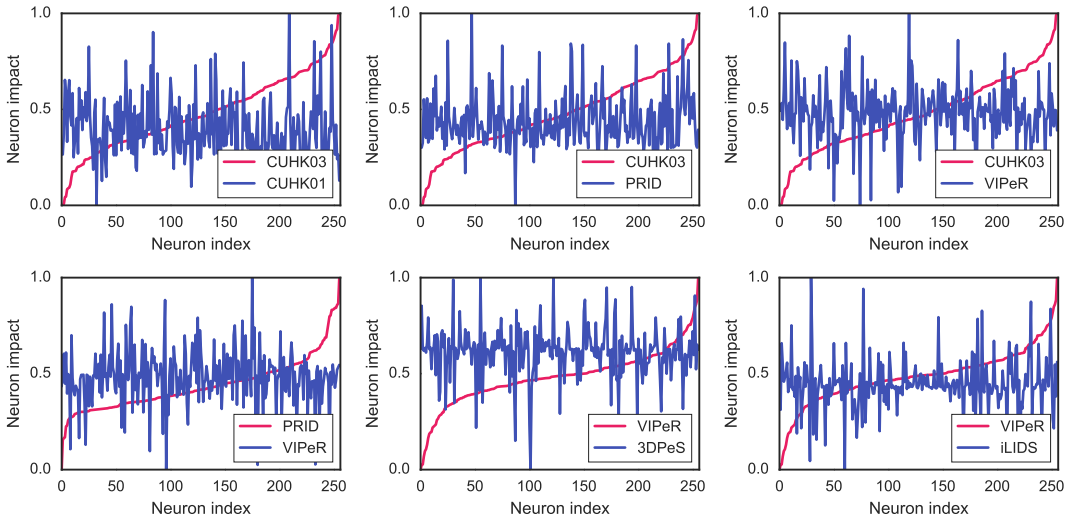


Figure 5.3: *The neuron impact scores between several pairs of domains. For each pair of domains (A, B), the neurons are sorted w.r.t. their impact scores on domain A (red curves). Their impact scores on domain B are shown in blue. The two curves have little correlation, which indicates that different domains have different effective neurons.*

passes, which is quite expensive if d is large. Therefore, we follow [106] to accelerate the process by using approximate Taylor's expansion of $\mathcal{L}(g(x))$ to the second order

$$s_i \approx -\frac{\partial \mathcal{L}}{\partial g(x)_i} g(x)_i + \frac{1}{2} \frac{\partial^2 \mathcal{L}}{\partial g(x)_i^2} g(x)_i^2. \tag{5.4}$$

We study the quality of this approximation empirically, and observe that it is more accurate for higher-level layers close to the loss function. Here we show in Figure 5.4 the difference between the approximation and its true values for the neurons of the fc7 layer.

After obtaining all the $\bar{s}_i$, we continue to train the CNN model, but with these impact scores as guidance to dropout different neurons for different domains during the training process. For all the samples belonging to a particular domain, we generate a binary mask m for the neurons according to their impact scores s, and then elementwisely multiply m with the neuron responses. Two schemes are proposed on how to generate the mask m. The first one is deterministic, which discards all the neurons

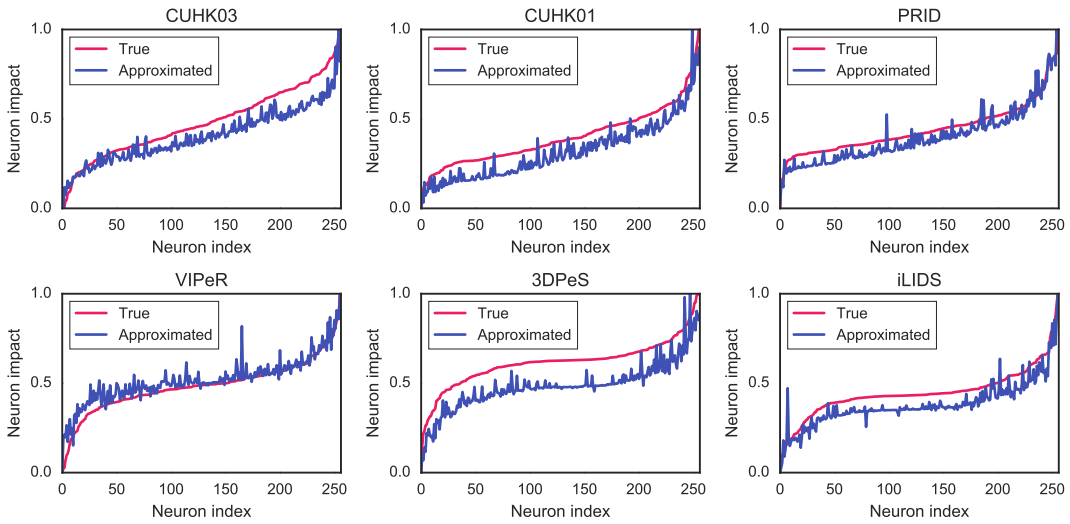


Figure 5.4: *Comparison of the true (Eq.* (5.3)*) and the approximated (Eq.* (5.4)*) neuron impact scores*

having non-positive impact scores

$$m_i = \begin{cases} 1 & \text{if } s_i > 0 \\ 0 & \text{if } s_i \leq 0 \end{cases} \tag{5.5}$$

The other one is stochastic, where m_i is drawn from a Bernoulli distribution with probability

$$p(m_i = 1) = \frac{1}{1 + e^{-s_i/T}}. \tag{5.6}$$

Here we use the sigmoid function to map a impact score to $(0, 1)$, and T is the temperature that controls how significantly the scores s would affect the probabilities. When $T \to 0$, it is equivalent to the deterministic scheme; when $T \to \infty$, it falls back to the standard Dropout with a ratio of 0.5. We study the effect of T empirically in Section 5.5.3.

We apply the Domain Guided Dropout to the fc7 neurons and resume the training process. The network's learning rate policy is changed to decay polynomially from 0.01 with the power parameter set to 0.5. The whole network is trained for 10 more epochs.

During the test stage, for the deterministic scheme, the neurons are also discarded if their impacts are no greater than zero. While for the stochastic scheme, we keep all the neuron responses but scale the i-th one with $1/(1 + e^{-s_i/T})$.

5.5 Experiments

We conducted experiments on several popular person re-identification datasets. In this section, we first detail the characteristics of each dataset and the test protocols we followed in Section 5.5.1. Then we compare the results of our approach with state-of-the-arts, showing the effectiveness of our multi-domain deep learning pipeline in Section 5.5.2. Section 5.5.3 analyzes the Domain Guided Dropout module through a series of experiments, and discusses its properties based on the results. At last, we present some figures that help us understand the underlying mechanisms. The code is publicly available on GitHub[1] .

5.5.1 Datasets and protocols

There exist many challenging person re-identification datasets. In our experiments, we chose seven of them to cover a wide range of domain varieties. CUHK03 [23] is one of the most largest published person re-identification datasets, it consists of five different pairs of camera views, and has more than 14,000 images of 1467 pedestrians. CUHK01 [86] is also captured on the same campus with CUHK03, but only has two camera views and 1552 images in total. PRID [87] extracts pedestrian images from recorded trajectory video frames. It has two camera views, each contains 385 and 749 identities, respectively. But only 200 of them appear in both views. Shinpuhkan [107] is another large-scale dataset with more than 22,000 images. The highlight of this dataset is that it contains only 24 individuals, but all of them are captured with 16 cameras, which provides rich information on intra-personal variations.

The remaining three datasets are relatively quite small. VIPeR [88] is one of the most challenging dataset, since it has 632 people but with various poses, viewpoints, image resolutions, and lighting conditions. 3DPeS [108] has 193 identities but the number of images for each person is not fixed. iLIDS [109] captures 119 individuals by surveillance cameras in an airport, and thus consists of large occlusions due to luggages and other passengers.

Since Shinpuhkan dataset has only 24 people, it cannot be used for testing the performance of re-identification systems. Thus we only use it in the training phase. For the other datasets, we mainly follow the settings in [102] to generate the test probe

[1] https://github.com/Cysu/person_reid

Dataset	#ID	#Training images	#Validation images	#Probe ID	#Gallery ID
CUHK03 [23]	1467	21012	5252	100	100
CUHK01 [86]	971	1552	388	485	485
PRID [87]	385	2997	749	100	649
VIPeR [88]	632	506	126	316	316
3DPeS [108]	193	420	104	96	96
i-LIDS [109]	119	194	48	60	60
Shinpuhkan [107]	24	18004	4500		

Table 5.2: *Statistics of the datasets and evaluation protocols*

and gallery sets. But our training set has two differences with theirs. First, both the manually cropped and automatically detected images in CUHK03 were used. Second, we sampled 10 images from the video frames of the training identities in PRID. We also randomly drew roughly 20% of all these images for validation. Notice that both the training and validation identities have no overlap with the test ones. The statistics of all the datasets and evaluation protocols are summarized in Table 5.2. In our experiments, we employed the commonly used CMC [110] top-1 accuracy to evaluate all the methods.

5.5.2 Comparison with state-of-the-art methods

We compare the results of our approach with those by state-of-the-art ones on all the six test datasets. For the 3DPeS and iLIDS datasets, the best previous method are [103] and [101], respectively. While for the other four datasets, the best results are reported by [102]. Both methods are built upon hand-crafted features, and exploit a ranking ensemble of kernel-based metrics to boost the performance. However, our method relies on the learned CNN features and uses the Euclidean distance directly as the metric, which stresses the quality of the learned features representation rather than the metrics.

In order to validate our approach, we first obtain a baseline by training the CNN individually on each domain. Then we merge all the domains jointly with a single-task learning objective (JSTL) and train the CNN from scratch using all these domains. Next, we improve the learned CNN with the proposed deterministic Domain Guided Dropout (JSTL+DGD). Notice that this step provides a single model working on all the domains simultaneously. To show our best possible results, we further fine-tune

Method	CUHK03	CUHK01	PRID	VIPeR	3DPeS	iLIDS
Best	62.1 [102]	53.4 [102]	17.9 [102]	**45.9** [102]	54.2 [103]	52.1 [101]
Individually	72.6	34.4	37.0	12.3	31.1	27.5
JSTL	72.0	62.1	59.0	35.4	44.5	56.9
JSTL+DGD	72.5	63.0	60.0	37.7	45.6	59.6
FT-JSTL	74.8	66.2	57.0	37.7	54.0	61.1
FT-JSTL+DGD	**75.3**	**66.6**	**64.0**	38.6	**56.0**	**64.6**

Table 5.3: *CMC top-1 accuracies of different methods*

the CNN separately on each domain with the stochastic Domain Guided Dropout (FT-JSTL+DGD). We also adopt a baseline method by fine-tuning from the JSTL model on each domain with standard dropout (FT-JSTL) for comparison. The results are summarized in Table 5.3.

CNN structure. We first evaluate the effectiveness of the proposed CNN structure. When the network is trained only with the CUHK03 dataset, which is large enough for training CNN from scratch, we improve the state-of-the-art result by more than 10% to 72.6% (row 2 of Table 5.3). Compared with the previous best deep learning method [38], whose result is 54.7%, our method achieves a gain of 18% in the performance. A two-stream network is used in [38] to compute the verification loss given a pair of images, while we opt for learning a single CNN through an ID classification task and directly computing Euclidean distance based on the features. When the training set is large enough, this classification objective makes the CNN much easier to train. The CMC curves of different methods on the CUHK03 dataset are shown in Figure 5.5. However, when the dataset is quite small, it would be insufficient to learn such a large capacity network from scratch, which is demonstrated in Table 5.3 by the results of training the CNN only on each of the VIPeR, 3DPeS, and iLIDS datasets.

Joint learning. To overcome the scale issue of small datasets, we propose to merge all the datasets jointly as a single-task learning (JSTL) problem. In row three of Table 5.3, we can see the performance increase on most of the datasets. This indicates learning from multiple domains jointly is very effective to produce generic feature representations for all the domains. An interesting phenomenon is that the performance on CUHK03 decreases slightly. We hypothesize that when combining different datasets together without special treatment, the larger domains would leverage their information

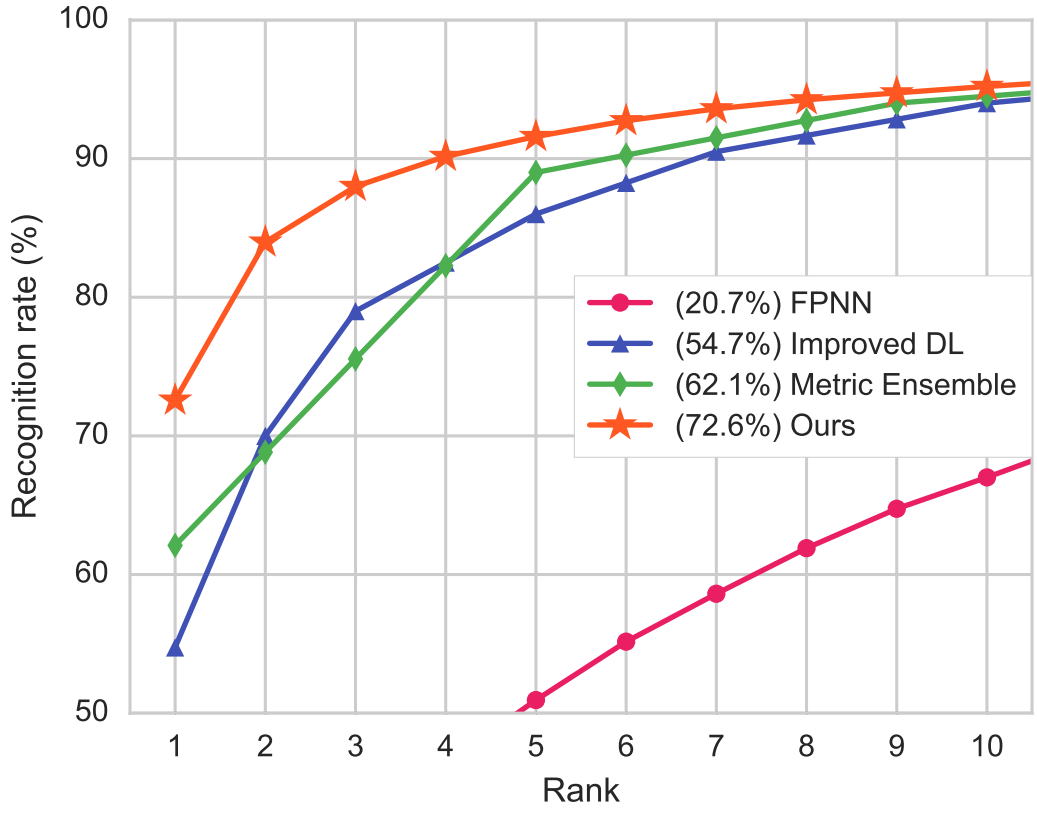


Figure 5.5: *CMC curves of different methods on CUHK03 dataset*

to help the learning on the others, which makes the features more robust on different datasets but less discriminative on the larger ones themselves. Note that we do not balance the data from multiple sources in a mini-batch, as it would give more weights on smaller datasets, which leads to severe overfitting.

Domain Guided Dropout. The fourth row of Table 5.3 shows the effectiveness of applying the proposed Domain Guided Dropout (DGD) to the JSTL scheme. Based on the JSTL pretrained model, we compute the neuron impact scores of the fc7 layer on different domains, replace the standard Dropout layer with the proposed deterministic Domain Guided Dropout layer, and continue to train the network for several epochs. Although the original JSTL model has already converged to a local minimum, utilizing Domain Guided Dropout consistently improves the performance on all the domains by 0.5%-2.7%. This indicates that it is effective to regularize the network specifically for different domains, which maximizes the discriminative power of the CNN on all the

domains simultaneously.

At last, to achieve the best possible performance of our model on each domain, we fine-tune the previous JSTL+DGD model on each of them individually with stochastic Domain Guided Dropout. This step adapts the CNN to the specific domain biases and sacrifices the generalization ability to other domains. As a result, the final CMC top-1 accuracies are increased by several percents, as listed in the last row of Table 5.3. On the other hand, comparing with FT+JSTL, the results are improved by 3% on average, which indicates that JSTL+DGD provides better generic features. Note that FT+JSTL on PRID results in even worse performance than JSTL. Such overfitting problem is resolved by applying DGD.

5.5.3 Effectiveness of Domain Guided Dropout

After evaluating the overall performance of our pipeline, we also investigate in details the effects of the proposed Domain Guided Dropout module in this subsection.

Temperature T. As the temperature T significantly affects the behavior and performance of the stochastic Domain Guided Dropout scheme, we first study the effects of this hyperparameter. From the theoretical analysis we know that the stochastic Domain Guided Dropout falls back to the standard Dropout (ratio equals to 0.5) when $T \to \infty$, and to the deterministic scheme when $T \to 0$. However, it is still unclear how to set it properly in real applications. Therefore, we provide some empirical results of tuning the temperature T. We use the 3DPeS dataset as an example, and fine-tune the JSTL+DGD model on it with different values of T. For each temperature, all the fc7 neurons have certain probabilities to be reserved according to Eq (5.6). We count the histogram of the neurons with respect to their probabilities to be reserved, and plot the cumulative distribution function in Figure 5.6. We can see that the best performance can be achieved when T is in a certain range that makes $\max_i p(m_i = 1) \approx 0.9$. This phenomenon indicates that a good T should assign the most effective neuron a high enough probability (0.9) to be reserved. We set T according to this empirical observation when using the stochastic Domain Guided Dropout scheme in our experiments.

Deterministic vs. stochastic. The next question is whether the deterministic and stochastic Domain Guided Dropout have similar behaviors, or one outperforms

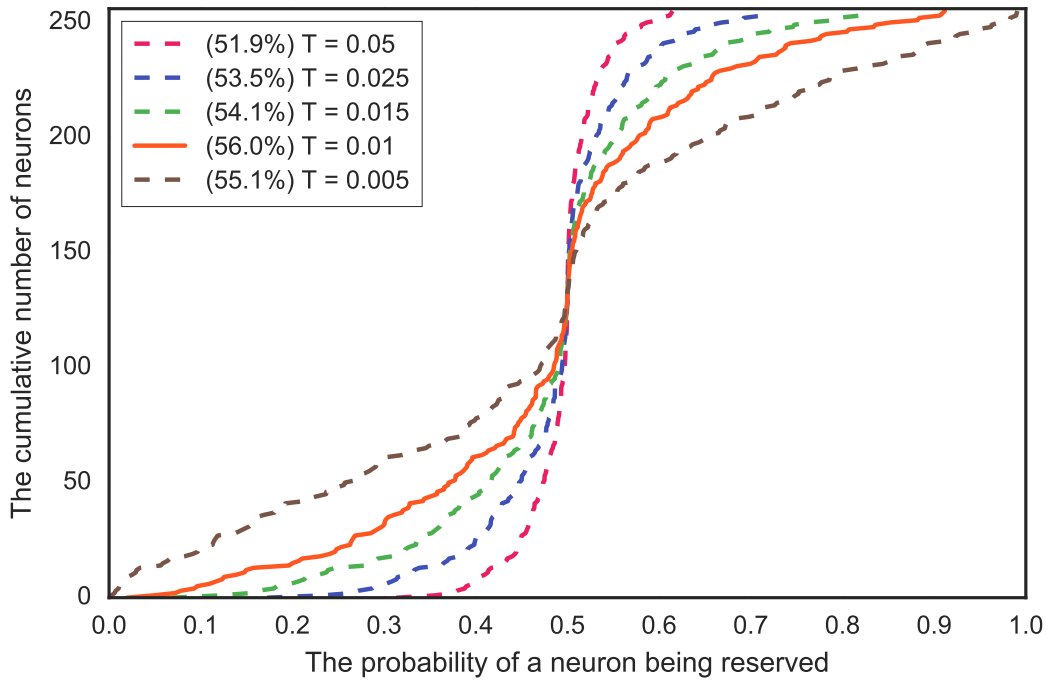


Figure 5.6: *The cumulative number of neurons to be reserved under certain probabilities. Different temperature T settings and corresponding CMC top-1 accuracies are shown in the legend.*

the other in certain pipeline stages. We compare these two strategies within the JSTL+DGD and FT-JSTL+DGD stages in our pipeline. Their gains on the CMC performance for each domain under different settings are shown in Figure 5.7 as the blue and green bars, respectively.

From Figure 5.7(a) we can see that when feeding the network with the data from all the domains, deterministic Domain Guided Dropout is better in general. This is because the objective here is to learn generic representations that are robust for different domains. The deterministic scheme strictly constrains that data from each domain are used to update only a specific subset of neurons. Thus it eliminates the potential confusion due to the discrepancies between different domains. On the contrary, when fine-tuning the CNN with the data only from one specific domain, the domain discrepancy no longer exists. All the inputs follow the same underlying distribution, so we can use stochastic Domain Guided Dropout to update all the neurons with proper guidance to determine the dropout rate for each of them, as shown in Figure 5.7(b). As a conclusion, the deterministic DGD is more effective when it is used to train the CNN

Figure 5.7: *Comparison of different Dropout schemes*

jointly with all the domains, while the stochastic DGD is superior when fine-tuning the net separately on each domain.

Standard Dropout vs. Domain Guided Dropout. At last, we compare the proposed Domain Guided Dropout with the standard Dropout under different scenarios. The results are summarized in Figure 5.7. First, when resuming the training of the JSTL pretrained model, we applied the deterministic Domain Guided Dropout. From Figure 5.7(a) we can see that since the model is already converged, continue to use standard Dropout scheme cannot further improve the performance. The performance would rather jitter insignificantly or decrease on particular domains due to overfitting. However, by using the deterministic Domain Guided Dropout scheme, the performance improves consistently on all the domains, especially for the small-scale ones. On the other hand, by comparing the orange and the green bars in Figure 5.7(b), we can

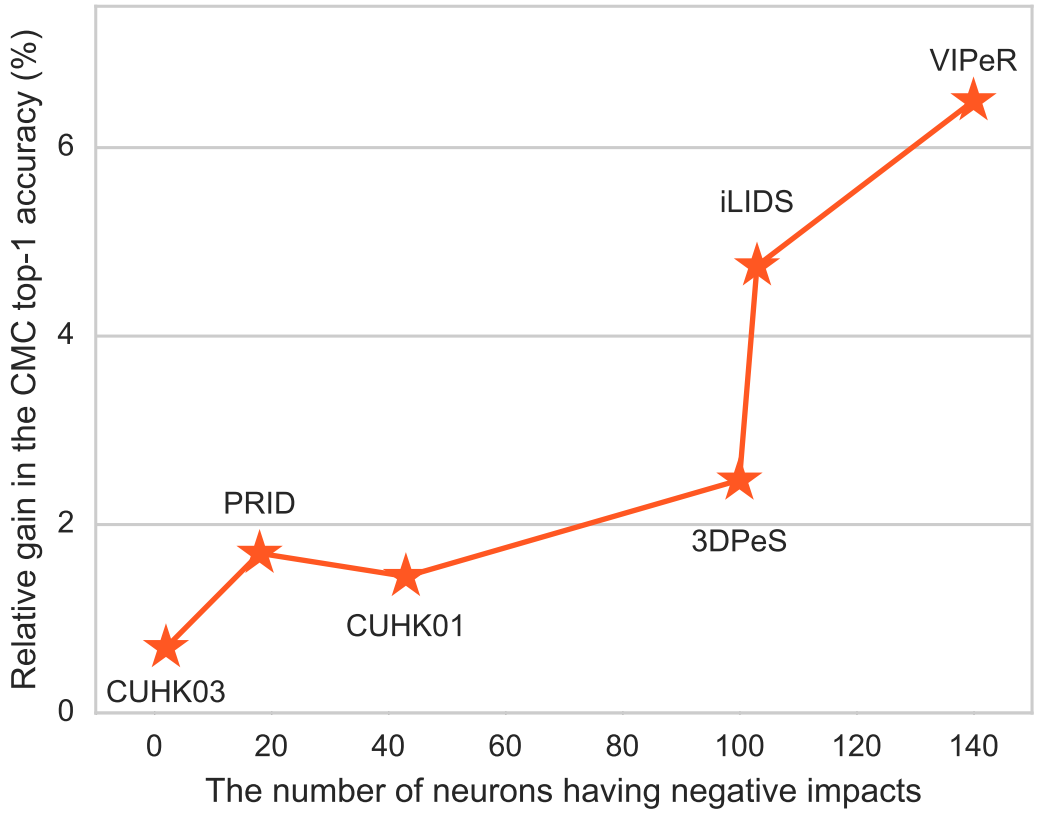


Figure 5.8: *Relative performance gain with respect to the number of neurons having negative impact scores on specific domain in the deterministic Guided Dropout scheme*

validate the effectiveness of the stochastic Domain Guided Dropout when fine-tuning the CNN model. This is because we utilize the domain information to regularize the network better, which keeps the CNN in the right track when training data is not enough.

We further investigate how does the deterministic Domain Guided Dropout change the network behavior by evaluating the relative performance gain on each domain with respect to the number of neurons having negative impact scores on that domain. As shown in Figure 5.8, smaller datasets tend to have more useless neurons to be dropped out, meanwhile the performance would be increased more significantly. This again indicates that we should not treat all the domains equally when using all their data, but rather regularize the CNN properly for each of them.

5.6 Conclusions

In this chapter, we raise the question of learning generic and robust CNN feature representations from multiple domains. An effective pipeline is presented, and a Domain Guided Dropout algorithm is proposed to improve the feature learning process. We conduct extensive experiments on multiple person re-identification datasets to validate our method and investigate the internal mechanisms in details. Moreover, our results outperform state-of-the-art ones by large margin on most of the datasets, which demonstrates the effectiveness of the proposed method.

Chapter 6

From Person Re-Identification to Person Search

Existing person re-identification benchmarks and methods mainly focus on matching cropped pedestrian images between queries and candidates. However, it is different from real-world scenarios where the annotations of pedestrian bounding boxes are unavailable and the target person needs to be searched from a gallery of whole scene images. To close the gap, we propose a new deep learning framework for person search. Instead of breaking it down into two separate tasks — pedestrian detection and person re-identification, we jointly handle both aspects in a single convolutional neural network. An Online Instance Matching (OIM) loss function is proposed to train the network effectively, which is scalable to datasets with numerous identities. To validate our approach, we collect and annotate a large-scale benchmark dataset for person search. It contains $18,184$ images, $8,432$ identities, and $96,143$ pedestrian bounding boxes. Experiments show that our framework outperforms other separate approaches, and the proposed OIM loss function converges much faster and better than the conventional Softmax loss.

The contribution of this chapter is three-fold. First, we propose a new deep learning framework to search a target person from a gallery of whole scene images. Instead of simply combining the pedestrian detectors and person re-id methods, we jointly optimize both objectives in a single CNN and they better adapt with each other. Second, we propose an Online Instance Matching loss function to learn identification features more effectively, which enables our framework to be scalable to large datasets with numerous identities. Together with the fast inference speed, our framework is much closer to the real-world application requirements. At last, we collect and annotate a large-scale benchmark dataset for person search, covering hundreds of scenes from street and movie snapshots. The dataset contains $18,184$ images, $8,432$ identities, and

96, 143 pedestrian bounding boxes. We validate the effectiveness of our approach comparing against other baselines on this dataset. The dataset and code are made public to facilitate further research[1] .

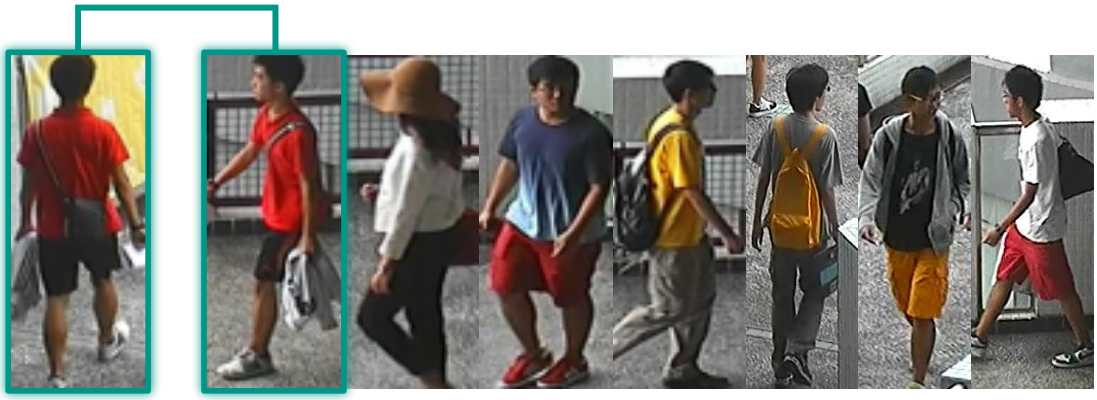
6.1 Related Work

Person re-identification. Person re-identification (re-id) [22, 111] aims at matching a target person with a gallery of pedestrian images. It has many video surveillance applications, such as finding criminals [112], cross-camera person tracking [113], and person activity analysis [114]. The problem is challenging because of complex variations of human poses, camera viewpoints, lighting, occlusion, resolution, background clutter, *etc.*, and thus draws much research attention in recent years [6, 23, 24, 91, 102, 115].

Early person re-identification methods addressed the problem by manually designing discriminative features [41, 116, 117], learning feature transforms across camera views [118–120], and learning distance metrics [102, 118, 121–123]. Recent years, many researchers have proposed various deep learning based methods that jointly handle all these aspects. Li *et al.* [23] and Ahmed *et al.* [38] designed specific CNN models for person re-id. Both the networks utilize as input a pair of cropped pedestrian images and employ a binary verification loss function to train the parameters. Ding *et al.* [101] and Cheng *et al.* [124] exploited triplet samples for training CNNs to minimize the feature distance between the same person and maximize the distance between different people. Apart from using pairwise or triplet loss functions, Xiao *et al.* [115] proposed to learn features by classifying identities. Multiple datasets are combined together and a domain guided dropout technique is proposed to improve the feature learning. Several recent works addressed on solving person re-id on abnormal images, such as low-resolution images [125], or partially occluded images [126].

Although numerous person re-id datasets and methods have been proposed, there is still a big gap between the problem setting itself and real-world applications. In most benchmarks [23, 86, 88, 109, 127], the gallery only contains manually cropped pedestrian images (Figure 6.1a), while in real applications, the goal is to find a target person in a gallery of whole scene images, as shown in Figure 6.1b. Following the protocols of these

[1] https://github.com/ShuangLI59/person_search



(a) Person re-id: matching with manually cropped pedestrians

(b) Person search: finding from whole scene images

Figure 6.1: *Comparison between person re-identification and person search. The person search problem setting is closer to real-world applications and more challenging, as detecting pedestrians would inevitably produce false alarms, misdetections, and misalignments.*

benchmarks, most of the existing person re-id methods assume perfect pedestrian detections. However, these manually cropped bounding boxes are unavailable in practical applications. Off-the-shelf pedestrian detectors would inevitably produce false alarms, misdetections, and misalignments, which could harm the final searching performance significantly.

In 2014, Xu *et al.* [25] made the first step towards closing this gap. They introduced the person search problem to the community, and proposed a sliding window searching strategy based on a combination of pedestrian detection and person matching scores. However, the performance is limited by the handcrafted features, and the sliding window framework is not scalable.

Concurrent with our prior arXiv submission, Zheng *et al.* [27] also contributed a

benchmark dataset for person search. They exploited separate detection and re-id methods with scores re-weighting to solve the problem, while in this work we propose a deep learning framework that jointly handles both aspects.

Pedestrian detection. DPM [128], ACF [129], and Checkerboards [130] are the most commonly used off-the-shelf pedestrian detectors. They rely on hand-crafted features and linear classifiers to detect pedestrians. Recent years, CNN-based pedestrian detectors have also been developed [131, 132]. Various factors, including CNN model structures, training data, and different training strategies are studied empirically in [133]. Tian *et al.* [134] exploited pedestrian and scene attribute labels to train CNN pedestrian detectors in a multi-task manner. Cai *et al.* [135] proposed a complexity-aware boosting algorithm for learning CNN detector cascades.

6.2 Method Overview

In this chapter, we propose a new deep learning framework for person search. Different from conventional approaches that break down the problem into two separate tasks — pedestrian detection and person re-identification, we jointly handle both aspects in a single CNN. Our CNN consists of two parts, given a whole input gallery image, a pedestrian proposal net is used to produce bounding boxes of candidate people, which are fed into an identification net to extract features for comparing with the target person. The pedestrian proposal net and the identification net adapt with each other during the joint optimization. For example, the proposal net can focus more on the recall rather than the precision, as false alarms could be eliminated through the latter features matching process. Meanwhile, misalignments of proposals are also acceptable, as they can be further adjusted by the identification net. To improve the scalability of the whole system, inspired by recent advances in object detection [136], we encourage both parts to share underlying convolutional feature maps, which significantly accelerates the inference procedure.

Traditional re-id feature learning mainly employs pairwise or triplet distance loss functions [23, 38, 101, 124]. However, they are not efficient as only several data samples are compared at each time, and there are $O(N^2)$ potential input combinations, where N is the number of images. Different sampling strategies could significantly impact the convergence rate and quality, but finding efficient sampling strategies becomes much

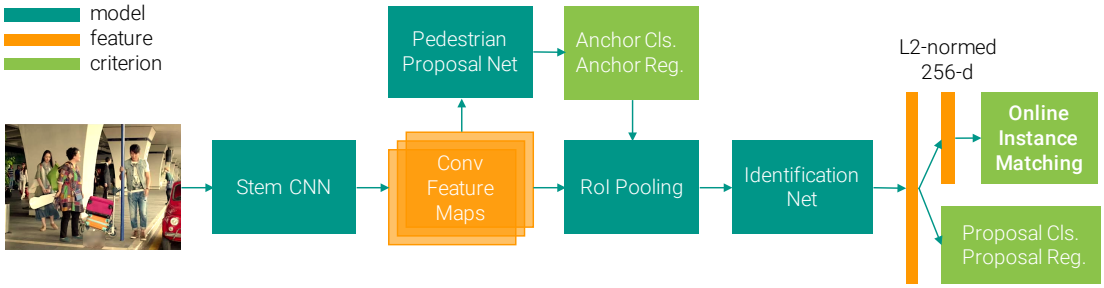


Figure 6.2: *Our proposed framework. Pedestrian proposal net generates bounding boxes of candidate people, which are fed into an identification net for feature extraction. We project the features to a L2-normalized 256-d subspace, and train it with a proposed Online Instance Matching loss. Both the pedestrian proposal net and the identification net share the underlying convolutional feature maps.*

more difficult as N increases. Another approach is learning to classify identities with the Softmax loss function [115], which effectively compares all the samples at the same time. But as the number of classes increases, training the big Softmax classifier matrix becomes much slower or even cannot converge. In this chapter, we propose a novel *Online Instance Matching (OIM)* loss function to cope with the problems. We maintain a lookup table of features from all the labeled identities, and compare distances between mini-batch samples and all the registered entries. On the other hand, many unlabeled identities could appear in scene images, which can be served as negatives for labeled identities. We thus exploit a circular queue to store their features also for comparison. This is another advantage brought by the person search problem setting. The proposed parameter-free OIM loss converges much faster and better than the Softmax loss in our experiments.

6.3 Network Architecture

We propose a new deep learning framework that jointly handles the pedestrian detection and person re-identification in a single convolutional neural network (CNN), as shown in Figure 6.2. Given as input a whole scene image, we first use a stem CNN to transform from raw pixels to convolutional feature maps. A pedestrian proposal net is built upon these feature maps to predict bounding boxes of candidate people, which are then fed into an identification net with RoI-Pooling [14] to extract L2-normalized 256-d features for each of them. At inference stage, we rank the gallery people according to their feature distances to the target person. At training stage, we propose an *Online*

Instance Matching (OIM) loss function on top of the feature vectors to supervise the identification net, together with several other loss functions for training the proposal net in a multi-task manner. Below we will first detail the CNN model structure, and then elaborate on the OIM loss function.

We adopt the ResNet-50 [10] as our base CNN model. It has a 7×7 convolution layer in front (named conv1), followed by four blocks (named conv2_x to conv5_x) each containing $3, 4, 6, 3$ residual units, respectively. We exploit conv1 to conv4_3 as the stem part. Given an input image, the stem will produce 1024 channels of features maps, which have 1/16 resolutions of the original image.

On top of these feature maps, we build a pedestrian proposal network to detect person candidates. A $512 \times 3 \times 3$ convolutional layer is first added to transform the features specifically for pedestrians. Then we follow [136] to associate 9 anchors at each feature map location, and use a Softmax classifier to predict whether each anchor is a pedestrian or not, as well as a linear regression to adjust their locations. We will keep the top 128 adjusted bounding boxes after non-maximum suppression as our final proposals.

To find the target person among all these proposals, we build an identification net to extract the features of each proposal, and compare against the target ones. We first exploit an RoI-Pooling layer [14] to pool a $1024 \times 14 \times 14$ region from the stem feature maps for each proposal. Then they are passed through the rest conv4_4 to conv5_3 of the ResNet-50, followed by a global average pooling layer to summarize into a 2048 dimensional feature vector. On one hand, as the pedestrian proposals would inevitably contain some false alarms and misalignments, we use again a Softmax classifier and a linear regression to reject non-persons and refine the locations. On the other hand, we project the features into a L2-normalized 256 dimensional subspace (id-feat), and use them to compute cosine similarities with the target person when doing inference. During the training stage, we supervise the id-feat with the proposed OIM loss function. Together with other loss functions for detection, the whole net is jointly trained in a multi-task learning manner, rather than using the alternative optimizations in [136].

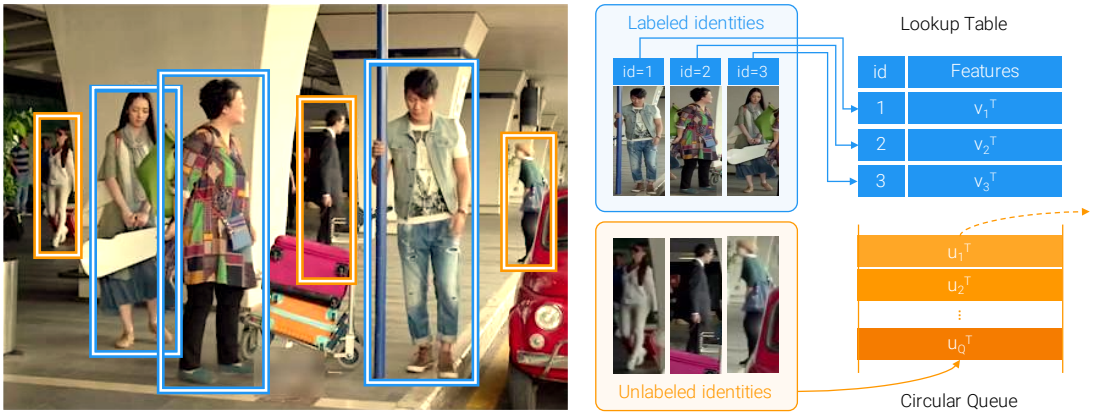


Figure 6.3: *Online Instance Matching. The left part shows the labeled (blue) and unlabeled (orange) identity proposals in an image. We maintain a lookup table (LUT) and a circular queue (CQ) to store the features. When forward, each labeled identity is matched with all the stored features. When backward, we update LUT according to the id, pushing new features to CQ, and pop out-of-date ones. Note that both data structures are external buffer, rather than the parameters of the CNN.*

6.4 Online Instance Matching Loss

There are three different types of proposals, labeled identities, unlabeled identities, and background clutter. Suppose there are L different target people in the training set, when a proposal matches a target person, we call it an instance of the labeled identity, and assign a class-id (from 1 to L) to it accordingly. There are also lots of proposals predicting pedestrians correctly, but do not belong to anyone of our target people. We call them unlabeled identities in such cases. We demonstrate some examples of labeled and unlabeled identities in Figure 6.3 with blue and orange bounding boxes, respectively. Other proposals are just false alarms on other objects or background regions. In the proposed loss function, we only consider the labeled and unlabeled identities, while leave the other proposals untouched.

As our goal is to distinguish different people, a natural objective is to minimize the features discrepancy among the instances of the same person, while maximize the discrepancy among different people. To fulfill this goal, we need to memorize the features of all the people. This could be done offline by doing network forward on all the training images, but it is not practical when using stochastic gradient descent (SGD) for optimization. Thus in our approach, we choose an online approximation instead. Denote the features of a labeled identity inside a mini-batch by $x \in \mathbb{R}^D$, where

D is the feature dimension, we maintain a lookup table (LUT) $V \in \mathbb{R}^{D \times L}$ to store the features of all the labeled identities, as demonstrated in Figure 6.3. During the forward propagation, we compute cosine similarities between the mini-batch sample and all the labeled identities by $V^T x$. During backward, if the target class-id is t, then we will update the t-th column of the LUT by $v_t \leftarrow \gamma v_t + (1-\gamma)x$, where $\gamma \in [0, 1]$, and then scale v_t to have unit L2-norm.

Apart from labeled identities, many unlabeled identities are also valuable for learning feature representations. They can be safely used as negative classes for all the labeled identities. We use a circular queue to store the features of these unlabeled identities that appear in recent mini-batches. Denote the features in this circular queue by $U \in \mathbb{R}^{D \times Q}$, where Q is the queue size, we can also compute their cosine similarities with the mini-batch sample by $U^T x$. After each iteration, we push the new feature vectors into the queue, while pop the out-of-date ones to keep the queue size unchanged.

Based on these two data structures, we define the probability of x being recognized as the identity with class-id i by a Softmax function

$$p_i = \frac{\exp(v_i^T x/\tau)}{\sum_{j=1}^{L} \exp(v_j^T x/\tau) + \sum_{k=1}^{Q} \exp(u_k^T x/\tau)}, \tag{6.1}$$

where higher temperature τ leads to softer probability distribution. Similarly, the probability of being recognized as the i-th unlabeled identity in the circular queue is

$$q_i = \frac{\exp(u_i^T x/\tau)}{\sum_{j=1}^{L} \exp(v_j^T x/\tau) + \sum_{k=1}^{Q} \exp(u_k^T x/\tau)}. \tag{6.2}$$

OIM objective is to maximize the expected log-likelihood

$$\mathcal{L} = \mathrm{E}_x \left[\log p_t\right], \tag{6.3}$$

and its gradient with respect to x can be derived as

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{1}{\tau}\left[(1-p_t)v_t - \sum_{\substack{j=1 \\ j \neq t}}^{L} p_j v_j - \sum_{k=1}^{Q} q_k u_k\right]. \tag{6.4}$$

It can be seen that our OIM loss effectively compares the mini-batch sample with all the labeled and unlabeled identities, driving the underlying feature vector to be

similar with the target one, while pushing it away from the others.

Why not Softmax loss? A natural question here is that why not learning a classifier matrix with a conventional Softmax loss to predict the class-id. There are mainly two drawbacks. First, large-scale person search datasets would have a large number of identities (more than $5,000$ in our training set), while each identity only has several instances and each image only contains a few identities. We need to learn more than $5,000$ discriminant functions simultaneously, but during each SGD iteration we only have positive samples from tens of classes. The classifier matrix suffers from large variance of gradients and thus cannot be learned effectively, even with proper pre-training and high momentum. Second, we cannot exploit the unlabeled identities with Softmax loss, as they have no specific class-ids.

Although our OIM loss formulation is similar to the Softmax one, the major difference is that the OIM loss is non-parametric. The LUT and circular queue are considered as external buffer, rather than the network parameters. The gradients directly operate on the features without the transformation by a classifier matrix. The potential drawback of this non-parametric loss is that it could overfit more easily. We find that projecting the features into a L2-normalized low-dimensional subspace helps reduce overfitting.

Scalability. Computing the partition function in Eq (6.1) and Eq (6.2) could be time consuming when the number of identities increases. To overcome this problem, we can approximate the denominators by sub-sampling the labeled and unlabeled identities,

$$p_i \approx \frac{\exp(v_i^T x/\tau)}{\sum_{j\in\mathcal{I}_L}\exp(v_j^T x/\tau) + \sum_{k\in\mathcal{I}_Q}\exp(u_k^T x/\tau)}, \tag{6.5}$$

where the index sets $\mathcal{I}_L$ and $\mathcal{I}_Q$ are subsets of $\{1, 2, \ldots, L\}$ and $\{1, 2, \ldots, Q\}$, respectively. Similar sub-sampling can be applied to q_i as well. Using these approximations indeed results in optimizing an upper-bound of the log-likelihood in Eq (6.3).

Relationship with Triplet loss. Triplet loss [105] is another commonly used loss function for learning feature embeddings. It samples a triplet of feature vectors $\{x, x^+, x^-\}$ at each time, where x is the anchor person, x^+ is a positive instance with the same identity, and x^- is a negative instance with different identities. The loss

formulation is defined as

$$\mathcal{L}_{triplet}\left(x, x^+, x^-\right) = \max\left(0, ||x - x^+||_2^2 - ||x - x^-||_2^2 + m\right), \tag{6.6}$$

where m is a constant called *margin*. The triplet loss aims at pushing the anchor towards the positive instance, while drawing it away from the negative instance, until their distances exceed a certain margin. If we derive the gradient, minimizing the triplet loss (suppose it is greater than zero) is essentially updating x with

$$x \leftarrow x + 2(x^+ - x^-), \tag{6.7}$$

which is very similar to our loss function when only x^+ and x^- are sub-sampled in Eq (6.5). In this case, the probability of x belonging to the class of x^+ becomes

$$p^+ = \frac{\exp(x^T x^+/\tau)}{\exp(x^T x^+/\tau) + \exp(x^T x^-/\tau)}. \tag{6.8}$$

Maximizing the log-likelihood $\log p^+$ updates x with

$$x \leftarrow x + \frac{1 - p^+}{\tau}(x^+ - x^-). \tag{6.9}$$

By comparing Eq (6.7) and Eq (6.9), we can see that the only difference between triplet loss and sub-sampled OIM loss is that triplet loss has a constant weight for all the pairs of positive and negative samples. While the sub-sampled OIM loss weights them adaptively according to the current classification probability.

Triplet loss compares the anchor with only two identity samples at each time, while our OIM loss (without sub-sampling) compares it against all the identity samples, which is arguably more effective for feature learning. Triplet loss is also very sensitive to the sampling strategy. In some preliminary experiments, we observe that OIM loss converges much faster and better than triplet loss on the training set, but could easily overfit on some small datasets. Their effects on large-scale datasets need to be further studied in the future.

Source / Split	# Images	# Pedestrians	# Identities
StreetSnap	12,490	75,845	6,057
Movie&TV	5,694	20,298	2,375
Training	11,206	55,272	5,532
Test	6,978	40,871	2,900
Overall	18,184	96,143	8,432

Table 6.1*: Statistics of the dataset with respect to data sources and training / test splits.*

6.5 Dataset

We collect and annotate a large-scale CUHK-SYSU Person Search Dataset to evaluate our proposed method. We exploit two data sources to diversify the scenes. On one hand, we use hand-held cameras to shoot street snaps around urban cities. On the other hand, we collect movie snapshots that contain pedestrians, as they could enrich the variations of viewpoints, lighting, and backgrounds. In this section, we will show the basic statistics of our dataset, as well as define the evaluation protocols and metrics.

6.5.1 Statistics

After collecting all the $18,184$ images, we first densely annotate all the $96,143$ pedestrians bounding boxes in these scenes, and then associate the person that appears across different images, resulting in $8,432$ labeled identities. The statistics of two data sources are listed in Table 6.1. We did not annotate those people who appear with half bodies or abnormal poses such as sitting or squatting. Moreover, people who change clothes and decorations in different video frames are not associated in our dataset, since person search problem requires to recognize identities mainly according to their clothes and body shapes rather than faces. We ensure that the background pedestrians do not contain labeled identities, and thus they can be safely served as negative samples for identification. Note that we also ignore the background pedestrians whose heights are smaller than 50 pixels, as they would be hard to recognize even for human labelers. The height distributions of labeled and unlabeled identities are demonstrated in Figure 6.4. It can be seen that our dataset has rich variations of pedestrian scales.

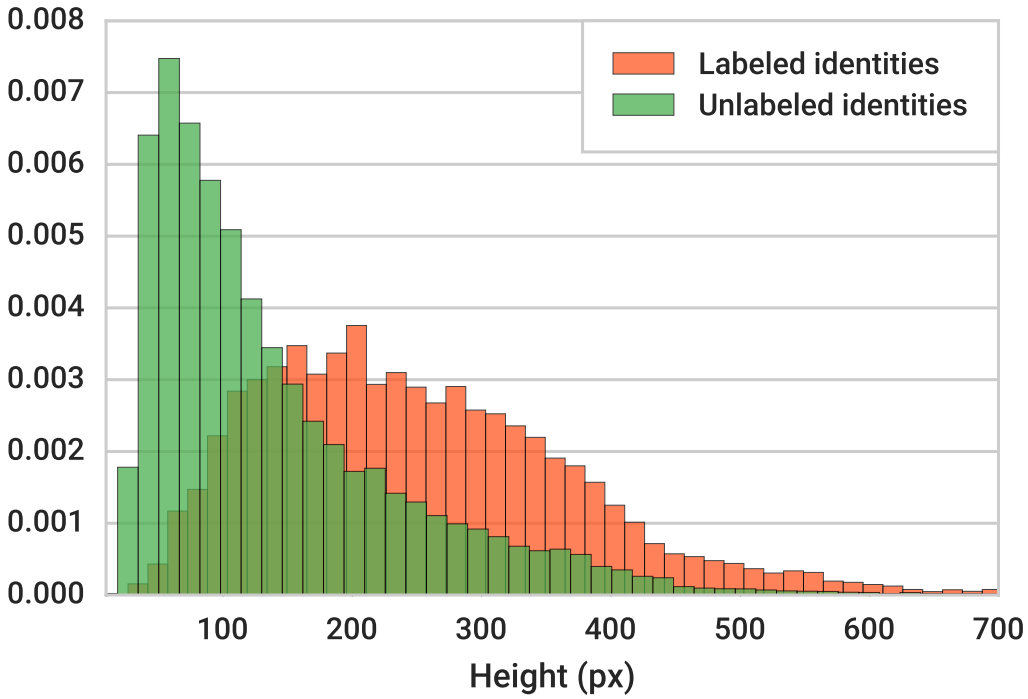


Figure 6.4: *The height distributions of labeled and unlabeled identities in our dataset.*

6.5.2 Evaluation Protocols and Metrics

We split the dataset into a training and a test subset, ensuring no overlapped images or labeled identities between them. Table 6.1 shows the statistics of these two subsets. We divide the test identity instances into queries and galleries. For each of the 2,900 test identities, we randomly choose one of his/her instances as the query, while the corresponding gallery set consists of two parts — all the images containing the other instances and some randomly sampled images not containing this person. Different queries have different galleries, and jointly they cover all the 6,978 test images.

To better understand how gallery size would affect the person search performance, we define a set of protocols with gallery size ranging from 50 to 4000. Taking gallery size of 100 as an example, as each image approximately contains 6 pedestrians, then our task is to find the target person among about 600 people. This setting is comparable with existing person re-id datasets (*e.g.*, CUHK-03, VIPeR) in terms of the number of gallery pedestrians, and is even more challenging as there could be thousands of background clutter bounding boxes distracting our attentions.

We employ two kinds of evaluation metrics — cumulative matching characteristics (CMC top-K) and mean averaged precision (mAP). The first one is inherited from the person re-id problem, where a matching is counted if there is at least one of the top-K predicted bounding boxes overlaps with the ground truths with intersection-over-union (IoU) greater or equal to 0.5. The second one is inspired from the object detection tasks. We follow the ILSVRC object detection criterion [137] to judge the correctness of predicted bounding boxes. An averaged precision (AP) is calculated for each query based on the precision-recall curve, and then we average the APs across all the queries to get the final result.

6.6 Experiments

To evaluate the effectiveness of our approach and study the impact of various factors on person search performance, we conduct several groups of experiments on the new dataset. In this section, we first detail the baseline methods and experiment settings in Section 6.6.1. Then we compare our joint framework with the baselines of using separate pedestrian detection and person re-identification in Section 6.6.2. Section 6.6.3 shows the effectiveness of our proposed Online Instance Matching (OIM) loss. At last, we present the influence of various factors, including detection recall and gallery size.

6.6.1 Experiment Settings

We implement our framework based on Caffe [83, 138] and py-faster-rcnn [14, 136]. ImageNet-pretrained ResNet-50 [10] are exploited for parameters initialization. We fix the first 7×7 convolution layer and the batch normalization (BN) layers as constant affine transformations in the stem part, while keep the other BN layers as normal in the identification part. The temperature scalar τ in Eq. (6.1) and Eq. (6.2) is set to 0.1, the size of the circular queue is set to $5,000$. All the losses have the same loss weight. Each mini-batch consists of two scene images. The learning rate is initialized to 0.001, dropped to 0.0001 after 40K iterations, and kept unchanged until the model converges at 50K iterations.

We compare our framework with conventional methods that break down the problem into two separate tasks — pedestrian detection and person re-identification. Three

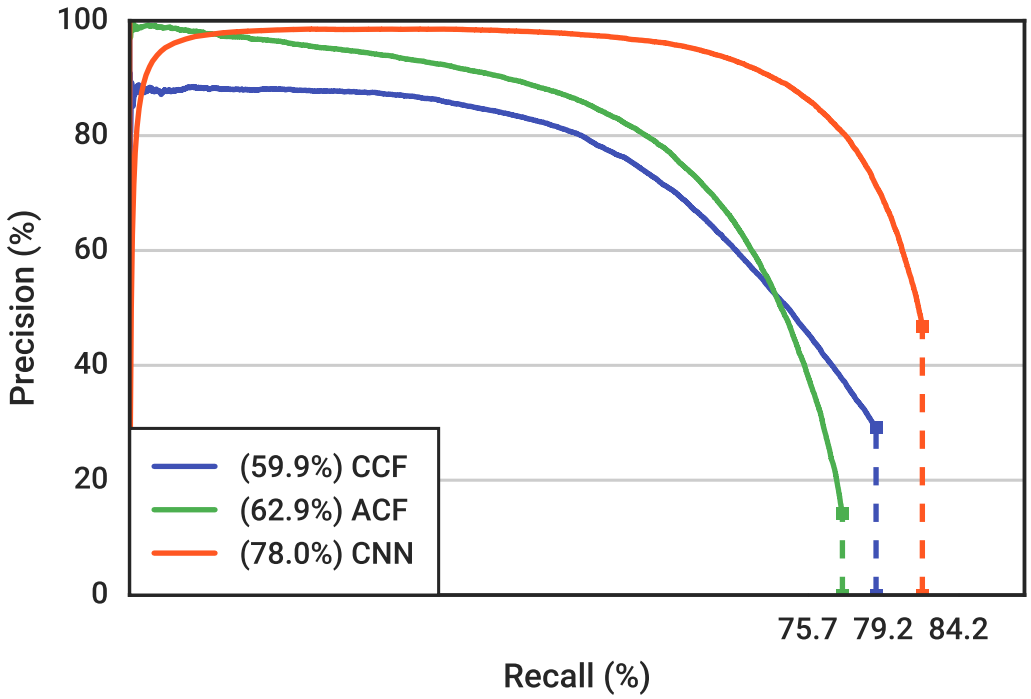


Figure 6.5: *Recall-Precision curves of different detectors. APs are listed in the legend.*

pedestrian detection and five person re-id methods are used in our experiments, resulting in 15 baseline combinations. For pedestrian detection, we directly use the off-the-shelf deep learning CCF [131] detector, as well as two other detectors specifically fine-tuned on our dataset. One is the ACF [129], and the other is Faster-RCNN (CNN) [136] with ResNet-50, which is equivalent to our framework but without the identification task. The recall-precision curve of each detector on our dataset are plotted in Figure 6.5. We also use the ground truth (GT) bounding boxes as the results of a perfect detector.

For person re-identification, we use several popular re-id feature representations, including DenseSIFT-ColorHist (DSIFT) [41], Bag of Words (BoW) [24], and Local Maximal Occurrence (LOMO) [6]. Each feature representation is used in conjunction with a specific distance metric, including Euclidean, Cosine similarity, KISSME [36], and XQDA [6], where KISSME and XQDA are trained on our dataset. Moreover, by discarding the pedestrian proposal network in our framework and training the remaining net to classify identities with Softmax loss from cropped pedestrian images, we get

CMC top-1 (%)	CCF	ACF	CNN	GT
DSIFT+Euclidean	11.7	25.9	39.4	45.9
DSIFT+KISSME	13.9	38.1	53.6	61.9
BoW+Cosine	29.3	48.4	62.3	67.2
LOMO+XQDA	46.4	63.1	74.1	76.7
IDNet	57.1	63.0	74.8	78.3
Ours (w/o unlabeled)	—	—	76.1	78.5
Ours	—	—	**78.7**	**80.5**
mAP (%)	CCF	ACF	CNN	GT
DSIFT+Euclidean	11.3	21.7	34.5	41.1
DSIFT+KISSME	13.4	32.3	47.8	56.2
BoW+Cosine	26.9	42.4	56.9	62.5
LOMO+XQDA	41.2	55.5	68.9	72.4
IDNet	50.9	56.5	68.6	73.1
Ours (w/o unlabeled)	—	—	72.7	75.5
Ours	—	—	**75.5**	**77.9**

Table 6.2: *Comparisons between our framework and separate pedestrian detection + person re-id methods.*

another baseline re-id method (IDNet). This training scheme has been exploited in [115] to learn discriminative re-id feature representations. In our experiments, when training IDNet with detector boxes, we found that adding background clutter as a unique class improves the result, while adding unlabeled identities does not.

The following results are reported using the protocol with gallery size equal to 100 if not specified.

6.6.2 Comparison with Detection and Re-ID

We first compare our proposed person search framework (with or without using unlabeled identities) with other 15 baseline combinations that break down the problem into separate detection and re-identification tasks. The results are summarized in Table 6.2. Our method outperforms the others by large margin. Comparing with CNN+IDNet, the gain comes from the joint optimization of the detection and identification parts, as well as the effective use of unlabeled identities in the OIM loss.

From Table 6.2 we can also see that different detectors affect the person search performance significantly for each re-id method. Directly using an off-the-shelf detector

may not be a good choice when applying existing re-id methods in the real-world person search applications. Otherwise the detector could become a bottleneck that diminishes the returns of better re-id methods.

On the other hand, the relative performance of different re-id methods are consistent across all the detectors. It implies that existing person re-id datasets could still guide us to design better feature representations, but it may lose some valuable data, such as unlabeled identities and background clutter, which come with the person search datasets.

Another interesting phenomenon is that although IDNet and LOMO+XQDA have similar performance when using GT or fine-tuned ACF and CNN detectors, IDNet is significantly better when using off-the-shelf CCF detector. We observe that the CCF detection results contain many misalignments. Hand-crafted features in such cases are not as robust as the IDNet counterpart.

6.6.3 Effectiveness of Online Instance Matching

We validate the effectiveness of the proposed Online Instance Matching (OIM) loss by comparing it against Softmax baselines with or without pretraining the classifier matrix. The training identification accuracy and test person search mAP curves are demonstrated in Figure 6.6. First, we can see that using Softmax loss without pretraining classifier remains at low accuracy during the whole process. This phenomenon verifies our analysis in Section 6.4 that learning a large classifier matrix is difficult. Even with proper pretraining, the training accuracy still improves slowly, and the test mAP keeps at around 60%.

On the contrary, the proposed OIM loss starts with a low training accuracy but converges much faster and also consistently improves the test performance. The parameter-free OIM loss learns features directly without needing to learn a big classifier matrix. Moreover, the mismatch between training and test criterion no longer exists, as both are computed based on the inner product of L2-normalized feature vectors, which represents the cosine similarity.

We further evaluate the impact of OIM loss on the standard person re-identification task. We train two different base CNNs, Inception [115] (from scratch) and ResNet-50 [10] (ImageNet pretrained), with either Softmax loss or OIM loss, on three large-scale

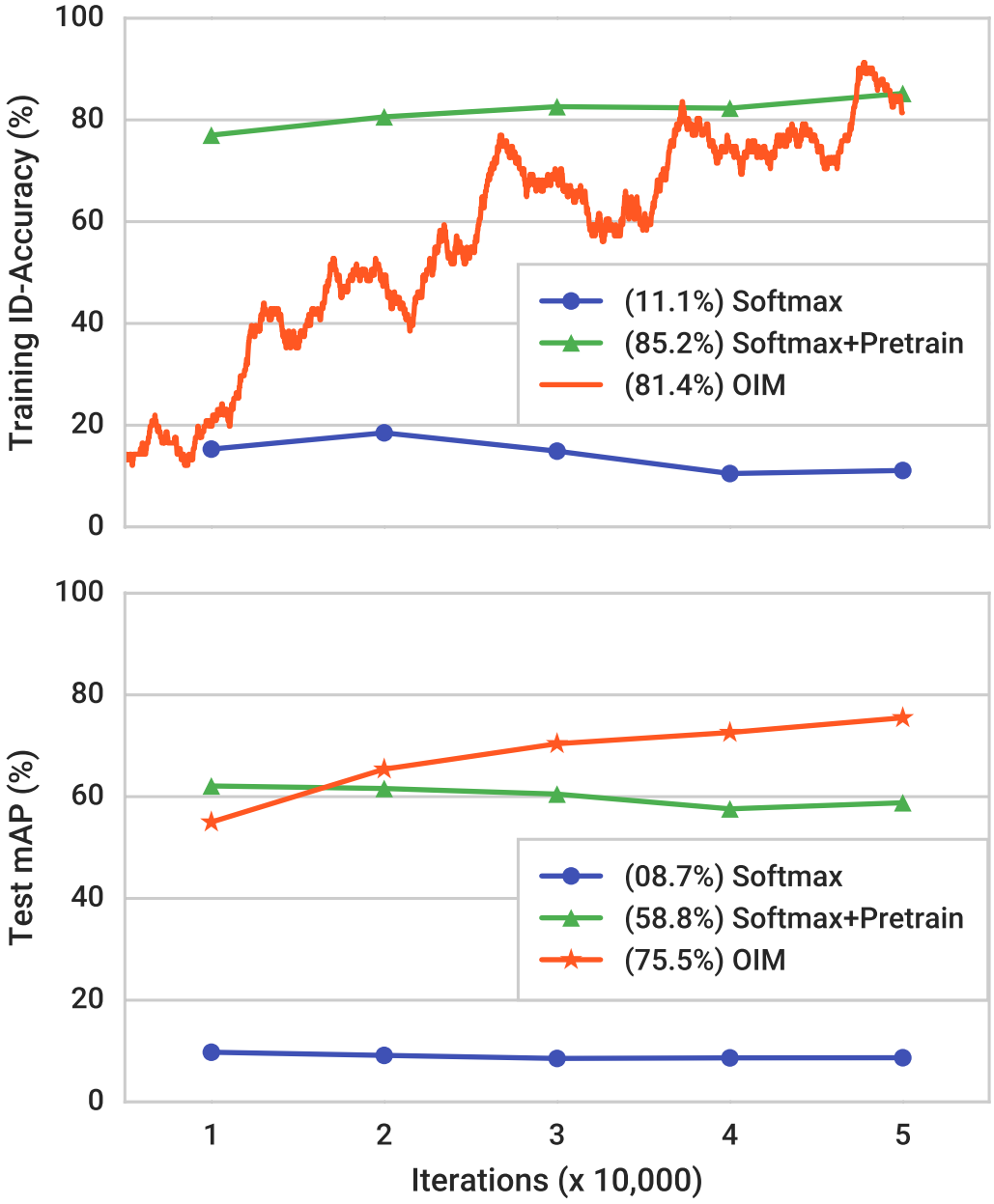


Figure 6.6: *Comparisons between using the proposed Online Instance Matching (OIM) and Softmax loss (with and without pretraining the Softmax classifier) in our framework. The final accuracies and mAPs are shown in the legends.*

person re-id datasets, CUHK03 [23], Market1501 [24], and Duke [58, 139]. Following their own protocols, we evaluate the CMC top-1 accuracy of using different loss functions, as listed in Table 6.3. OIM loss consistently outperforms Softmax loss, regardless

Network	Loss	CUHK03	Market1501	Duke
Inception	Softmax	73.2	75.8	54.4
Inception	OIM	**77.7**	**77.9**	**61.7**
ResNet-50	Softmax	70.8	81.4	62.5
ResNet-50	OIM	**77.5**	**82.1**	**68.1**

Table 6.3: *CMC top-1 accuracy (%) of using Softmax or OIM loss for standard person re-id task.*

Dimension	N/A	128	256	512	1024
top-1 (%)	59.3	65.9	**78.7**	78.2	78.5
mAP (%)	54.2	62.1	75.5	75.3	**75.7**

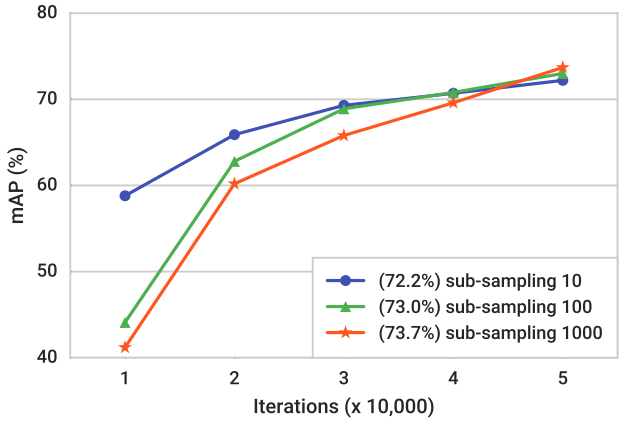
Table 6.4: *Comparisons among different dimensions of L2-normalized feature subspace. N/A means that we directly use the L2-normalized* 2048*-d global pooled feature vector.*

of which base CNN is used. We refer readers to Open-ReID[2] benchmarks for more details.

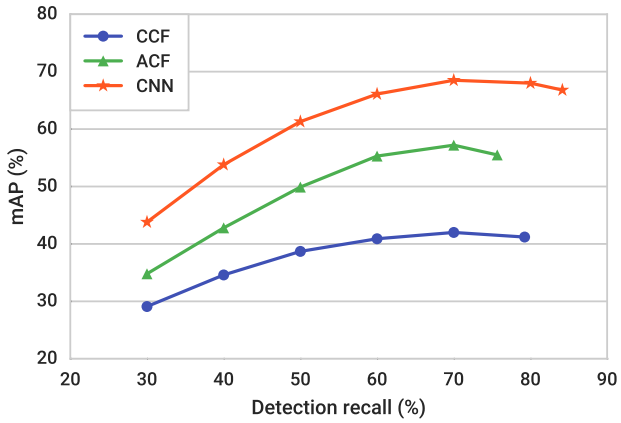
Sub-sampling the identities. As the number of identities increases, the computation time of the OIM loss could become the bottleneck of the whole system. Thus we proposed in Section 6.4 to approximate Eq. (6.1) and Eq. (6.2) by sub-sampling both the labeled and unlabeled identities in the denominators. We validate this approach here by training the framework with sub-sampling size of 10, 100, and 1000. The test mAP curves are demonstrated in Figure 6.7a. In general, sub-sampling a small number of identities relaxes the training objective, which leads to slightly inferior performance but much faster convergence rate. This indicates that our framework is scalable to larger datasets with even more identities by using proper sub-sampling rate.

Low-dimensional subspace. We further investigate how the dimension of the L2-normalized feature vector affects the person search performance. The results are summarized in Table 6.4. We observe that using the 2048-d global pooled feature vector directly with L2-normalization leads to lower training error, but its test performance is 20% worse. This suggests that projecting the features into a proper low-rank subspace is very important to regularize the network training. In our experiments, 256 to

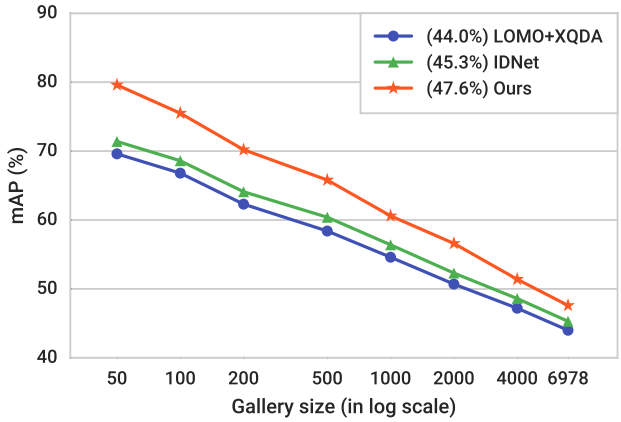
[2] https://github.com/Cysu/open-reid



(a) Sub-sampling size for the OIM loss



(b) Detection (LOMO+XQDA) recall



(c) Test gallery size

Figure 6.7: *Test mAP curves of different factors. The final mAPs are shown in the legend if applicable.*

1024 dimensions have similar test performance, and we choose 256-d to accelerate the computation of feature distances.

6.6.4 Factors for Person Search

Detection Recall. We investigate how detection recalls would affect the person search performance by using LOMO+XQDA as the re-id method and setting different thresholds on detection scores. A lower threshold reduces misdetections (increases the recall) but results in more false alarms. We choose the recall rates ranging from 30% to the maximum value of each detector. The final person search mAP under each setting is demonstrated in Figure 6.7b. An interesting observation is that higher recall does not necessarily lead to higher person search performance, which means re-id method could still get confused on some false alarms. This again indicates that we should not focus solely on training re-id methods with manually cropped pedestrians, but should consider the detections jointly under the person search problem setting.

Gallery size. Person search could be more challenging as the gallery size increases. We evaluate several methods under different test gallery sizes from 50 to full set of 6,978 images, following the protocols defined in Section 6.5.2. The test mAPs are demonstrated in Figure 6.7c. Note that for each test query, the corresponding gallery images are randomly sampled from the whole set. All test images are covered even with small gallery sizes. The performance gaps among different methods are reduced as the gallery size increases, indicating all the methods may suffer from some common hard samples, and we could further improve the performance with hard example minings.

6.7 Conclusions

In this chapter, we propose a new deep learning framework for person search. It jointly handles detection and identification in a single CNN. An Online Instance Matching loss function is proposed to train the network effectively. Its non-parametric nature enables faster yet better convergence, which is validated through series of experiments.

Chapter 7

Conclusions

In this dissertation we developed a series of deep learning based approaches to make human identification scalable towards real-world data and applications. From the data perspective, on one hand we addressed the challenge of lacking enough supervised data. A deep learning framework is proposed to utilize semi-supervised noisy-labeled data, which is much cheaper and efficient to collect. On the other hand, we tackled the problem of using multiple existing small datasets that each has its own data bias. A joint single task learning algorithm and domain guided dropout technique are developed to handle the domain bias explicitly in a single model. From the application perspective, we addressed on the more realistic problem setting, proposed a unique deep learning framework for simultaneous person detection and identification. A novel Online Instance Matching loss function is also proposed to learn identification features more effectively.

Despite our efforts toward scalable human identification, the problem itself still poses many challenges. We elaborated in Section 3.2 a technical roadmap for human identification. More research should be conducted to answer the questions of how to learn good features from a single image, how to compare a pair of feature maps, and how to structure a set of features. More application scenarios and problem settings are also worth investigation, including video-based human identification, as well as connecting images and text descriptions.

Bibliography

[1] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *arXiv:1406.2080*, 2014.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

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

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *CVIU*, 2008.

[5] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani, "Person re-identification by symmetry-driven accumulation of local features," in *CVPR*, 2010.

[6] S. Liao, Y. Hu, X. Zhu, and S. Z. Li, "Person re-identification by local maximal occurrence representation and metric learning," in *CVPR*, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.

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

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error-propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1.* MIT Press, Cambridge, MA, 1986, vol. 1, no. 6088, pp. 318–362.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[17] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *CVPR*, 2014.

[18] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *NIPS*, 2014.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.

[20] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.

[21] N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev, "Beyond frontal faces: Improving person recognition using multiple cues," in *CVPR*, 2015.

[22] W. Zajdel, Z. Zivkovic, and B. Krose, "Keeping track of humans: Have i seen this person before?" in *ICRA*, 2005.

[23] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *CVPR*, 2014.

[24] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *ICCV*, 2015.

[25] Y. Xu, B. Ma, R. Huang, and L. Lin, "Person search in a scene by jointly modeling people commonness and person uniqueness," in *ACM Multimedia*, 2014.

[26] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang, "Joint detection and identification feature learning for person search," in *CVPR*, 2017.

[27] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang, and Q. Tian, "Person re-identification in the wild," in *CVPR*, 2017.

[28] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *ECCV*, 2016.

[29] S. Li, T. Xiao, H. Li, B. Zhou, D. Yue, and X. Wang, "Person search with natural language description," in *CVPR*, 2017.

[30] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.

[31] M. K. Titsias, "One-vs-each approximation to softmax for scalable estimation of probabilities," in *NIPS*, 2016.

[32] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[33] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *ICML*, 2007.

[34] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *NIPS*, 2005.

[35] B. McFee and G. R. Lanckriet, "Metric learning to rank," in *ICML*, 2010.

[36] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," in *CVPR*, 2012.

[37] L. Zhang, T. Xiang, and S. Gong, "Learning a discriminative null space for person re-identification," in *CVPR*, 2016.

[38] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in *CVPR*, 2015.

[39] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *ICCV*, 2013.

[40] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *IJCV*, 2016.

[41] R. Zhao, W. Ouyang, and X. Wang, "Unsupervised salience learning for person re-identification," in *CVPR*, 2013.

[42] R. Zhao, W. Oyang, and X. Wang, "Person re-identification by saliency learning," *TPAMI*, 2017.

[43] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," *CVPR*, 2017.

[44] C. Liu, C. Change Loy, S. Gong, and G. Wang, "Pop: Person re-identification post-rank optimisation," in *ICCV*, 2013.

[45] S. Li, T. Xiao, H. Li, W. Yang, and X. Wang, "Identity-aware textual-visual matching with latent co-attention," in *ICCV*, 2017.

[46] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CVPR*, 2015.

[47] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[48] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015.

[49] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *CVPR*, 2016.

[50] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015.

[51] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *NIPS*, 2015.

[52] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *ICCV*, 2015.

[53] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *CVPR*, 2017.

[54] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *CVPR*, 2016.

[55] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *ECCV*, 2016.

[56] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *ICCV*, 2017.

[57] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[58] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," *arXiv preprint arXiv:1701.07717*, 2017.

[59] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, "Panda: Pose aligned networks for deep attribute modeling," in *CVPR*, 2014.

[60] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *TPAMI*, 2013.

[61] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *ECCV*, 2014.

[62] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik, "A multi-view embedding space for modeling internet images, tags, and their semantics," *IJCV*, 2014.

[63] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *TNNLS*, 2014.

[64] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial Intelligence Review*, 2004.

[65] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial intelligence review*, 2010.

[66] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and O. Pechenizkiy, "Class noise and supervised learning in medical domains: The effect of feature extraction," in *CBMS*, 2006.

[67] E. Beigman and B. B. Klebanov, "Learning with annotation noise," in *ACL-IJCNLP*, 2009.

[68] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE Transactions on Cybernetics*, 2013.

[69] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, 2006.

[70] C.-M. Teng, "A comparison of noise handling techniques," in *FLAIRS*, 2001.

[71] R. Barandela and E. Gasca, "Decontamination of training samples for supervised pattern recognition methods," in *ICAPR*, 2000.

[72] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *JAIR*, 1999.

[73] A. L. Miranda, L. P. F. Garcia, A. C. Carvalho, and A. C. Lorena, "Use of classification algorithms in noise detection and elimination," in *HAIS*, 2009.

[74] N. Matic, I. Guyon, L. Bottou, J. Denker, and V. Vapnik, "Computer aided cleaning of large databases for character recognition," in *IAPR*, 1992.

[75] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *ICML*, 2012.

[76] J. Larsen, L. Nonboe, M. Hintz-Madsen, and L. K. Hansen, "Design of robust neural network classifiers," in *ICASSP*, 1998.

[77] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *NIPS*, 2013.

[78] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Technical Report CMU-CALD-02-107, Carnegie Mellon University, Tech. Rep., 2002.

[79] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, 2012.

[80] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *CVPR*, 2014.

[81] H. Azizpour, A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *CVPR Workshops*, 2015.

[82] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition." in *ICML*, 2014.

[83] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM Multimedia*, 2014.

[84] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop*, 2013.

[85] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014.

[86] W. Li and X. Wang, "Locally aligned feature transforms across views," in *CVPR*, 2013.

[87] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof, "Person re-identification by descriptive and discriminative classification," in *Scandinavian Conference on Image Analysis (SCIA)*, 2011.

[88] D. Gray, S. Brennan, and H. Tao, "Evaluating appearance models for recognition, reacquisition, and tracking," in *PETS*, 2007.

[89] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[90] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[91] X. Chu, W. Ouyang, H. Li, and X. Wang, "Structured feature learning for pose estimation," in *CVPR*, 2016.

[92] W. Yang, W. Ouyang, H. Li, and X. Wang, "End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation," in *CVPR*, 2016.

[93] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in *CVPR*, 2016.

[94] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *ICML*, 2015.

[95] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015.

[96] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *ICCV*, 2015.

[97] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *NIPS*, 2012.

[98] J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *NIPS*, 2013.

[99] R. Zhao, W. Ouyang, and X. Wang, "Person re-identification by salience matching," in *ICCV*, 2013.

[100] ——, "Learning mid-level filters for person re-identification," in *CVPR*, 2014.

[101] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognition*, 2015.

[102] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Learning to rank in person re-identification with metric ensembles," in *CVPR*, 2015.

[103] F. Xiong, M. Gou, O. Camps, and M. Sznaier, "Person re-identification using kernel-based metric learning methods," in *ECCV*, 2014.

[104] S. Liao, Z. Mo, Y. Hu, and S. Z. Li, "Open-set person re-identification," *arXiv preprint arXiv:1408.0872*, 2014.

[105] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.

[106] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR*, 2014.

[107] Y. Kawanishi, Y. Wu, M. Mukunoki, and M. Minoh, "Shinpuhkan2014: A multi-camera pedestrian dataset for tracking people across multiple cameras," in *20th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, 2014.

[108] D. Baltieri, R. Vezzani, and R. Cucchiara, "3dpes: 3d people dataset for surveillance and forensics," in *ACM workshop on Human gesture and behavior understanding*, 2011.

[109] W.-S. Zheng, S. Gong, and T. Xiang, "Associating groups of people," in *BMVC*, 2009.

[110] H. Moon and P. J. Phillips, "Computational and performance aspects of pca-based face-recognition algorithms," *Perception-London*, 2001.

[111] N. Gheissari, T. B. Sebastian, and R. Hartley, "Person reidentification using spatiotemporal appearance," in *CVPR*, 2006.

[112] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern recognition letters*, 2013.

[113] S.-I. Yu, Y. Yang, and A. Hauptmann, "Harry potter's marauder's map: Localizing and tracking multiple persons-of-interest by nonnegative discretization," in *CVPR*, 2013.

[114] C. C. Loy, T. Xiang, and S. Gong, "Multi-camera activity correlation analysis," in *CVPR*, 2009.

[115] T. Xiao, H. Li, W. Ouyang, and X. Wang, "Learning deep feature representations with domain guided dropout for person re-identification," in *CVPR*, 2016.

[116] X. Wang, G. Doretto, T. Sebastian, J. Rittscher, and P. Tu, "Shape and appearance context modeling," in *ICCV*, 2007.

[117] O. Hamdoun, F. Moutarde, B. Stanciulescu, and B. Steux, "Person re-identification in multi-camera system by signature based on interest point descriptors collected on short video sequences," in *ICDSC*, 2008.

[118] B. Prosser, W.-S. Zheng, S. Gong, T. Xiang, and Q. Mary, "Person re-identification by support vector ranking." in *BMVC*, 2010.

[119] F. Porikli, "Inter-camera color calibration by correlation model function," in *ICIP*, 2003.

[120] Y. Shen, W. Lin, J. Yan, M. Xu, J. Wu, and J. Wang, "Person re-identification with correspondence structure learning," in *ICCV*, 2015.

[121] W.-S. Zheng, S. Gong, and T. Xiang, "Person re-identification by probabilistic relative distance comparison," in *CVPR*, 2011.

[122] D. Gray and H. Tao, "Viewpoint invariant pedestrian recognition with an ensemble of localized features," in *ECCV*, 2008.

[123] S. Liao and S. Z. Li, "Efficient psd constrained asymmetric metric learning for person re-identification," in *ICCV*, 2015.

[124] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *CVPR*, 2016.

[125] X. Li, W.-S. Zheng, X. Wang, T. Xiang, and S. Gong, "Multi-scale learning for low-resolution person re-identification," in *ICCV*, 2015.

[126] W.-S. Zheng, X. Li, T. Xiang, S. Liao, J. Lai, and S. Gong, "Partial person re-identification," in *ICCV*, 2015.

[127] M. Hirzer, C. Beleznai, P. M. Roth, and H. Bischof, "Person re-identification by descriptive and discriminative classification," in *Image Analysis*, 2011.

[128] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *TPAMI*, 2010.

[129] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *TPAMI*, 2014.

[130] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," in *CVPR*, 2015.

[131] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in *ICCV*, 2015.

[132] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster r-cnn doing well for pedestrian detection?" in *ECCV*, 2016.

[133] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a deeper look at pedestrians," in *CVPR*, 2015.

[134] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian detection aided by deep learning semantic tasks," in *CVPR*, 2015.

[135] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for deep pedestrian detection," in *ICCV*, 2015.

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

[137] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, 2014.

[138] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: towards good practices for deep action recognition," in *ECCV*, 2016.

[139] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV Workshop*, 2016.