Deep similarity metric learning for multiple object tracking

Bonan Cuan

N°d'ordre NNT : 2019LYSEI065

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA LYON

Ecole Doctorale N° 512
INFORMATIQUE ET MATHEMATIQUES

Spécialité de doctorat : Informatique et applications

Soutenue publiquement le 12/09/2019, par :
Bonan CUAN

Deep Similarity Metric Learning for Multiple Object Tracking

Devant le jury composé de :

PAINDAVOINE, Michel	Professeur	Université de Bourgogne	Président
CHATEAU, Thierry	Professeur	UCA	Rapporteur
CAPLIER, Alice	Professeur	Grenoble-INP	Rapporteure
GARCIA, Christophe	Professeur	INSA Lyon	Co-directeur de thèse
IDRISSI, Khalid	Maître de conférences, HDR	INSA Lyon	Directeur de thèse

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

I would like to dedicate this thesis to my loving parents and fiancée, Xiaoxiao. ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Bonan CUAN
December 2019

Acknowledgements

And I would like to acknowledge my thesis supervisors, Christophe Garcia and Khalid Idrissi, for their instruction and support in years. I would also like to express my gratitude to all the jury members, Thierry Chateau, Alice Caplier and Michel Paindavoine, for reporting and examining this dissertation. Thanks to my collegues in INSA Lyon and laboratory LIRIS for providing a great environment for learning and research. I am very grateful to my family and friends, especially my dear parents for their endless and selfless love in my entire life. Special thanks to my future wife, Xiaoxiao, for her priceless love, companion and support.

Abstract

Multiple object tracking, i.e. simultaneously tracking multiple objects in the scene, is an important but challenging visual task. Objects should be accurately detected and distinguished from each other to avoid erroneous trajectories. Since remarkable progress has been made in object detection field, "tracking-by-detection" approaches are widely adopted in multiple object tracking research. Objects are detected in advance and tracking reduces to an association problem: linking detections of the same object through frames into trajectories.

Most tracking algorithms employ both motion and appearance models for data association. For multiple object tracking problems where exist many objects of the same category, a fine-grained discriminant appearance model is paramount and indispensable. Therefore, we propose an appearance-based re-identification model using deep similarity metric learning to deal with multiple object tracking in mono-camera videos. Two main contributions are reported in this dissertation:

First, a deep Siamese network is employed to learn an end-to-end mapping from input images to a discriminant embedding space. Different metric learning configurations using various metrics, loss functions, deep network structures, etc., are investigated, in order to determine the best re-identification model for tracking. In addition, with an intuitive and simple classification design, the proposed model achieves satisfactory re-identification results, which are comparable to state-of-the-art approaches using triplet losses. Our approach is easy and fast to train and the learned embedding can be readily transferred onto the domain of tracking tasks.

Second, we integrate our proposed re-identification model in multiple object tracking as appearance guidance for detection association. For each object to be tracked in a video, we establish an identity-related appearance model based on the learned embedding for re-identification. Similarities among detected object instances are exploited for identity classification. The collaboration and interference between appearance and motion models are also investigated. An online appearance-motion model coupling is proposed to further improve the tracking performance. Experiments on Multiple Object Tracking Challenge benchmark prove the effectiveness of our modifications, with a state-of-the-art tracking accuracy.

Résumé

Le suivi d'objets multiples dans une scène est une tâche importante dans le domaine de la vision par ordinateur, et présente toujours de très nombreux verrous. Les objets doivent être détectés et distingués les uns des autres de manière continue et simultanée. Les approches «suivi par détection» sont largement utilisées, où la détection des objets est d'abord réalisée sur toutes les frames, puis le suivi est ramené à un problème d'association entre les détections d'un même objet et les trajectoires identifiées. La plupart des algorithmes de suivi associent des modèles de mouvement et des modèles d'apparence.

Dans cette thèse, nous proposons un modèle de ré-identification basé sur l'apparence et utilisant l'apprentissage de métrique de similarité. Nous faisons tout d'abord appel à un réseau siamois profond pour apprendre un maping de bout en bout, des images d'entrée vers un espace de caractéristiques où les objets sont mieux discriminés. De nombreuses configurations sont évaluées, afin d'en déduire celle offrant les meilleurs scores. Le modèle ainsi obtenu atteint des résultats de ré-identification satisfaisants comparables à l'état de l'art.

Ensuite, notre modèle est intégré dans un système de suivi d'objets multiples pour servir de guide d'apparence pour l'association des objets. Un modèle d'apparence est établi pour chaque objet détecté s'appuyant sur le modèle de ré-identification. Les similarités entre les objets détectés sont alors exploitées pour la classification. Par ailleurs, nous avons étudié la coopération et les interférences entre les modèles d'apparence et de mouvement dans le processus de suivi. Un couplage actif entre ces 2 modèles est proposé pour améliorer davantage les performances du suivi, et la contribution de chacun d'eux est estimée en continue. Les expérimentations menées dans le cadre du benchmark «Multiple Object Tracking Challenge» ont prouvé l'efficacité de nos propositions et donné de meilleurs résultats de suivi que l'état de l'art.

Contents

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
1.1 Context 1
1.2 Pipeline of multiple object tracking 2
1.3 Contribution 2
1.4 Outline 3

2 Literature review **5**
2.1 Introduction 5
2.2 Deep learning basics 6
2.2.1 Multi-layer perceptron 6
2.2.2 Convolutional neural network 10
2.2.3 State-of-the-art CNNs 12
2.3 Region-based object recognition 14
2.3.1 Localization through classification 14
2.3.2 Regions with CNN features 16
2.3.3 CNN-based location regression 17
2.3.4 Deep learning in pixel-level segmentation 20
2.3.5 Transfer learning for deep neural networks 22
2.4 Object re-identification and metric learning 23
2.4.1 Re-identification algorithms 23
2.4.2 Metric learning and Siamese network 25
2.5 Multiple object tracking in mono-camera videos 28
2.5.1 Tracking by detection with motion and appearance models 29
2.5.2 Formulation of data association 30

2.6 Conclusion and insights 31

3 Object Re-identification with Similarity Metric Learning 35
3.1 Introduction 35
3.2 Deep Siamese network for metric learning 37
3.2.1 Siamese structure 37
3.2.2 Metrics 40
3.2.3 Pairwise loss functions 41
3.2.4 Training data preparation 44
3.3 Classification-guided similarity metric learning 44
3.3.1 Motivation 44
3.3.2 Metric learning with classification loss 46
3.4 Re-identification in mono-camera video tracking 49
3.4.1 Comparison with verification-modeled structures 49
3.4.2 Comparison with classification-modeled structures 50
3.4.3 Auxiliary information for re-identification 51
3.5 Experiments and discussion 52
3.5.1 Influence of classification guidance 52
3.5.2 Comparison of metrics 54
3.5.3 Determination of regularizer λ 55
3.5.4 Influence of embedding space dimensionalities 56
3.5.5 Various head networks 57
3.5.6 Stride variation of feature maps 59
3.5.7 Different backbone networks 60
3.5.8 Comparison with the state of the art 61
3.6 Conclusion 62

4 Metric Learning for Multiple Object Tracking 67
4.1 Introduction 67
4.2 Data association formulation in tracking-by-detection 68
4.2.1 Data association graph models 68
4.2.2 Edge weights in tracking graphs 72
4.3 Appearance model with learned cosine similarity 73
4.3.1 Track hypothesis representation 73
4.3.2 Edge weights under softmax classification 76
4.4 Online appearance-motion coupling 78
4.4.1 Online model credibility for inference 78

4.4.2 Online model update 81
4.5 Experiments and discussion 83
4.6 Conclusion 85

5 Conclusion and Perspectives **89**
5.1 Conclusion 89
5.2 Perspectives 90

Bibliography **93**

Appendix A Binary operation for vectors with singleton expansion **103**

Appendix B Region-based tracking **105**

List of Figures

2.1 A linear neuron 7
2.2 A fully-connected MLP 8
2.3 Sobel operator and 2D convolution 9
2.4 AlexNet architecture 12
2.5 Selective search example 15
2.6 R-CNN system overview 17
2.7 Faster R-CNN illustration 18
2.8 Siamese network 26
2.9 "Stacked" network fusing body part information 28
2.10 An overview of the Multiple Object Tracking Challenge benchmark 29
2.11 Data association formulations 32

3.1 Samples from re-identification benchmark CUHK03 36
3.2 siamese structure 38
3.3 Feature pyramid network 39
3.4 Metrics illustration 40
3.5 Normalized distances 42
3.6 Re-identification via classification with a multitask fashion 46
3.7 Classification-guided metric learning 47
3.8 Normalized similarities 48
3.9 Experimental result contrast between Siamese network and its classification-guided version 53
3.10 Experimental result contrast among metrics 54
3.11 Experimental result contrast under different λ values 56
3.12 Influence of embedding space dimensionality 57
3.13 Comparison among various head structures 58
3.14 Influence on re-identification accuracy of feature map strides 60
3.15 Performance of different backbone networks 61

3.16 Comparison of re-identification performance on MOT datasets 65

4.1 Tracking graph model example 69
4.2 Intermediate trimming 71
4.3 Illustration of the proposed appearance model 74
4.4 Example of adjacent similar looking people in MOT16 79
4.5 Illustration of hypotheses concurrences 80
4.6 Example of tracker drift 81
4.7 Example of tracking results 86

List of Tables

3.1 Results on the MOT16 benchmark compared with leader algorithms 62

4.1 Ablation comparison of online model coupling strategies on the MOT16 benchmark 83
4.2 Results on the MOT16 benchmark compared with leader algorithms 84

Chapter 1

Introduction

1.1 Context

Within the scope of artificial intelligence and computer vision, object tracking is a visual task aiming at automatically identifying and localizing an object or objects of interest in videos or image sequences, and forming a trajectory or trajectories through time. The tracking result is important for further object behavior recognition and analysis. As Yilmaz et al. [113] pointed, object tracking is a key step in video analysis and is pertinent in visual tasks like automated video surveillance, human-computer interaction, unmanned/self-driving vehicle system, video compression, etc.

After being well studied for decades, object tracking remains a challenge. Problems like occlusions, scene clutter, abrupt motion of object and/or camera, non-rigid object deformation, object pose variation, etc., can prevent tracking algorithms from constantly and precisely finding object trajectories.

The situation deteriorates in multiple object tracking (MOT) problem, i.e. simultaneously tracking multiple objects in the scene. Object-object occlusion is more prone to occur, and interaction between objects makes their trajectories more difficult to estimate or even unpredictable by a stochastic model. Intersection of trajectories of similar objects may also raise the risk of trajectory merging or identity confusion [75]. Research on robust multiple object tracking algorithms is meaningful and necessary.

In this thesis, we focus on multiple object tracking in single-camera short-term outdoor videos where object appearance remains unchanged except for pose and illumination variation. Therefore, we propose to employ a robust appearance-based model to overcome the problems like occlusions and abrupt motion. The research is conducted on open-world datasets where objects for training rarely reappear in testing sets. Facing such specific constraints, object re-identification is introduced to learn a comparison model for tracking. Our new model

is combined with tracking-by-detection frameworks to achieve state-of-the-art tracking performance.

1.2 Pipeline of multiple object tracking

Every tracking method requires object detection [113]. In multiple object tracking problems, objects undergo sudden appearances or disappearances as well as long-term partial or complete occlusions. Therefore, a robust and constant detection in every frame is necessary to recover all object instances. Thanks to the significant progress in object recognition field [54, 94, 42, 35, 34, 84, 39], multiple object tracking has witnessed a burst of algorithms under the tracking-by-detection paradigm.

As the name indicates, tracking-by-detection approaches rely on robust detection independent of tracking. Objects are detected in each frame and then associated into trajectories through frame to frame, during which their appearance and motion information serves as important guidance of track inference [7, 8, 53, 102]. In another word, tracking task is decomposed into two separate phases: object detection and data association. The most creative work in object tracking is the design of robust and fast data association algorithms.

Within data association frameworks [53, 102], appearance models and motion models are critical for accurate instance matching. How to design and maintain a robust appearance model for multiple object tracking is a main research job in my thesis. Besides, the collaboration and interference of appearance and motion models is an interesting topic that can ameliorate tracking result.

1.3 Contribution

In this dissertation, we report two main contributions to the state of the art in mono-camera video-based multiple object tracking domain.

First, we propose a novel image-based object re-identification model using deep similarity metric learning. By reviewing and investigating metric learning structures, we employ an identity classification guidance for direct similarity metric learning without extra overhead. The learned re-identification model performs much better than the vanilla one.

Second, we introduce the re-identification model as an appearance model into multiple object tracking frameworks, which leads to an improvement in tracking performance. Modification made on the tracking-by-detection framework for the insertion of our model, as well as for balancing the influence of appearance and motion models, is reported.

1.4 Outline

In the next chapter, we review the previous work related to our multiple object tracking tasks. Modern tracking algorithms depend heavily on object recognition. For example, most tracking-by-detection approaches require a robust object detector while employ discriminant recognition models in data association. Hence, we first review the history and the state of the art of object recognition. The application of deep learning in this field is emphasized. Re-identification, among various kinds of object recognition tasks, is thoroughly discussed as it is the leading solution of appearance guidance in multiple object tracking. In this part, we focus on metric learning, an elegant and effective approach of realizing re-identification. Finally, we review current multiple object tracking approaches, especially those with the tracking-by-detection scheme. Different formulations of data association are introduced, along with the exploitation of appearance and motion clues for tracking.

Some insights on tracking performance improvement are drawn after the review. In Chapter 3, we come up with a robust re-identification model based on deep metric learning. Various model configurations are thoroughly investigated and contrasted. By simply appending a softmax classifier on the learned embedding, our re-identification model is significantly improved. Easy to train, well-designed for tracking tasks, our model still achieves satisfactory results in experiments.

In Chapter 4, we integrate the re-identification model into a state-of-the-art multiple object tracking framework as appearance guidance. In addition, we improve the collaboration between appearance and motion models. The online coupling of models helps alleviate track errors. Experiments on multiple tracking benchmarks prove our algorithm superior to the state of the art with the best tracking results.

At last, we conclude the thesis in Chapter 5. Some perspectives for future work are also presented.

Chapter 2

Literature review

2.1 Introduction

As summarized in Chapter 1, the objective of this thesis is to study the application of deep similarity metric learning in the field of multiple object tracking in mono-camera videos. Before discussing deep metric learning, some deep learning basics are reviewed in the first place. Classic deep learning structures like multi-layer perceptrons and convolutional networks are presented in the next section, along with techniques like non-linear activation, pooling, etc. By revisiting the history of deep learning, some state-of-the-art backbone convolutional networks for image classification are briefly introduced. Embedded in other networks, they have remarkably boosted the research of many visual recognition tasks, from generic object detection to pixel-to-pixel segmentation. The progress in these fields achieved with the help of deep learning techniques are detailed in Section 2.3, since it is fundamental to deep metric learning.

Research fields directly related to this thesis are reviewed right after. Multiple object tracking requires differentiation of object instances through video frames. Appearance-based object re-identification is one of the most adopted solutions. In Section 2.4, we discuss the history and progress of re-identification and focus on metric learning approaches, which are the state of the art in this field, from linear metric learning with engineered feature, to ingenious Siamese networks yielding nonlinear transformations. A good metric is critical for the success of metric learning. Some distance and similarity metrics are introduced and illustrated, followed by a comparison between metric learning-based reidentification and other approaches. The application of re-identification in multiple object tracking is presented in Section 2.5. By reviewing some state-of-the-art tracking algorithms, we demonstrate the function of appearance-based re-identification in the “tracking-by-detection” framework. Also, the relationship between appearance models and motion models is discussed.

To conclude this chapter, we summarize some insights obtained from previous work at the end. These insights for designing a good multiple object tracking algorithm will be detailed in the rest of this dissertation.

2.2 Deep learning basics

Feature extraction is fundamental to machine learning. In the history of pattern recognition, several ingenious feature representation methods, e.g. scale-invariant feature transform (SIFT) [66] and histogram of oriented gradients (HOG) [23], significantly boosted the progress of machine learning and artificial intelligence. These features are manually engineered using expert domain knowledge of specific tasks. However, coming up with features is difficult, time-consuming, and requires expert knowledge [71]. A promising solution is learning the feature representation itself. Deep learning proved to be very successful as a representation learning method.

Most modern deep learning models are based on artificial neural networks, which are vaguely inspired by human brains. An artificial neuron collects information from other cells to form its specific output. The first generation of neural networks is called *perceptron* [87]. A perceptron, the antetype of modern artificial neuron, learns a proper weight for each of its inputs (a bunch of engineered features) and outputs their weighted sum for binary classification [45]. A population of artificial neurons or basic perceptrons interrelate with each other and form artificial neural networks, aiming at more complex tasks. Although researchers like Minsky and Papert [69] criticized early-stage neural networks for their limitations, deep learning keeps evolving until being sophisticated enough to tackle with most of the computer vision and object recognition problems, as computers achieve far greater computational power and breakthrough algorithms (e.g. backpropagation [110, 88]) are proposed. In this section, we review some widely used basic deep learning techniques.

2.2.1 Multi-layer perceptron

Multi-layer perceptron (MLP), which consists mainly of perceptrons, is the simplest neural network. For any input m-dimensional vector $\mathbf{x} \in \mathbb{R}^m$, a linear neuron outputs a weighted scalar sum $y \in \mathbb{R}$ using its own weight vector (called *parameter* of this neuron) $\mathbf{w} \in \mathbb{R}^m$:

$$y = \sum_{i=1}^{m} w_i x_i = \mathbf{w}^T \mathbf{x} \tag{2.1}$$

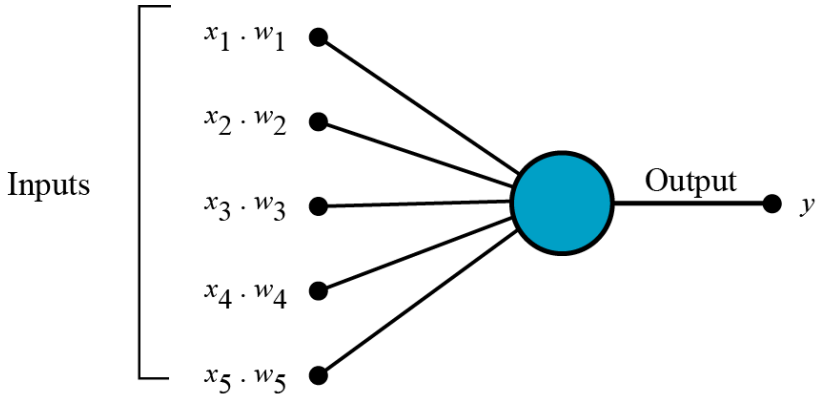


Figure 2.1 A linear neuron. Each neuron has its own parameter vector $\mathbf{w}$ which is multiplied with its input $\mathbf{x}$ to output a scalar y. When y is directly used for classification, the neuron is a perceptron. Given the same input vector $\mathbf{x}$, a layer consists of multiple neurons in parallel and outputs a vector $\mathbf{y}$.

Training a linear neuron (perceptron) means learning an optimal parameter $\mathbf{w}^*$ that minimizes the error between its actual output y and the desired output y^* in the ground truth (also called training *label*) for all training inputs. The error or *loss* is calculated with a loss function L, e.g. the squared loss $L_{quad}(y,y^*) = (y-y^*)^2$.

One single linear neuron is weak: scalar output means the m-dimensional inputs are reduced onto 1-dimensional space. However, it is convenient to juxtapose neurons in parallel: multiple neurons with different parameters $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n$ share the same input $\mathbf{x}$ and yield independent outputs $y_1, y_2, \cdots, y_n$. Often, bias terms $b_1, b_2, \cdots, b_n \in \mathbb{R}$ are also introduced. A tight format is as below with matrices:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} + \mathbf{b} \tag{2.2}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n] \in \mathbb{R}^{m\times n}$ and $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T, \mathbf{b} = [b_1, b_2, \cdots, b_n]^T \in \mathbb{R}^n$. The vector of these neurons is called a *layer* and the weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ are its two parameters. Usually, all the neurons in a layer are connected to the entire input (each component of vector $\mathbf{x}$), and the layer is called *fully-connected*. The optimization of $\mathbf{W}$ and $\mathbf{b}$ towards their solutions $\mathbf{W}^*$ and $\mathbf{b}^*$ is now driven by the loss between $\mathbf{y}$ and its label $\mathbf{y}^*$ under certain loss function $L : \mathbb{R}^n \mapsto \mathbb{R}$:

$$\mathbf{W}^*, \mathbf{b}^* = \arg\min_{\mathbf{W},\mathbf{b}} L(\mathbf{y}, \mathbf{y}^*) \tag{2.3}$$

As the name of MLP indicates, another way of fortifying the structure is stacking multiple layers: a layer's output vector becomes another layer's input. In feedforward networks, a

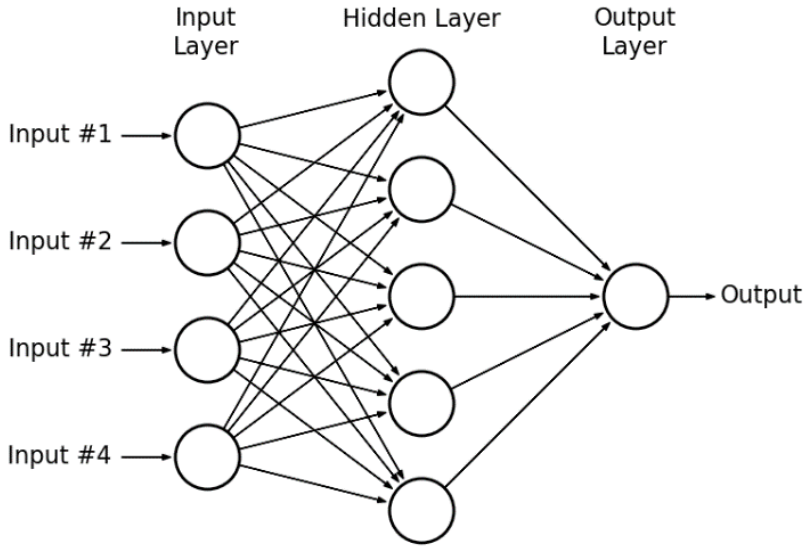


Figure 2.2 An example of fully-connected MLP. Its output can either be a vector or a simply a scalar (1-dimensional vector). It has a set of parameters to optimize and techniques like backpropagation are applied during training.

layer solely nourishes its following layer (sometimes layers), but neither its precedents nor itself. The stack of layers forms a structured and hierarchical network where interconnections among layers are unidirectional. Such feedforward networks have only one-way information flows through several layers with no loop. The intermediate layers situated between input and output are called hidden layers. MLPs are *deep neural networks* (DNNs) when they have more than one hidden layer. The advantage of deep networks over a shallow one with only one or even no hidden layer lies mostly in computational efficiency. Much less neurons will be needed in a deep network (polynomial rather than exponential in the input size m) to illustrate the same transformation from input to output, when compared to a huge shallow one.

However, direct concatenation of linear layers is meaningless in terms of functionality: the associativity of matrix multiplication makes such layered structure replaceable by a single layer, while a linear network can never resolve "wrap-around" problems like XOR operation [69]. As an amendment, nonlinear *activation* functions, e.g. Sigmoid, TanH and the family of Rectified Linear Unit (ReLU) [50, 70], are added between consecutive layers. With their help, MLPs acquire the capability of representing nonlinear patterns and become universal approximators [47].

However, the deeper a network is, the harder it is to train, especially for its hidden layers. Backpropagation [88] was invented in order to train in one pass every layer of a network. Loss of the final output is propagated backwards to all neurons directly or indirectly connected to

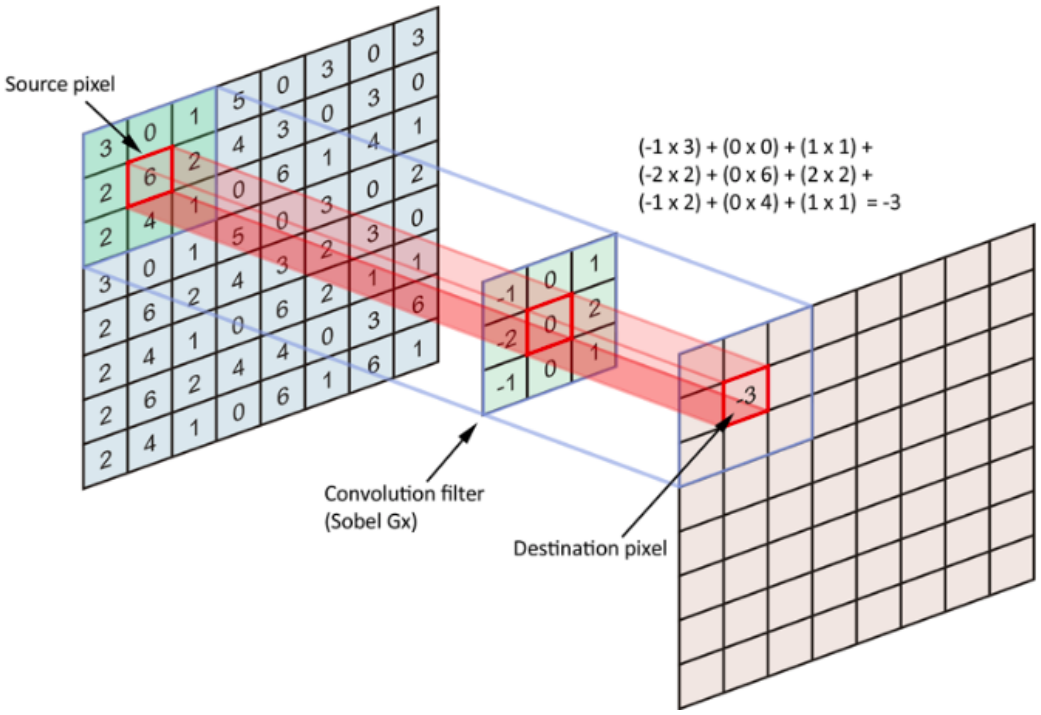


Figure 2.3 Sobel operator [96] is a perfect example of 2D convolution application in computer vision. Convolutional neural networks are designed to learn rather than engineer similar filters of different granularities.

the output, based on their partial derivatives/gradients. With backpropagation, gradient-based approximation methods like stochastic gradient descent (SGD) are applied to tune the whole network towards an optimized model. Denoting the entire transformation conducted by a network as a nonlinear function $\mathbf{y} = f(\mathbf{x})$, training the network is the process of finding the optimal parameter set (without differentiating bias terms) of all its layers $\{\mathbf{W}\}^*$:

$$\{\mathbf{W}\}^* = \arg\min_{\{\mathbf{W}\}} L(f(\mathbf{x}), \mathbf{y}^*) \tag{2.4}$$

In modern computer vision tasks, MLP is not suitable for large image inputs as a huge number of weights would be necessary in the first layers. A more practical way is implementing multiple fully-connected inner-product layers at the very end of neural networks, for high-level reasoning. Before benefitting their global connectivity, the size of original image input needs to be significantly reduced and extracted as low-dimensional features by other deep learning techniques like convolutional layers.

2.2.2 Convolutional neural network

Convolutional neural networks (CNNs) are a special kind of feedforward neural network containing convolutional layers. The difference between convolutional layers and common fully-connected layers introduced in 2.2.1 lies in how the neurons in different layers are connected. Convolution, a mathematical operation widely used for signal processing, takes the place of matrix multiplication so that such networks are named after it. By its definition, convolution requires a flipped *kernel*. However, in most machine learning libraries, convolution operation is implemented as cross-correlation, given the fact that the exact order (flipped or not) of element multiplication between input and kernel is not important at all in CNNs. The famous Sobel operate [96] for edge detection is based on simple 2D convolution with an engineered kernel.

In the scope of computer vision, the most common original inputs are geometrically two-dimensional digital images with a third dimension, image channels (RGB channels for example). 2D convolution is therefore extended for 3D input where the kernel is also a 3D matrix which has the same size in the channel dimension as the input. The 3D kernel performs nearly the same 2D convolution as its 2D counterpart, only the scalar product between the kernel and input patches is extended on 3D space. Such 2D convolution with 3D input yields the same 2D output as shown in Fig. 2.3.

With the help of the extended operation, convolutional layers often have a set of independent kernels to learn multiple filters at the same time. Their 2D outputs are stacked along the channel dimension. The channel number of the final output or *feature map*, equals the cardinality of the kernel set and is independent of the input. Given that the input and output are both 3D matrices, convolutional layers can be concatenated together as a network. As the output of convolution has the same spatial arrangement as its input, such network explicitly preserves the geometric information of its input layer by layer.

Kernels in state-of-the-art networks often have tiny sizes (e.g. 3×3). They respond only to small patches rather than the entire input, and their outputs are therefore geometrically limited. Meanwhile, the parameter of a kernel remains invariant as the sliding window moves. In another word, convolutional layers are a special kind of inner-product layers that perform localized weighted sum on all the input patches with the same parameter. Information extraction is limited within each patch while inter-channel exchange is allowed thanks to 2D convolution with 3D input.

Such *parameter sharing* mechanism leads to some useful properties. First, convolution layers substitute MLPs' dense input-output interaction with sparse local ones using a single parameter set, which significantly alleviates the demands of computational resources. Given

that low-level features are rarely globally connected to each other, convolutional layer is an excellent design when deployed at the first stages of representation learning.

Second, the convolutional kernels are driven to extract *translation-equivariant* features. The global location of features barely has influence on the output while only the local pattern matters. Cognitively salient objects may appear in any position in an image and location-irrelevance of features yielded by parameter sharing is therefore paramount.

Apart from saving computational power as in MLP, the concatenation of convolutional layers implies feature grouping. A deep neuron receives information from a patch of its input which is the feature map of its antecedent convolutional layer. On the one hand, a deeper layer groups the features of a shallower layer into higher-level ones. On the other hand, neurons of the deeper layer are influenced by larger regions of the original image (called *receptive fields*) than their predecessors, which helps extract more global features. Given the small size of kernels in modern CNNs, the features learned in layers are forced to gradually grow sophisticated and abstract. In another word, cascade of convolutional layers is able to realize perceptual grouping [65], only features and grouping laws are both learned rather than hand-engineered.

There are also side effects using small kernels. Their sensitivity to small local variation is preferable when dealing with low-level features than higher-level ones, which demand invariance to some degree. Goodfellow et al. [37] took human face for example: only the presence of eyes in certain regions, rather than their precise locations, determines whether a face exists; the feature representation should be tolerant of local inaccuracy within a rough limit.

To achieve this goal, *pooling* is introduced to report a summary of regions. Mean-pooling calculates the mean value over the region, while a more powerful and more adopted function, max-pooling, outputs the maximum activation. Such pooling functions ensure small variations within the spatial region has little or even no influence on the output. Also, pooling operation reduces the layer size, which further saves computational resources. Nowadays, the successful pooling layers and its "coarse coding" seem to be harmful for precise object recognition tasks, as Geoffrey Hinton pointed [1], especially when applied on high-level features in deep layers. With the burst of GPU computation, cutting-edge convolutional networks use much fewer pooling layers.

1. https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyj4jv/

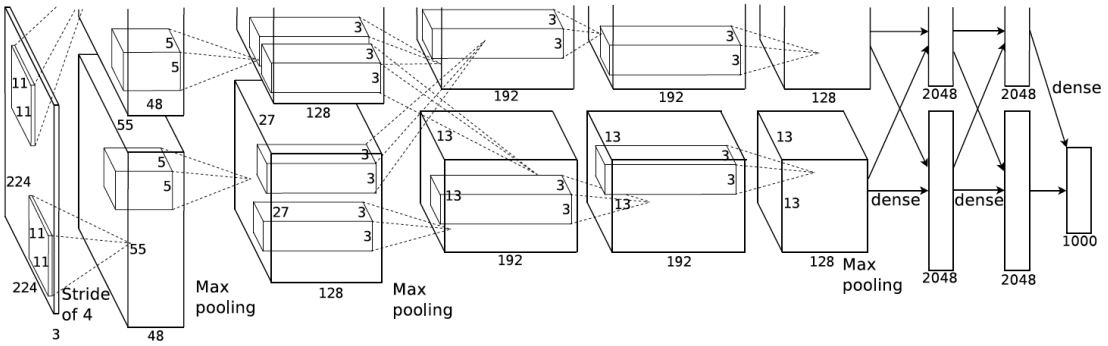


Figure 2.4 The AlexNet architecture, copied directly from [54]. The network employs a convolutional subnet (2D convolution with 3D input) for feature extraction and a 3-layer fully-connected subnet for classification reasoning.

2.2.3 State-of-the-art CNNs

As mentioned above, state-of-the-art neural networks use a deep convolutional network to extract translation-equivariant features from images. The final feature map as well as certain intermediate maps when necessary, are often linked to one or several multi-layer fully-connected networks for global reasoning. A classifier (e.g. Softmax classifier) or regressor with suitable loss functions is appended at last. The network is trained with the help of backpropagation. For multi-task learning, we may find several losses working together.

The first modern deep CNN is AlexNet [54], winner of the ImageNet Large Scale Visual Recognition Challenge in 2012 (ILSVRC12) [26, 89]. It laid the foundation for the recent burst of GPU-based DNNs. The canonical structure with 5 convolutional layers and 3 fully-connected layers is passed down through all its followers.

In the next stage, CNNs became deeper and more complex. It was empirically proven that under the same time constraint, a deeper network could achieve better performance than a shallower one [40]. Hence, convolutional layers with large kernels (e.g. 11×11) in AlexNet were replaced by sets of sublayers with smaller kernels. For isntance, the popular VGG16 network (16 layers in total) in the family of VGG (standing for Visual Geometry Group of University of Oxford) nets [94], used only 3×3 kernels and the original 5 convolutional layers in AlexNet were split into a series of two or three sublayers. Another famous example is GoogLeNet [99]. In addition to serial sublayers, GoogLeNet also parallelly expanded the structure. The inception layer proposed in GoogLeNet convolves its input with kernels of different sizes in parallel at the same time. By concatenating all these convolutional outputs together, the layer aims to extract a feature containing information of various granularities.

CNNs kept growing deeper until their performance growth contrasted the concept "deeper is better". Despite the notorious vanishing gradient problem which has been dealt with by intermediate normalization layers (e.g. batch norm), very deep networks have intrinsic defects. When networks are deep enough (more than 30 layers), their image classification accuracy becomes saturated and even degrades [42]. The possible cause might be the inability of current optimization methods facing very deep systems. He et al. [42] proposed deep residual learning to reduce the optimization burden of the solvers, by adding shortcut connections between originally non-adjacent layers. All the layers within the range of the shortcut form a block of network, as shown in FIGURE. The shortcut performs an identity mapping (i.e. a copy of the first layer of the block) and bypasses a stack of intermediate sublayers, which would be the only branch in traditional networks like VGG. The last layer aggregates both branches by outputting their sum.

In non-residual networks, the transformation is expected to be improved layer by layer, which may be difficult to achieve for very deep layers. In residual networks, such objective is factored into two parts: maintaining the input transformation while calculating a residual transformation as its improvement. The identity mapping shortcut relieves the stacked layers from the first objective by providing a reference of established transformation. Unlike the highway networks [97, 98] using gated shortcuts with parameters, identity mapping in residual networks is a simple copy and never closed (always referencing the input transformation).

The basic hypothesis in residual learning is that the residual is easier and faster to learn than the original transformation. A residual learning block is more efficient than its branch of stacked layers alone. Built with such residual blocks, extremely deep networks are therefore feasible. The hypothesis was backed up by experiments [42]: deep residual networks with more than 100 layers were trained with ease and achieved better image classification accuracy than non-residual structures.

The success of deep residual learning has been widely discussed. An afterthought is that blocks, in lieu of layers, are the elementary parts of residual networks. Each block is a relatively shallow and independent enclosure. Layers in different blocks are largely disentangled from each other by the shortcut. A very deep residual network is actually much "shallower" since the improvement of mapping happens block by block, rather than at each layer. As Veit et al. [104] described, residual networks behave like ensembles of relatively shallow networks. In other words, residual learning introduces a new notion of independent subnetworks into the classic neuron-layer-network hierarchy.

From the simplest AlexNet with 7 layers in total to complex residual networks containing 50, 101 or even 152 layers, CNNs became deeper in terms of the number of layers, as well as hierarchies. In general, deeper networks have better performance, at the cost of more memory

and calculation. They were originally designed for image classification tasks, i.e. class label assignment of the most prominent object in each image. Beyond that, more complicated object recognition tasks like detection and segmentation also benefit from state-of-the-art CNNs when employ them as *backbone* networks for feature extraction.

2.3 Region-based object recognition

One of the greatest successes of deep learning applications in computer vision has occurred in the object recognition field. The backbone networks reviewed in section 2.2.3 have rejuvenated object detection and segmentation algorithms by replacing engineered features with learned representation. Unlike the simple image classification task, object recognition comprehends two interrelated aspects: classification and localization. An image may contain multiple objects in different categories, of different scales and at different locations.

Under some circumstances, classification and localization can be found untangled. For example, given the hypothesis of stationary background, background subtraction algorithms [76] detect all objects deviating from the background model, without knowing their categories. Nonetheless, object classification and localization often benefit from each other. Object category provides information for localization, e.g. color, shape, contour, texture, etc., whilst precise object location makes feature extraction more targeted and efficient for classification. Recent success of the region-based object detection and segmentation algorithms [105, 23, 29, 103, 35, 34, 84, 39] incarnates the superiority of combining the two aspects.

In this section, we briefly review the history of object recognition. The influence of deep learning in this field, from bounding-box-level detection to pixel-level segmentation, is discussed chronologically to show the progress of the state of the art.

2.3.1 Localization through classification

In the history of object detection, data-driven localization was prior to and prerequisite by classification. Segmentation algorithms aiming at separating foreground from background therefore set new trends in the field of detection. These algorithms employ low-level information to obtain perceptually consistent regions [65]. During the perceptual grouping, some generic middle-level constraints may be introduced [55] to cope with the drawback of local features. Segmentation results can either be the finale of object localization [15], or be intermediate (e.g. edge detection [14], oversegmentation or superpixel [3]) and serve

Figure 2.5 An example of selective search process, copied directly from [103]. Regions are proposed via bottom-up perceptual grouping.

as input for further processing. Either way, these *bottom-up* segmentation methods seek a category-irrelevant image partitioning since no high-level information can be referred to.

However, as [103] discussed, there might not be any omni-functional segmentation solution without ambiguity for generic images. Besides, global semantic information sometimes may never be induced from a combination of local features. For example, occlusion can prevent segmentation algorithm from jointing together geometrically separated parts of the same object, especially when the parts differ from each other in terms of color, texture, etc. Hence, ever since cascade of simple features [105] and computationally efficient features like HOG [23] were proved successful, *top-down* localization through classification has become more popular.

In such a point of view, object detection is described as examining every possible pose for object existance, rather than heuristically generating object poses. Yet objects may appear in any position and of any scale, not to mention a huge number of possible aspect ratios due to varied orientation and deformation. Searching literally every composition is impossible. Thence, a more feasible way is sampling the search space. Sliding windows (sometimes with multiple aspect ratios) on a multi-scale pyramid is the most popular solution for exhaustive

search. A pre-trained appearance model is then applied to image regions in each window for classification.

This compromised solution remarkably reduces the number of regions to be examined. Even though, there remains so many locations to visit that this scheme used to be computationally infeasible. Viola and Jones [105] and Dalal and Triggs [23] found two different approaches to tackle this problem.

The first idea is to avoid thoroughly examining every region. [105, 30] showed that simple classifiers are capable of eliminating most of the background regions. By hierarchically applying a series of classifiers from simple to complex, classification result gets refined in each layer. Such a cascade of classifiers ensures computation is well allocated amongst possible locations.

The second approach is sharing computation. The success of HOG [23] lies on not only its representative feature design, but also the feature map shared by all sliding windows. Histograms of overlapping sub-regions (descriptor) are pre-calculated once and for all. Feature of each region is calculated by simply summing up histograms of all the sub-regions it contains. Time consumption is reduced by avoiding duplicated computation.

Despite these improvements, exhaustive search still suffers from inaccuracy of scale and local position, and also lots of unnecessary computation, due to its top-down structure. To deal with it, selective search [103] introduces segmentation (oversegmentation) result into the region-based detection framework. Rather than uniformly or randomly sampled windows, selective search proposes regions of all scales by hierarchically grouping segments. Region proposals are more accurate in terms of overlapping rate, thanks to local features. Besides, less poses are proposed, and the propositions are more targeted compared to exhaustive search, which allows the employment of more powerful classifier.

2.3.2 Regions with CNN features

In the location through classification schema, features are extracted in every region and then sent to a classifier or a set of classifiers. The classification models varied from simplest combination Haar-like feature and threshold classifier [105] to complex bag-of-word feature through kernel SVM [103]. Such classic engineered models seemed to be suboptimal as representation learning techniques like CNNs dominated the image classification field. Girshick et al. [36] showed that under the same deformable part model structure, convolutional features significantly outperform HOG features.

Regions with convolutional features or Region-based CNN (R-CNN) [35] was naturally brought about by introducing CNN-based feature extraction into existing region-based detection algorithm. In its first edition, every region proposal obtained by selective search

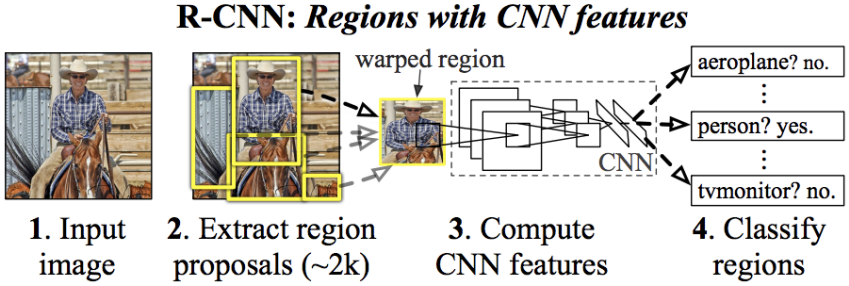


Figure 2.6 R-CNN system overview, copied directly from [35].

[103] are warped into the same size. AlexNet [54] takes the place of bag-of-word to extract features of fixed length from the resized regions. A linear SVM (no need for non-linear kernel SVM) is used as classifier of the convolutional features. For the detected objects, their features can also be exploited by a bounding box regression model [29] for more precise localization. R-CNN achieved far better results than any other algorithms at that time.

The second edition, Fast R-CNN [34], resolved the computational efficiency problem. Since the region proposals are extensively overlapping, feature sharing among them is a good idea borrowed from HOG [23]. One single convolutional feature map is calculated from the input image and features of all regions come from it with simple operation. Regions in original image are geometrically related to the feature map. Their features are obtained using RoI (Region of Interest) pooling, invented in SPPnet [41], on the corresponding areas on the feature map. The regions are no longer resized before input into the network. Instead, a wrapping on feature map level is introduced in RoI pooling: each area is uniformly subdivided into a fixed number of bins. Pooling is conducted on each bin and accordingly results in a fixed-length feature. RoI pooling adopts quantization to cope with misalignment between the discrete granularity of the feature map and the ideal floating-number RoI, as well as its bins.

In Fast R-CNN, the features pass through two fully-connected layers. The output is simultaneously sent to a softmax classifier (instead of SVM) and a bounding box regressor which is now integrated in the network rather than detached from detection phase. Thus, classification and localization are optimized at the same time and benefit from each other during multi-task training. With such modification, Fast R-CNN saw an improvement in terms of time efficiency as well as detection accuracy.

2.3.3 CNN-based location regression

An objective of object recognition is the prediction of object location. It can be interpreted either as region-level detection of object bounding box, or more precisely, as pixel-level

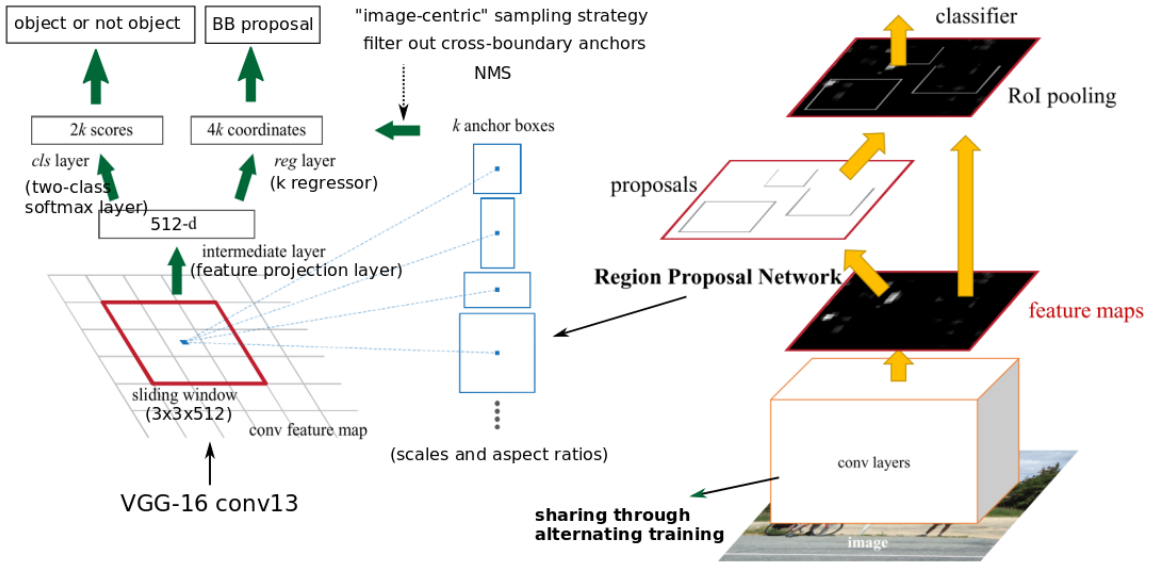


Figure 2.7 Illustration of all components of Faster R-CNN, adapted from [84].

segmentation. The second task is often considered as a classification problem of pixel labeling and will be reviewed later in subsection 2.3.4. Here, we discuss the bounding box prediction problem.

Object bounding box in 2-D digital images is the minimum axis-aligned rectangle that encloses the object. For the facility of notation and calculation, a bounding box is defined by coordinates of the rectangular boundary. As a very simple descriptor of approximate object location, bounding box is widely adopted when pixelwise recognition is not required. For bottom-up recognition algorithms like [103], low-level features are often geometrically sensitive. After perceptual grouping, objects in question have exact locations and their bounding boxes are as simple as the minimum enclosure of the grouped pixels or superpixels. On the contrary, top-down algorithms, especially those based on sliding-window search, always face the problem of bounding box determination. The search space is sampled and therefore, object location and scale are far from being accurate. Besides, high-level features are rarely capable of establishing precise object boundary. The best solution so far is bounding box regression.

In Deformable Parts Models (DPM) [28, 29], the most successful object detection algorithm before deep learning era, Felzenszwalb et al. designed a model specific bounding box predictor using least-squares regression. The detector outputs not only the object category but also a 4-dimensional vector indicating the bounding box. Such location is not precise

enough when derived from high-level holistic detector (called root filter). Deformation of object parts is neglected in the root model. In the approved version [29], models of the parts were also taken as input of the regression, which gave a significant boost on localization accuracy.

The bounding box regression idea was later borrowed by the family of R-CNN algorithms [35, 34]. Instead of calculating directly the bounding box, convolutional feature of the proposed region outputs an offset of the box with respect to the region location. Also, the least-squares loss function was substituted by smooth L1 loss. Convolutional features proved to be efficient in refining coarse regions windows into precise bounding boxes. As mentioned in subsection 2.3.2, feature map is geometrically corresponding to its input image, as its neurons have limited receptive fields. Therefore, the region configuration is critical for the box prediction. Intuitively, when the proposed region is closer to object, which is conventionally measured by the overlapping ratio of two rectangles, the corresponding feature is likely to get a better regression result. The question is, to what extent the CNN-based regression is effective and reliable? Can an arbitrary region correctly predict the object location after training? Some work demonstrated the capability of convolutional features.

Faster R-CNN [84] is an improved version in the R-CNN algorithm family. Ren et al. [84] proposed a region proposal network (RPN) to replace the CPU-implemented heuristic proposal method [103]. RPN exhaustively searches the input image to output potential object regions. RPN shares the same feature map with R-CNN branch, and no feature pyramid is created. Objects of different scales are represented on a mono-scale feature map. Facing similar plight as Fast R-CNN [34], RPN chooses a different approach than RoI pooling. RPN uses a single sliding window on the feature map to examine multiple object poses (*anchors*) at the same time. Anchors are predefined boxes of different scales, aspect ratios and offsets with respect to the center of the sliding window. The feature within the window is sent to a subnetwork (RPN) with classifiers to predict whether the anchors contain any object (positive) or not (negative). For positive anchors, bounding box regression is applied. One thing to emphasize is classification and regression of all the anchors use the same feature, of which the corresponding area on the input image may hardly coincide with some anchors or objects.

From another point of view, RPN is actually a simplified Fast R-CNN. The features as well as the reasoning network of RPN are much less powerful but faster, since its tasks are easier: binary *objectness* classification and relatively inaccurate localization. The positive anchors after regression transformation are presented to the Fast R-CNN in [34] as region proposals. In conclusion, Faster R-CNN concatenates two R-CNNs, of which the simpler purges most of the impossible object configurations to reduce the search space, while the

more powerful refines the results. Hence, algorithms like Faster R-CNN are sometimes referred as *two-stage* detection. Experiments showed that Faster R-CNN provides remarkable detection results while significantly saves overall computational time, as all the modules are realized with GPUs and share the same feature map.

Inasmuch as CNN-based bounding box regression is reliable, several *one-shot* detection algorithms without postclassification stage were proposed recently, e.g. Single Shot MultiBox Detector (SSD) [63] and the family of YOLO [81–83]. Feature maps are uniformly divided, which is a special case of sliding window where the window size equals to the sliding stride. A set of anchors is applied on every feature map part which is responsible to correspondingly predict object category labels and bounding box regression results. As in RPN, the anchors are predefined default boxes of poses, sometimes learned with clustering techniques and category-specific. Unlike YOLO and RPN, SSD examines multiple layers of the convolutional network, i.e. feature maps at different scales. Still, the problem of noncoincidence between feature and the object, alongside with scarcity of searching window, causes lower detection accuracy than two-stage algorithms. The compensation of one-shot fashion is much less time consuming with a nearly real-time performance.

2.3.4 Deep learning in pixel-level segmentation

Beyond *object detection* which denotes object recognition via bounding box, pixel-level *object segmentation* or *instance segmentation* is a finer-grained recognition task. It aims at labeling out all the pixels belonging to each object instance. It requires a special *semantic image segmentation* that differentiates instances. Image segmentation was considered as a perceptual grouping task by bottom-up aggregating low-level features, e.g. minimum cost cut [11] and SLIC superpixel [3]. Under the guidance of high-level semantic feature, heuristic segmentation results are grouped and recognized as object instance [103]. As deep neural networks made a breakthrough in object recognition field, top-down pixel-level segmentation is brought about.

However, deep learning-based recognition algorithms reviewed above are not designed for per-pixel classification. First, most of them work on the very deep feature map of their backbone networks. After layers of convolution and/or pooling, neurons on the final feature map have a very large receptive field. They are not suitable for fine-grained analysis. For example, vanilla Faster R-CNN [84] has an intrinsic problem of ignoring small objects, not to mention per-pixel recognition. Second, techniques like RoI pooling and fully connected layers (MLP) are not pixel-accurate. The coarse spatial quantization and global reasoning can sabotage the pixel-to-pixel alignment. To overcome such problems, several improvements have been proposed.

Fully convolutional network (FCN) [64] substitutes the MLP with convolutional layers to preserve the geometric information. Image classification networks from AlexNet [54] to ResNet [42] require input images resized 227×227 (or 224×224 with convolutional kernel padding). Most networks output a 7×7 feature map at their last layer, followed by a MLP for classification or regression. When serving in region-based object recognition networks as backbones, feature map patches of region proposals become the inputs of the MLP. In order to yield fixed-length features while benefit the pretrained models (e.g. classification models trained on ImageNet [26, 89]), patches are pooled or warped as 7×7. In FCN, the pretrained MLP is reshaped as a 7×7 multilayer convolutional kernel. Instead of applying on patches, FCN convolves the entire output feature map with this kernel. Therefore, FCN comprises only convolutional structures and outputs a heat map of object class labels. After upsampling the heat map with a stride of 32 (as the backbone nets reduces 224 image pixels into 7 neurons), a pixel-to-pixel image segmentation is constructed.

As the segmentation result derives from the final feature map, its scale of detail is very limited after the upsampling with 32-pixel stride. Hence, shallower feature maps are taken into consideration, as they prove to be more robust for small objects and low-level recognition. In FCN, the heat map upscaled by 2 or 4 is combined with feature maps of different strides. High-level information from the heat map controls "what" while low-level information from shallower feature maps provides "where". By fusing information of different granularities, the segmentation detail is significantly improved.

FCN realized top-down semantic segmentation, i.e. every pixel of the input image is labeled with object class labels. The remaining task is separating labels belonging to different object instances. Mask R-CNN [39] accomplished instance segmentation by combining Faster R-CNN [84] with FCN. Besides classification and bounding box regression, a mask prediction branch deriving from FCN is added for the segmentation of object instance. Multitask learning of the additional mask branch also improves the performance of other branches; at the same time, instance-oriented region-based approach allows instance-specific mask prediction. Some seemingly minor changes in Mask R-CNN have large impact. RoI alignment operation takes the place of RoI pooling, which results in a remarkable improvement of mask accuracy. In RoI alignment, a bilinear interpolation [49] eliminates the misalignment introduced by the coarse spatial quantization of RoI pooling. Exact geometric locations and spatial relations of region proposals are therefore faithfully preserved. Another minor modification is class-specific mask prediction. In Mask R-CNN, mask prediction branch outputs a mask for each object category, as done in the bounding box regression branch. The class-specific mask prediction achieves higher accuracy than class-agnostic approach.

FCN provides an idea for recognizing object of various scales as well. For a long time, region-based convolutional approaches suffered from the coarse granularity of deep feature map so much that small objects were barely recognizable. Image pyramid approaches were discarded due to their inefficiency. The research track veered towards the reuse of multiple feature maps of different granularities. SSD [63] reviewed in subsection 2.3.3 explored bounding box regression with feature maps of different scales. Scale Dependent Pooling (SPD) [112] established a correspondence between region proposal scales and feature map granularities. In these algorithms, feature maps are isolate from each other and recognition results were not agreeable enough. Feature Pyramid Network (FPN) [61] borrowed the idea of fusing multi-level information in FCN and created skip connections between feature maps. Recognition is realized at different levels, and even the finest level can benefit from the high-level reasoning results. Experiments showed FPN is beneficial region-based recognition algorithms including Mask R-CNN.

2.3.5 Transfer learning for deep neural networks

Deep neural networks have a lot of parameters to learn, which requires immense training data. Big data is indispensable for the success of deep learning. Sufficient training data can prevent the complex networks from overfitting and increase their generalization ability. ImageNet [26] is the most famous large-scale visual recognition dataset and is involved in nearly every state-of-the-art recognition algorithm. It provides resized images for image classification task. Lots of ingenious networks have been proposed in recent years. From AlexNet [54] to ResNet [42], convolutional approaches even outperform human expert in image classification field, thanks to the large-scale data in ImageNet.

However, big data demands uncountable work of collection, retrieval and labeling. Until today, no other dataset designed for more complex visual recognition tasks is comparable with ImageNet. As a result, *transfer learning* or *knowledge transfer* [73] is an efficient solution for deep neural network training. Confucius once said: one should infer three corners after being shown one. So as machines: adjusting acquired knowledge or learned model to solve unfamiliar problems is appealing since expensive efforts of re-learning can be saved by adequate analogy. Transfer learning describes such analogy as improving a new problem (target) with the help of an old, solved problem (source): between the source and the target, either the domains (data) or the tasks (regression, classification, etc.), or even both, are different.

For example, well-trained parameters of image classification networks, has been used as pretrained model on other datasets, where the patterns found from the huge amount of ImageNet images usually fit well. A strategy called *fine-tuning* aims at reconciling the

inconsistency between the source and the target. Instead of retained the exact knowledge, the learned model itself or part of it is taken into the optimization and progressively aligned to the target domain and problem. Apart from transferring among domains, such knowledge can also be used in other tasks. All the object recognition algorithms reviewed in this section employed image classification networks as their convolutional backbone, along with the pretrained model for parameter initialization. Initialized networks were then trained on object recognition datasets using fine-tuning. PASCAL (standing for Pattern Analysis, Statistical Modeling and Computational Learning, a Network for Excellence funded by European Union) Visual Object Classes (PASCAL VOC) dataset [27] and Microsoft COCO (standing for Common Object in COntext) [62] are the two most adopted datasets for various recognition tasks. Successfully trained models are inherited by newly proposed algorithms so that retraining from ImageNet is unnecessary. In this dissertation, our work also adopts pretrained models with fine-tuning.

2.4 Object re-identification and metric learning

2.4.1 Re-identification algorithms

Object re-identification is another fine-grained recognition task. Unlike all previously reviewed recognition tasks that endeavor to realize inter-category classification, re-identification focuses on intra-category recognition and aims at differentiating instances of different identifications. Unlike generic object recognition with predefined categories, identities within a category are uncountable. Datasets for re-identification often comprise many identities, each of which has much fewer entities. As a result, training and testing domains are barely overlapping and identity-specific models like classification are less effective or even meaningless.

Correspondingly, re-identification is modeled differently. Given an object instance of interest as probe, a re-identification algorithm is demanded to predict whether a query instance has the same identity as the probe. It can be applied for single query classification (pairwise verification), sorting a list of query instances (ranking), or finding instances of the referenced object from an instance "gallery" (instance retrieval). It is encouraged to establish a matching model via instance comparison. The relationship among identities rather than themselves is the desired knowledge to learn.

Object re-identification is fundamental to video surveillance and security. It was first brought up during the study of multi-camera tracking [114, 106]. Due to various camera configurations, the same object observed by different cameras undergoes significant appearance variations. The objective is to establish the correspondence among object instances of

different cameras. Camera calibration information was often included to aid the matching. Later, image-based object re-identification became independent from multi-camera system [32]. No explicit camera information is provided in image-based re-identification tasks, but only the exploitation of visual cues is allowed. In this dissertation, we refer re-identification as purely appearance-based recognition without spatio-temporal reasoning. As more datasets and benchmarks are established [58, 119], image-based re-identification is also widely involved in other visual tasks. For example, in mono-camera video-based tracking problem, disappearance or occlusion of an object may require re-identification to cope with object pose variation.

Although the name of re-identification was brought up not very long ago [114] and usually associated with pedestrian recognition, the problem itself has been studied for a long time. The most famous example is visual face recognition. Besides face localization or detection, identity verification is also an important and meaningful task in security and surveillance. Famous classic algorithms like Fisherfaces [10] and Local Binary Patterns (LBP) histograms [4, 5] were proposed to discriminate faces with different identities. Well-engineered features and descriptors specifically designed for human faces were the key to successful recognition.

Along with the progress made in object detection field, the research domain of generic object re-identification keeps growing. Various descriptors were engineered, from low-level features as color and shape, to middle-level features like texture, then to recent deep convolutional features. For example, various color spaces such as RGB, HSV, Lab and Log-RGB were thoroughly investigated [107] in the context of object re-identification to eliminate the influence of illumination variation. HOG [23] used to be the most popular method of shape extraction, since it is robust to local translation and rotation. Texture filters like Gabor wavelets [24] and local texture descriptors like SIFT [66] and LBP [4, 5], as well as their variations (color SIFT [2], center-symmetric LBP [43], over-complete LBP [9], etc.), contributed a lot to the progress of re-identification and still can provide robust texture features [117, 118]. Recently, deep learning-fashioned features have gradually taken place. They are either embedded in previous learning frameworks with discriminant analysis [53, 57], or trained end-to-end in deep neural networks [56, 102, 12].

Based on various kinds of object features, different genres of knowledge can be learned. Some approaches learn to find an optimal subset of features containing the most discriminant information: linear discriminant analysis (LDA) [10] and other dimension reduction algorithms [91] project the original high dimensional features onto a low dimensional discriminant space, in order to facilitate and accelerate the training of classifiers. Some other approaches learn to directly design robust classifiers like kernel SVMs [80] or discriminant models [57, 53]. For image-based multi-camera re-identification, although no explicit camera

information is provided, it is popular to learn transformation models between cameras. Photometric transformations [51, 79] and spatiotemporal transitions [33] learned from training data can be readily applied on pairs of query images shot by the same cameras. However, in mono-camera video tracking, object appearance variation is often caused by long-term occlusion and therefore more random. Such supplementary knowledge may not be helpful.

2.4.2 Metric learning and Siamese network

As previously reviewed, numerous approaches seek object re-identification solutions from various angles. If we look back into these algorithms, they still have some intuitions in common. Principal components analysis (PCA) and linear discriminant analysis (LDA) achieved remarkable success [95, 10] via applying proper linear transformations on input feature spaces. In other approaches, the linear transformations are incorporated in classification models [57, 53]. As for kernel SVMs, they actually perform transformations defined by kernels, which can be nonlinear, from the feature space to a new one before applying vanilla SVM. In other words, a proper feature mapping is nontrivial and may be the key to re-identification tasks.

This conclusion leads to the research of metric learning in object re-identification. Instead of comparing two instances directly on their original feature vector space, they are explicitly transformed onto a target vector space where a metric is defined to indicate the degree of two instances belonging to the same object. Object re-identification via metric learning consists of two interrelated parts: choosing a proper metric on the target vector space, and learning an optimal mapping that yielding an embedding suitable for the metric.

Mathematically, given two instances and their feature vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^m$ on the input space, we refer them as a positive pair when they have the same identity and a negative one on the opposite. In metric learning, the probability of having a positive pair is related to the distance, which is defined by a metric function $d : \mathbb{R}^n \times \mathbb{R}^n \mapsto [0, +\infty)$, between their transformed feature vectors on the embedding space $\mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^n$:

$$d(\mathbf{y}_1, \mathbf{y}_2) = d(f(\mathbf{x}_1), f(\mathbf{x}_2)) \tag{2.5}$$

where $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ is the mapping function to embedding space.

Re-identification problem is naturally formulated as minimization of the loss between the distance d and its desired value d^* in the ground truth. Since the metric and its loss function is defined before optimization, metric learning reduces to train an optimal mapping function f^*:

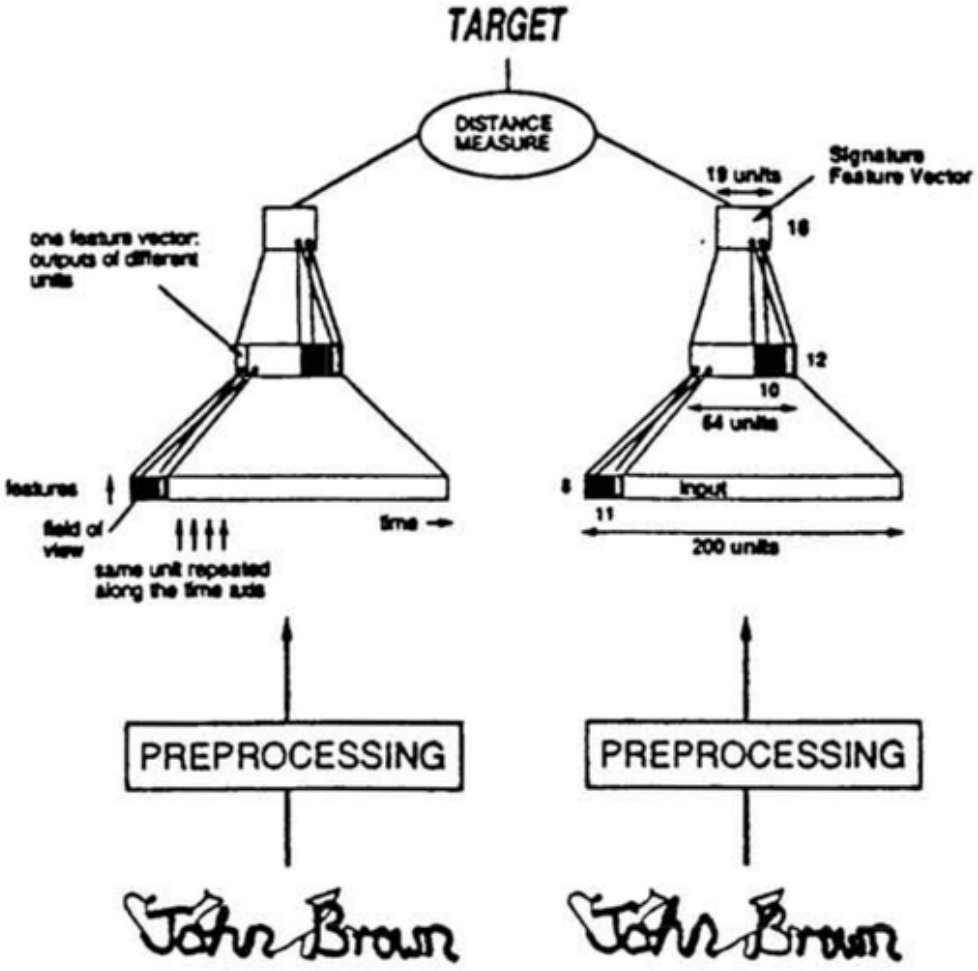


Figure 2.8 Original Siamese network architecture, copied directly from [12].

$$f^* = \arg\min_f L(d, d^*) \tag{2.6}$$

State-of-the-art metric learning algorithms choose to learn a nonlinear and powerful mapping function. Siamese network [12] which has two identical branches was designed as a solution of instance comparison. The instance pair are input respectively into each branch, where the same mapping is applied on both entries, thanks to the parameter sharing between the two branches. In its initial realization, the transformed features output by the Siamese structure are compared with each other under Euclidean distance. A contrastive loss [12] between their distance and the ground-truth label is calculated and backpropagated, to

optimize the mapping for better reasoning. In recent work, more metrics [72, 118] are utilized and bounded losses [20, 38, 90] are preferred. Very deep convolutional backbone structures pretrained on ImageNet or other object recognition datasets are also introduced into Siamese structure. In such deep Siamese networks, feature representation and its reasoning is learned end-to-end. From the point of view of deep learning, Siamese networks stand for an entire metric learning that maps the input image rather than extracted features onto a target vector space. Choices of object descriptors, feature dimension reduction and classifier design are all integrated and realized within a single deep neural network.

There are also some variations of deep Siamese networks. "Stacked" networks [56, 101, 102] were proposed to encourage early-stage information exchange between the two branches. The input images are "stacked" together, at either the input level by expanding the RGB channel, or intermediate levels by concatenating their feature vectors. Auxiliary information is often stacked together: associated optical flow maps [56] or body part detectors [101, 102] serve as an alignment guidance and contribute a lot to pairwise comparison. The earlier the "stack" happens, the better its performance is [56]. However, such a connection is asymmetric within the pair and the two inputs are not identically transformed. A more serious problem is, jointly using information of both observations at early stage means more pair-specific computation, which cannot be shared by other pairs. Every layer after "stack" operation needs to be recalculated. In multiple object tracking, long sequence and lots of observations can make the calculation immense and very time-consuming, even with the help of powerful GPUs.

It should be remarked that although the metric is a distance function of $\mathbf{y}_1$ and $\mathbf{y}_2$, the original prediction between $\mathbf{x}_1$ and $\mathbf{x}_2$ is not necessarily a distance: a non-injective mapping may easily violate the identity of indiscernibles. Specifically, when the metric is Euclidean distance and the mapping is linear, the prediction function itself becomes Mahalanobis distance [108] on the input space. On the other hand, similarity functions defined as below

$$s(\mathbf{y}_1, \mathbf{y}_2) = \frac{\mathbf{y}_1^T \mathbf{y}_2}{N(\mathbf{y}_1, \mathbf{y}_2)} \tag{2.7}$$

are also considered as pseudo-metrics and adopted in metric learning (called similarity metric learning). $N : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}\backslash\{0\}$ is a normalization function. Specifically, when $N(\mathbf{y}_1, \mathbf{y}_2) = 1$, the function s is called bilinear similarity [16]; when $N(\mathbf{y}_1, \mathbf{y}_2) = \|\mathbf{y}_1\| \|\mathbf{y}_2\|$, it is called cosine similarity [72].

A more thorough research on metrics, loss functions and deep architectures employed in Siamese networks will be presented in Chapter 3 with experimental results.

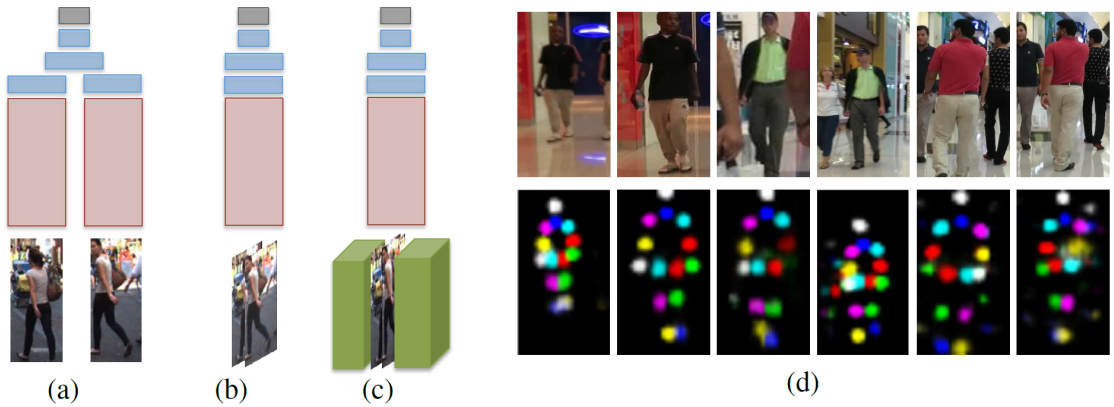


Figure 2.9 “Stacked” network for re-identification in tracking used in [102], copied directly from [102]. (a-c) Architectures of stacked networks. Red rectangles indicate the convolutional backbone of VGG16 [94] (convolutional, relu and pooling layers). Blue rectangles indicate the reasoning head consisting of fully-connected layers. Grey rectangles on the top of each network are the loss layers, which often employs verification/binary classification losses. Green boxes are the stacked body part score maps. (a) Unstrict Siamese network with early stage information exchange: feature vectors of two branches are concatenated together rather than directly compared to each other. (b) Stacked network: input images are stacked along the channel dimension. (c) Stacked network stacking human body part detection maps shown in (d) with input image.

2.5 Multiple object tracking in mono-camera videos

The principal objective of this thesis is to propose a robust multiple object tracking algorithm in mono-camera videos. Multiple object tracking is a more challenging task, compared to single target tracking. Objects need to be not only detected from cluttered background, but also differentiated from each other. Correspondingly, the tracking-by-detection framework is widely adopted in multiple object tracking. Interclass detection is taken care of outside the tracking phase. In this way, tracking reduces to data association, i.e. the process of linking detection hypotheses into full and disjoint trajectories without branch or circle [77, 115, 92]. Meanwhile, intraclass differentiation of pre-detected objects in all frames becomes the only objective of tracking models.

In this section, we review some multiple object tracking approaches along with their appearance models. We first explain some different graph models used for data association in tracking and then discuss the optimization strategies based on the graphs.

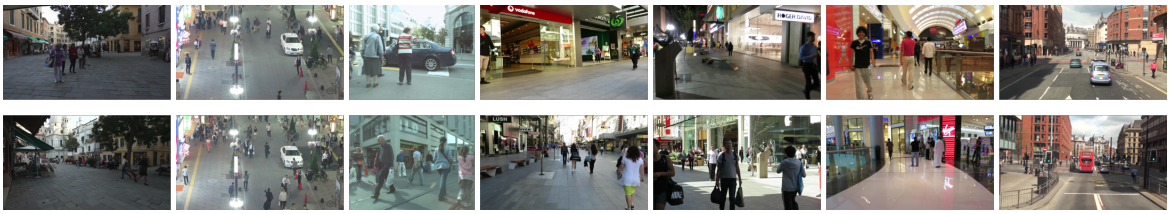


Figure 2.10 An overview of the Multiple Object Tracking Challenge benchmark, images chosen from the sequences in MOT16 [67]. Top: training sequences. Bottom: testing sequences. All sequences are extracted from continuously captured mono-camera videos.

2.5.1 Tracking by detection with motion and appearance models

Tracking-by-detection framework has recently seen heavy use for MOT tasks. Network flow-based methods [116, 13, 25], continuous energy minimization [68], multiple hypothesis tracking (MHT) [53], and minimum cost multi-cut (JMC) [101] or lifted multi-cut (LMP) [102] are some trending approaches using this framework. Such approaches formulate tracking as the correspondence among detected objects through time. Objects are often detected independently in every frame before associated with each other across frames. The detection results (also called *observations*) are linked into tracklets or trajectories.

During observation association, various kinds of information can be exploited. The most obvious solution is using spatiotemporal information. In most cases, object and camera movements are smooth and continuous, producing predictable trajectories through successive frames. The poses of an object in consecutive frames are therefore pertinent. Their patterns can be induced from object observations within a short period, with pre-designed motion models. The simplest motion model is vicinity gating, i.e. excluding all the links between distant observations. Some more sophisticated ones from linear quadratic estimation with Kalman filter [53] to spatiotemporal relation metric [102] have been proposed to cope with complex trajectories.

Such models are less computationally demanding than appearance-based ones, but the motion assumptions that they rely on are often invalid. Abrupt camera and/or object motion and complex occlusions occur frequently in multiple object tracking tasks. Object occlusions and temporary disappearances from the scenes are prevalent in multiple object tracking tasks, where motion models are incapable of bridging long-term and long-distance gaps. Besides, nonrigid objects may undergo irregular deformation that violates the smooth motion hypothesis. The motion models alone are not robust enough. Appearance-based association is accordingly brought up to overcome such problems and enhance the tracking performance. A remarkable fact is that nowadays, most appearance models employ deep learning features.

Appearance models, especially the deep learning-based ones, are playing a more important role in data association.

Kim et al. [53] revisited MHT with CNN features. Each observation is represented by a feature vector extracted with a pretrained deep convolutional network. Every potential object has its own linear regressor which indicates whether an observation belongs to it (positive) or not (negative). Each regressor is a linear projection from feature vector space to a scalar response, independent of each other. An ideal linear regressor responds to positive observations with 1, and to negative ones with -1.

The training of linear regressors is online: observations in a new frame not only are examined by regressors of all track hypotheses but also serve as their training data. Each hypothetical object incorporates a possible observation (sometimes none) into its positive instances while disperses the rest. Its regressor will then be adjusted to minimize the square error of its responses to all observations after the update.

This online strategy results in robust discriminative appearance model. After a short sequence of training, the online model is usually steady enough against drifting. Also, CNN features prove to be powerful for fine-grained inner-category recognition. The multiclass recognition problem is factorized into several 2-class regression/classification tasks, and hence, low-dimensional (e.g. 256-D) feature vector space can work well. There are also some drawbacks. The convolutional network for feature extraction is trained apart from the data association task. Therefore, the feature vector space is not optimized for the inner-category linear regressors which may also be improved by introducing nonlinearity.

Tang et al. [102] employed a deep stacked network shown in Fig. 2.9 as appearance model. By fusing human body part information into the convolutional backbone of deep CNNs, the appearance model is much more powerful when well-trained on object re-identification datasets. Please refer to subsection 2.4.2 and 3.4.1 for more discussion on this appearance model, or to [102] for implementation details.

These appearance models prove to be more robust than motion models in multiple object tracking tasks. However, appearance models are far from perfect. In practice, both kinds of models are combined: motion models help reduce the search space for data association and boost the object re-identification while accurate association results provided by appearance models render the motion prediction more precise. The interference and collaboration between models will be further discussed and investigated in Chapter 4.

2.5.2 Formulation of data association

In state-of-the-art tracking algorithms, data association is usually modeled as an optimization problem on a graph where every detected object is a node. Each node is assigned

with a vertex weight w_v, which usually related to its objectness credibility obtained during standalone detection. Any two different nodes are connected by an edge and no self-loop is allowed. Edges are properly weighted (denoted by w_e) according to the affinities between their vertices in terms of geometry or appearance, or both. Probabilistic models are often employed for weighing edges given the outputs of the previously reviewed tracking models.

Usually, nodes are chronologically linked through frames in these algorithms, resulting in a directed graph. Tracking is then modeled as finding minimum-cost/maximum-weight disjoint paths [116, 77, 92, 53] on the graph. For any node in the graph, at most one outgoing and one incoming edge are permitted remain in the final result. The optimization is a maximum-weight independent set (MWIS) problem, which is NP-hard. MWIS is closely related to the maximum clique problem. When a graph is sparse, the maximum clique can be found in linear time [19]. Therefore, multiple object tracking algorithms like [53] seek optimal disjoint paths solution using clique techniques by reducing the graph density as hard as possible. In this process, a discriminant appearance model that eliminates most impossible edges can be very helpful.

Ideally, an object can be observed in detection results of a frame for no more than once and data association should be dealing with a multipartite graph. Unfortunately, imperfect detections often leave inaccurate observations that cannot be eliminated by non-maximum suppression. To address this issue, Tang et al. [100, 101, 6, 102] modeled data association as a minimum cost multicut problem. Negative edges are removed until the graph is divided into subgraphs, and the graph decomposition is optimized with linear program. Each subgraph is obliged to be complete within itself (referred to as circle constraint in [100]), so that only duplicated detections are absorbed into the clusters whereas inconsistent noises are excluded.

The multicut approaches prove to be more efficient and accurate. Detection duplications are eliminated thanks to a forced absorption within each frame. Compared to direct disjoint paths searching, intra-identity edges are kept and fewer edge eliminations are required. Fewer competitions among duplicated observations in the same frame incur fewer extra tracks (which will be considered as false positives) and ID switches. Such a strategy can be borrowed by disjoint paths approaches. It can be realized either with a finer objectness classifier that filters inaccurate detections, or via re-identification-based non-maximum suppression.

2.6 Conclusion and insights

In this chapter, we reviewed the *status quo* of the multiple object tracking field, which is currently dominated by tracking-by-detection regime. Given an image sequences extracted

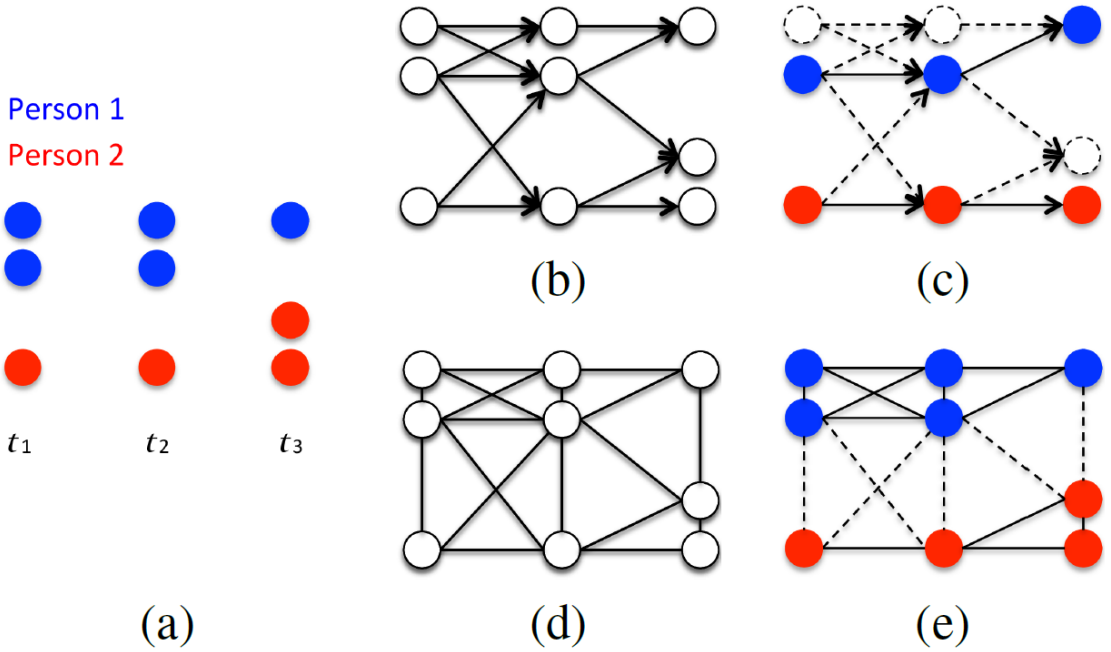


Figure 2.11 Comparison of two different data association formulation with graph models, copied directly from [100]. (a) Ground truth of detections in each frame. (b) Directed graph model for disjoint paths searching approaches. (c) Minimum-cost/maximum-weight disjoint paths found with respect to the graph in (b). (d) Undirected graph for multicut formulation. (e) Minimum-cost graph decomposition of the graph in (d).

from a mono-camera video, objects are first detected independently from every frame, and then linked together over time into disjoint trajectories. In both phases, robust object recognition is paramount to successful tracking.

Object detection has been thoroughly studied for decades. From the era of hand-engineered features to deep learning, this domain has witnessed a remarkable progress in terms of both accuracy and speed. Although the research on object detection itself is beyond the scope of this thesis, it is highly involved and prerequisite in the multiple object tracking tasks. Trending MOT benchmarks provide public detection results for data association evaluation, thanks to the state-of-the-art detection algorithms. Besides, the deep learning architectures and strategies used in detection research can be borrowed in tracking jobs. In the next chapter, we will investigate and compare the influence of such architectures and strategies when they are applied in the data association phase of tracking.

As for data association, we went through the history of few fine-grained intra-category recognition and reviewed its state of the art. Specifically, we investigated a few algorithms, leading by metric learning, that can serve as appearance-based models in the detection

association phase. However, none of the existing data association algorithms is near perfect. On the one hand, per-identity recognition remains to be improved, especially when dealing with imperfect detection results or adjacent similar-looking objects. On the other hand, the association between the established tracks and objects remaining to be identified is flawedly interpreted in most tracking algorithms.

Hence, we come up with some insights on proposing a robust, effective and efficient multiple object tracking algorithm. Obviously, a powerful appearance model can significantly improve the tracking performance. Accordingly, we aim at proposing a discriminant object re-identification model. Metric learning is a promising tool towards this goal. We prove in the next chapter that the classic Siamese structure can achieve remarkable re-identification accuracy with the guidance of per-identity classification, by means of small modification on the loss, which brings few overheads. The re-identification model can later be integrated in tracking as appearance model for data association. Analogous to existing tracking approaches, a well-designed probabilistic model is necessary for the integration, and identity classification rather than verification appears to be more efficient. The flaw in the previous association models is corrected. Additionally, research of the interaction between appearance and motion models is interesting and profitable.

Chapter 3

Object Re-identification with Similarity Metric Learning

3.1 Introduction

As mentioned in the previous chapter, object tracking requires robust object appearance models as visual guidance to overcome challenges like long-term and long-distance occlusions. In multiple object tracking tasks, there are always abundant objects of the same category (e.g. pedestrians) in the scene. The appearance models are therefore demanded to be not only representative but also discriminant to intraclass object instances with different identities.

Among all the appearance models, those based on object re-identification stand out recently, as the tracking-by-detection regime dominates multiple object tracking field. With per-category classified objects and their coarse locations provided by detection algorithms, the appearance models for data association can focus only on fine-grained intraclass discrimination, for which re-identification is designed. Since the objective of this thesis is the application of re-identification in mono-camera video tracking, contextual methods using calibration and transformation information between cameras are excluded from the discussion.

After reviewing and analyzing state-of-the-art re-identification strategies, we come up with a similarity metric learning framework suitable for multiple object tracking tasks. It is realized with a Siamese network that learns similarity metrics. In the next section, we give the network structure and implementation details of our proposed re-identification algorithm. Choices of feature extraction backbone networks, reasoning networks and training data mining approaches are discussed. We also introduce a classification loss to guide the metric

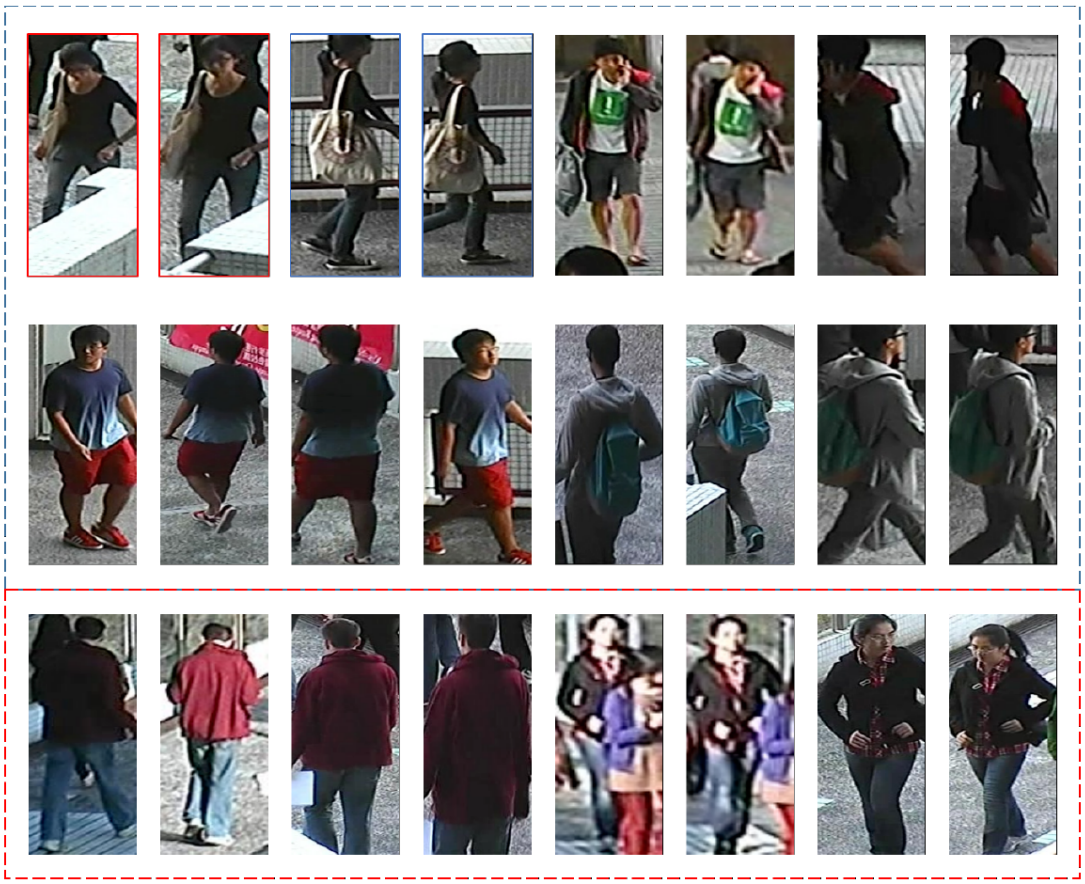


Figure 3.1 Samples from the re-identification benchmark CUHK03 [58], copied directly from [58]. The two adjacent samples have the same identity.

learning. In opposite to the existing algorithms with separated classification branches, our classification model is directly constructed on metric learning results. Then, we compare the proposed framework with state-of-the-art strategies to explain the insights beneath our design. At last, some experimental results are listed to demonstrate the learned re-identification embeddings under different design choices.

Conventionally, we refer a pair of instances possessing the same identity as a positive pair, or a negative pair otherwise. The training and inference of deep networks are realized under the deep learning framework Caffe [52] and with GPU acceleration. To compute the distances or similarities of all the feature vector pairs, we designed a binary operation for feature vectors with the MATLAB "singleton expansion" style, which will be detailed in Appendix A). Necessary functional layers are added.

3.2 Deep Siamese network for metric learning

Siamese network was proposed in 1993 [12] for pairwise signature verification, which is a re-identification task. The network uses a mirror structure with two branches sharing the same parameter set. The two subnetwork branches take an input each, and output their features with the same extraction process. The two feature vectors are compared under a chosen metric (distance or similarity) function. A loss deriving from the error between the comparison result and the desired distance or similarity is used for training the network. The name "Siamese" was taken from the "Siamese twins" to express the conjoined twin network structure [1].

As discussed before in Section 2.4, it is preferable to acquire a comparison model via metric learning in re-identification tasks rather than an identity-specific representative model. Identities in training sets can rarely be observed in testing, and each identity has very few instances for training. Hence, we choose to learn an embedding where each identity is mapped to a neighborhood and the clusters are distant from each other.

Recently, Siamese networks and its idea of learning an embedding have been widely applied in object re-identification field. Many loss functions have been proposed for various metrics. For distance metrics (Euclidean distance for most of the cases), contrastive loss [20, 38] and triplet loss [90] are two most adopted loss functions. For similarity metrics, a very recent work investigated the combination of cosine similarity metric learning with softmax classifier [111]. In this section, we propose a simple Siamese structure for similarity metric learning. Since the final objective is multiple objective tracking, we also report some corresponding designs.

3.2.1 Siamese structure

We adopt the same Siamese structure as in [12] and use modern deep neural networks for feature extraction. As shown in Fig 3.2, the Siamese network takes resized images containing coarsely aligned objects as input. Images are directly projected onto a compact embedding by the same nonlinear mapping realized with a pair of parameter-sharing deep subnetworks. As mentioned before in subsection 2.4.2, such deep architectures learn an end-to-end metric from original images to the embedding without explicitly separating feature extraction and vector space mapping.

The deep architecture is the core of metric learning. *Backbone* networks are critical to all kinds of object recognition. Many re-identification algorithms design their own simplified bespoke networks with less parameters and easier to train than ImageNet models. On the

1. https://en.wikipedia.org/wiki/Chang_and_Eng_Bunker

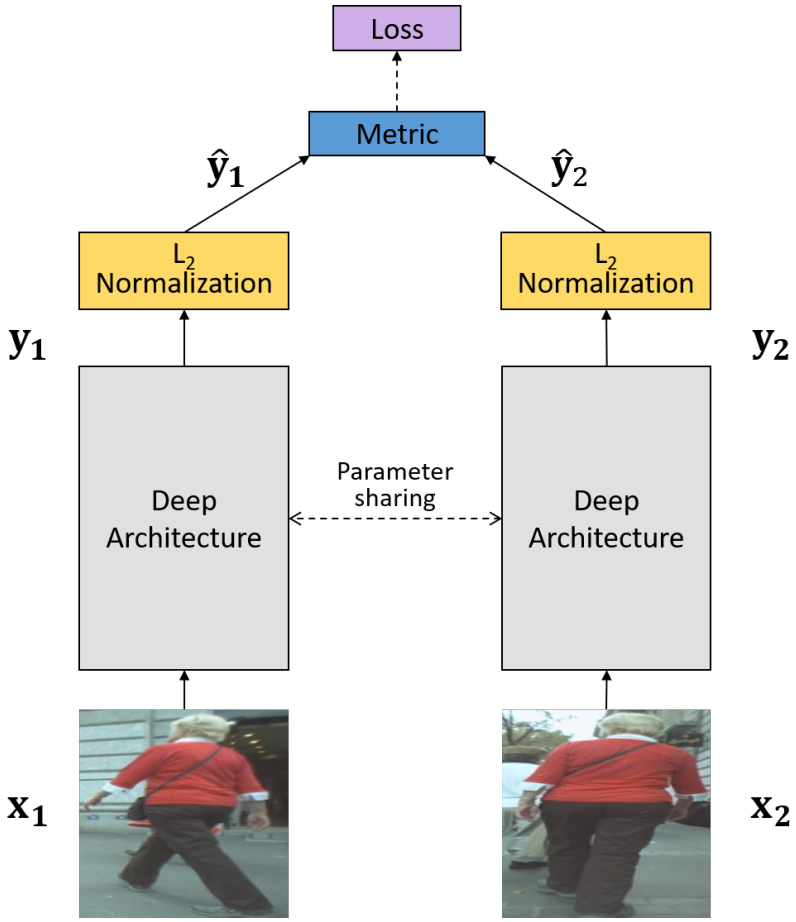


Figure 3.2 Illustration of Siamese network.

contrary, we investigate in this thesis two following state-of-the-art convolutional backbones: 16-layer VGG net (VGG16) [94] and 50-layer residual net (ResNet50) [42]. The idea is to further integrate re-identification into generic object recognition framework like Mask R-CNN [39].

Aside from backbone nets, Ren et al. [85] argued global reasoning *head* networks are as important. We believe a well-designed deep reasoning network can also help to learn the embedding. Residual networks employ a simple average pooling layer of large (global) kernel. Multi-layer perceptrons (MLPs) prove to be effective [94, 85] but incur severe memory demands caused by a huge amount of parameters. The last feature maps of backbone networks still contain too many neurons ($7 \times 7 \times 512$ in VGG16 and $7 \times 7 \times 2048$ in ResNet50) for fully connected reasoning. Such dense layers are difficult to train and prone to overfit, and consequently, a large training data set and other techniques like Dropout [46] are necessary. In our experiments, embeddings yielded by such direct dense mappings are not satisfactory enough.

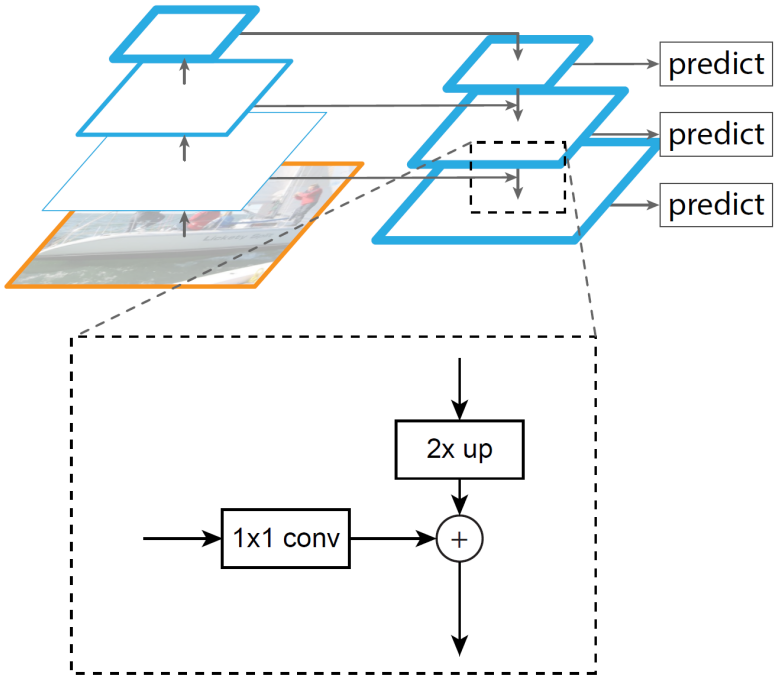


Figure 3.3 Feature pyramid network, copied directly from [61]. Feature maps of different scales from the backbone net are followed by the same simple convolutional head net. Their outputs are top-down combined together with a lateral connection illustrated in the dotted box.

An alternative solution is using another dimension reduction (from 512 or 2048 to 256) convolutional network with small kernels (1×1 or 3×3) on the backbone feature maps [61, 39, 85]. Besides, the appended convolutional network imposes a reasoning explicitly preserving local spatial information. In our work, we choose a convolutional head containing at least two convolutional layers. The first uses a 256-channel 1×1 kernel for dimension reduction, and the second conduct 3×3 convolution. This head network design is identical to the process on the coarsest resolution feature map (denoted as FPN $P5$) in feature pyramid structure (FPN) [61]. Experiments show a significant improvement over MLPs and other head networks, in terms of both effectiveness and efficiency.

We also report a further research on the influence of feature maps of different granularities. Until 2017, most canonical object recognition networks operate on a final feature map with a stride of 32. A 224×224 input image is reduced to a 7×7 feature map, which is conventional from the first deep neural network [54]. Very recent work prefers less coarse feature maps, among which the one of the second last scale (stride 16) is the most exploited: in VGG16, it is the feature map (conv5_3) before the fifth pooling layer (pool5); in ResNet50, it is the output of the fourth convolutional stage (res4f). Object detection accuracy is reported to

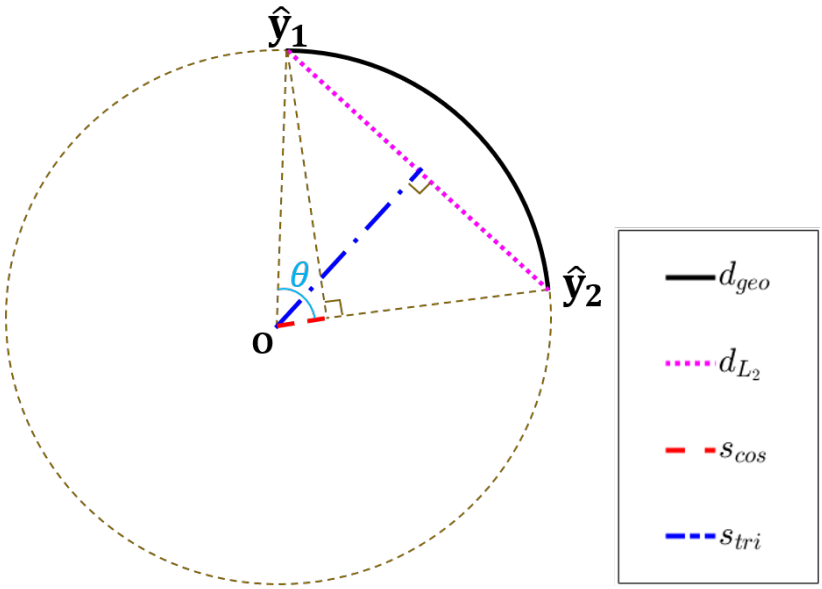


Figure 3.4 Illustration of four directional metrics between two unit vectors $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$: geodesic distance (this work), L_2 (Euclidean) distance [20, 38, 90], cosine similarity [72] and triangular similarity [118]. They are depicted by the corresponding line or curve segments within a great circle of the unit hypersphere, of which $\mathbf{o}$ is the center. All the metrics are directly determined by the angle between the two vectors θ.

gain from using shallower feature maps [42, 85]. Since re-identification is a finer-grained recognition task, the embeddings mapped from short-stride layers are supposed to be more accurate. We experiment on two different scales (with strides 32 and 16), along with the fusion of both feature maps. The latter is realized in the feature pyramid fashion [61], i.e. the FPN $P4$ map. Thanks to the two-layer head deliberately borrowed from FPN, an ablation comparison of different feature maps is possible.

3.2.2 Metrics

In our Siamese framework, outputs of the deep architectures are followed by L_2 normalization, which produces an embedding on a unit hypersphere, as done in some previous works [72, 90]. Within the compact feature vector space, most works choose Euclidean distance as metric [20, 38, 90], features normalized or not. Alongside the existing metrics, we also investigate the great-circle distance, a metric subject to the embedding manifold, i.e. the unit hypersphere.

The great-circle distance is the length of the geodesic (the shorter part) between two unit vectors $\hat{\mathbf{y}}_1 = \frac{\mathbf{y}_1}{\|\mathbf{y}_1\|}$ and $\hat{\mathbf{y}}_2 = \frac{\mathbf{y}_2}{\|\mathbf{y}_2\|}$. It is the most intuitive distance along the hypersphere

surface (or the embedding manifold). By its definition, the geodesic distance $d_{geo} \in [0, \pi]$ can be calculated as below:

$$d_{geo} = \arccos(\hat{\mathbf{y}}_1^T \hat{\mathbf{y}}_2) \tag{3.1}$$

It is directly related to other directional metrics, e.g. cosine similarity $s_{cos} \in [-1, 1]$ in [72], triangular similarity $s_{tri} \in [0, 1]$ in [118] and the Euclidean distance between normalized vectors $d_{L_2} \in [0, 2]$ in [90],

$$\begin{cases} s_{cos} = \hat{\mathbf{y}}_1^T \hat{\mathbf{y}}_2 \\ s_{tri} = \frac{1}{2} \|\hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2\| \\ d_{L_2} = \|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_2\| \end{cases} \tag{3.2}$$

Their relationship is illustrated in Fig 3.4. The black bold boundary is the great circle of the unit hypersphere that passes the two points indicating the feature vectors $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$. When $\hat{\mathbf{y}}_1 \neq \hat{\mathbf{y}}_2$, the great circle is unique. The geodesic distance between the two vectors is the length of the arc lying between the two points on the circle. Euclidean or L_2 distance is the chord length. They are both tangential while similarities measure radial lengths. Triangular similarity equals to the distance from the hypersphere center to the chord (or the difference between hypersphere radius and the circular segment sagitta). Cosine similarity measures the projection length of the radius passing one point on the other.

These metrics are not directly comparable to each other. On the one hand, distances are supposed to be the opposite of similarities, by their definitions. The minimization of distances implies the maximization of similarities. On the other hand, their ranges are not the same. However, they can be normalized onto a polar coordinate system with simple operations like scaling (including flipping) and translation. An example of such normalization is shown in Fig. 3.5, where their ranges are all $[0, 1]$ and follows the tangential measurement (0 indicating identical vectors as for distances). The transformation for each metric is shown in the corresponding legend in the figure.

3.2.3 Pairwise loss functions

There is no specific loss function for similarity metric learning. In cosine similarity [72] and triangular similarity [118] metric learning, their loss functions are designed to draw every positive instance pair together ($s_{cos} = 1$ or $s_{tri} = 1$, respectively), and separate every negative pair as far as possible ($s_{cos} = -1$ or $s_{tri} = 0$, respectively). In this subsection, we seek to absorb these functions into the explicit contrastive losses and provide a uniform

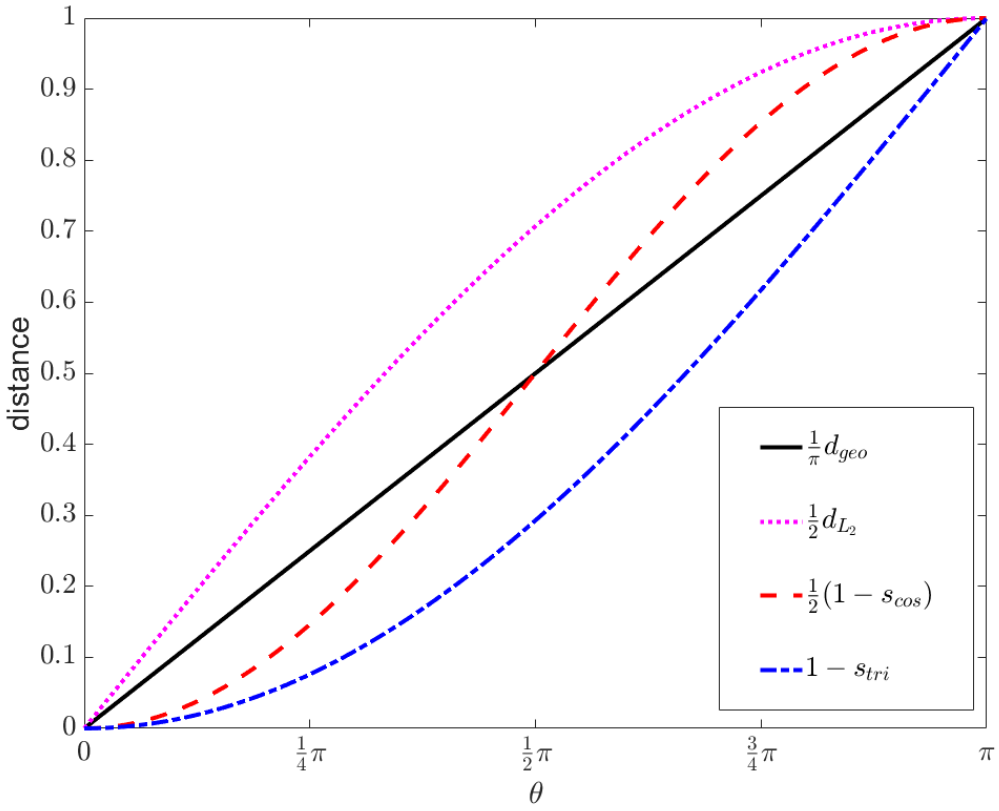


Figure 3.5 Comparison of normalized distances. After scaling and translation, all metrics have a range of $[0, 1]$ with 0 indicating identical points on the unit hypersphere and 1 for the opposite spectrum. They are compared under a polar coordinate system where the horizontal axis depicts the included angle in radian of the two vectors and the vertical axis for normalized value.

loss function. All these loss functions including the contrastive loss [38] are designed in a pairwise fashion. Recently, triplet losses [90] are becoming more adopted for their better re-identification performance.

One of the most important modifications of triplet loss over pairwise losses (Euclidean distance-based or directional similarity-based) is the loose constraint on intraclass distribution. Pairwise losses are designed to project instances of the same identity onto a single point on the embedding, while enlarging the distance between different identities. Triplet loss discards the first objective and only focuses on creating a wide enough interclass moat. This allows a relatively arbitrary distribution within every class that may differ from each other, as long as there is a clear boundary between intraclass and interclass distances.

The design of triplet loss is beneficial to clustering. The learned embedding is good at ranking and instance retrieval. In multiple object tracking, however, we prefer a tight

intraclass distribution so that the appearance-based classification has an omni-functional threshold. Also, training with triplet losses is much more complex. Therefore, we stick to the pairwise structure. Let P be the set containing all instance pairs (or the entire batch during batch optimization). We inherit the contrastive loss defined in [38]

$$L_{pair} = \frac{1}{2|P|} \sum_{(\mathbf{y}_1,\mathbf{y}_2)\in P} \chi[d(\mathbf{y}_1,\mathbf{y}_2)]^2 + (1-\chi)[\max(\mu - d(\mathbf{y}_1,\mathbf{y}_2),0)]^2 \tag{3.3}$$

where $\mu > 0$ is the desired margin between different clusters and $|P|$ is the cardinality of the set P. Originally, the distance d is the Euclidean distance and χ denotes the ground truth. Specifically, χ is a pair indicator

$$\chi = \begin{cases} 1, & \text{if } (\mathbf{y}_1,\mathbf{y}_2) \text{ is positive} \\ 0, & \text{if } (\mathbf{y}_1,\mathbf{y}_2) \text{ is negative} \end{cases} \tag{3.4}$$

Based on the contrastive loss defined in Eq. (3.3), we come up with a directional metric version for unit vectors after L_2 normalization:

$$\hat{L}_{pair} = \frac{1}{2|P|} \sum_{P} \chi\hat{d} + (1-\chi)\max(\mu - \hat{d},0)^2 \tag{3.5}$$

where $\hat{d} \in [0,1]$ stands for any normalized directional metric. Enumeratively, we have

$$\begin{cases} \hat{d}_{geo} = \frac{1}{\pi} d_{geo} \\ \hat{d}_{L_2} = \frac{1}{2} d_{L_2} \\ \hat{d}_{cos} = \frac{1}{2}(1 - s_{cos}) \\ \hat{d}_{tri} = 1 - s_{tri} \end{cases} \tag{3.6}$$

With the definition of such unified loss functions, we can train to learn an embedding with different metrics under the same framework and readily compare them with each other. Given the range of normalized metrics, the margin μ is limited to $(0,1]$ and specifically, when $\mu = 1$, the loss $\hat{L}_{pair}$ in Eq. (3.5) is equivalent to the original loss function in similarity metric learning [72, 118] to some extent. Recent successful metric learning designs suggest a bounded loss of negative pairs. According to our experiments, the bounded similarity margin $\mu < 1$ indeed works better. The probable explanation is that an unbounded loss separates negative pairs as far as possible until the extreme, which means the exact opposition on the hypersphere with the included angle of any negative pair $\theta = \pi$. For a certain instance, all its rivals are pushed towards the same point, although they have different identities and should

be discriminated among themselves. Oscillations occur as clusters compete for their proper neighborhoods before an equilibrium, which can be avoided with a small margin.

3.2.4 Training data preparation

The Siamese network needs pairs of images containing object instances for training. Both positive and negative pairs with a proper relative ratio are necessary to ensure tight distribution within class and large interclass margin at the same time. The training datasets for re-identification tasks usually consist of hundreds or thousands of identities and nearly ten thousand instances in total. Combination of instances results in a set of pairs that is intractably large. To learn a metric with respect to the pair set, stochastic optimization approaches like stochastic gradient descent (SGD) are indispensable. Selection of training pairs from the immense set is therefore important and tricky.

An intuitive strategy is to randomly sample the training set. For convenience, the set is often divided into positive and negative pair subsets [93]. Instead, we use a two-shot random sampling: identifications in the batch are first randomly selected (duplicated identification allowed) and for each chosen identification, several instances are then randomly picked. Each instance passes through the deep architecture only once and is represented by a fixed-length unit vector on the target feature space. Pairs are generated by examining all the combination of different instances. Each pair is assigned a ground-truth label by contrasting the identities of its members, along with a distance or similarity calculated with their feature vectors. With this random selection, we ensure that for any selected instance, the mini-batch contains both positive and negative pairs of it. The pair selection result resembles triplets, but with a explicit constraint on positive pairs.

In the meantime, hard pair mining [93] is necessary. We use an online hard positive and negative mining. For each iteration, a selector is applied on the distances during feedforward phase. Only the hard pairs are allowed to backpropagate their losses. Beyond that, we maintain a probability list for all identities. Once an instance is found in a hard positive or negative pair, its corresponding identities gets an elevation in the probability of being chosen.

3.3 Classification-guided similarity metric learning

3.3.1 Motivation

Pairwise and triplet losses try to straightforwardly manipulate the distances and impose a boundary between identity clusters. Each cluster pulls its members together while pushes others away until un equilibrium among all clusters. Centers and distributions of clusters

are fixed through the competition among identities. Such a strategy has been criticized for ignorance of neighborhood context [86]. Due to the tiny size of mini-batches compared to the entire pair pool, inconsistency between successive iterations may hinder the convergence as well as performance.

Personally, we encountered a dilemma regarding this drawback during our research. Starting from the models pre-trained on ImageNet for inter-category recognition, our Siamese network requires a high learning rate to break the local optimum state when transferring to re-identification tasks. However, direct metric learning strategy suffers from oscillations caused by the high learning rate. The network spent a long time before convergence or could even not converge in our experiments. Re-identification performance of learned metric was not satisfactory when applied for object tracking.

Apart from direct metric learning above, learning with a surrogate loss has been recently argued to be a better solution for re-identification [44]. Given that re-identification is often modeled as pairwise verification, identity classification or ranking, several algorithms choose to use task-related losses or their combinations [56, 102, 17, 31, 59, 121].

However, metric learning is partially or even totally discarded by the algorithms mentioned above. Specifically, pairwise verification is not directly based on the distance or similarity between feature vectors on the embedding space. Instead, the branches are fused together long before the verification loss is calculated. Either an additional deep verification subnet is appended after the fusion of intermediate outputs from the learned embedding [31], or metric learning is entirely replaced by pair-specific binary classification structure [102]. On the other hand, when trained with an identify classification loss alone, the learned model for re-identification can hardly be transferred to other domains. Therefore, an instance verification branch using either direct metric learning or verification loss is needed in parallel [44]. Moreover, deep neural networks in classification and verification modules are usually isolated from each other, causing computational power waste. For further discussion on this topic, please refer to Section 3.4.

In this section, we propose a classification model established on metric learning, where similarities defined on the learned embedding are directly employed for classification. In our multiple object tracking algorithm design [22], each potential identity has its own appearance model based on a well-learned metric. Any incoming detected object instance is classified according to its similarities to established models of all the potential identities. Hence, it is natural to explore classification-guided metric learning with the family of directional metrics in Eq. (3.5). In accordance with appearance models for tracking [22], softmax cross-entropy loss is chosen for classification training. For more details regarding the application of our re-identification model in a tracking framework, please refer to Chapter 4.

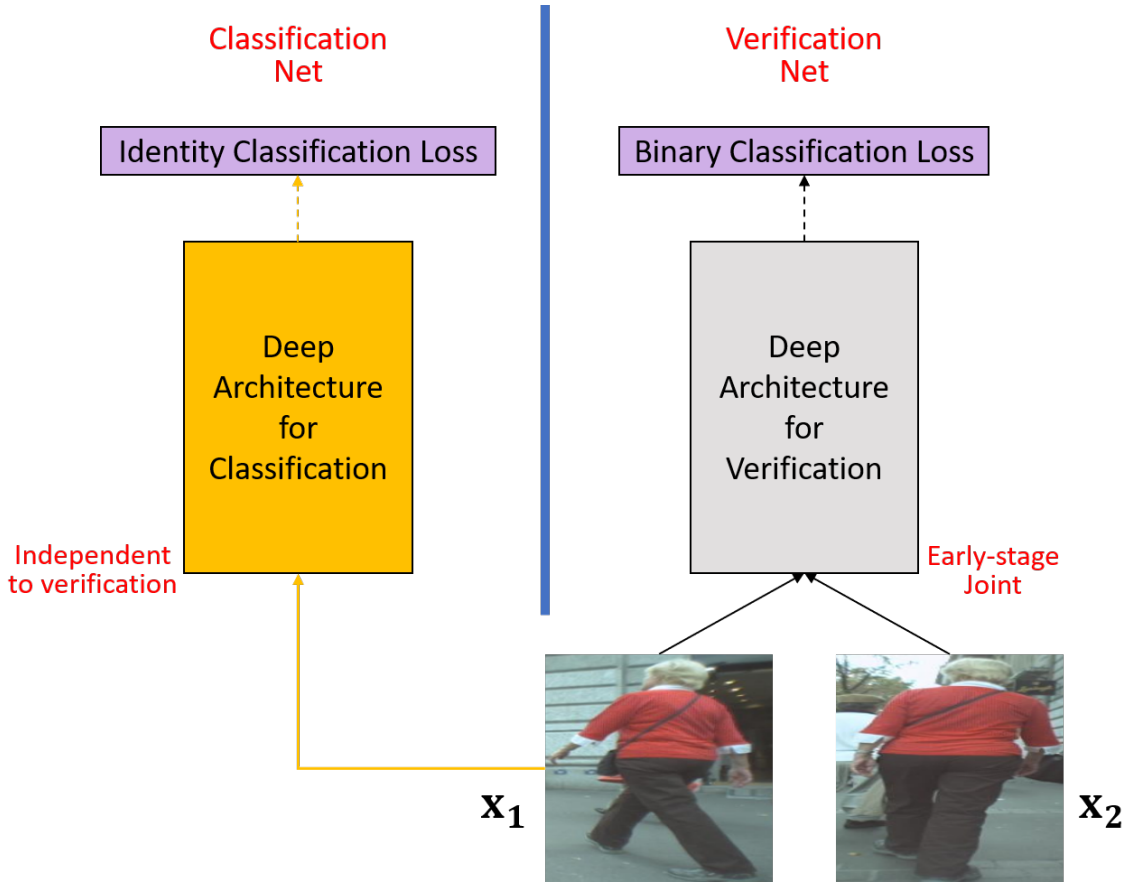


Figure 3.6 Illustration of re-identification via classification with a multitask fashion which can be partially or entirely observed in [31, 102, 17]. Compared to the design shown in Fig. 3.7, different tasks are separated from each other and employ independent deep architectures. In the verification task, early-stage joint of two branches are commonly used and no explicit embedding is learned.

3.3.2 Metric learning with classification loss

The motivation of using classification loss is to provide explicit neighborhood context within identity clusters. Instead of pulling instances in positive pairs together and pushing those in negative ones away, it may be easier and more efficient to draw instances to their own common cluster centers while keeping the centers apart from each other.

Therefore, we maintain an anchor for each identity in the training set. An anchor is a unit vector on the embedding space, working as a temporary cluster center. For each iteration of stochastic gradient descent optimization, instances in the mini-batch are compared to all the anchors, contrary to comparison with each other. Concretely, we inherited the singleton expansion in direct metric learning to calculate the directional metrics, and one of the two

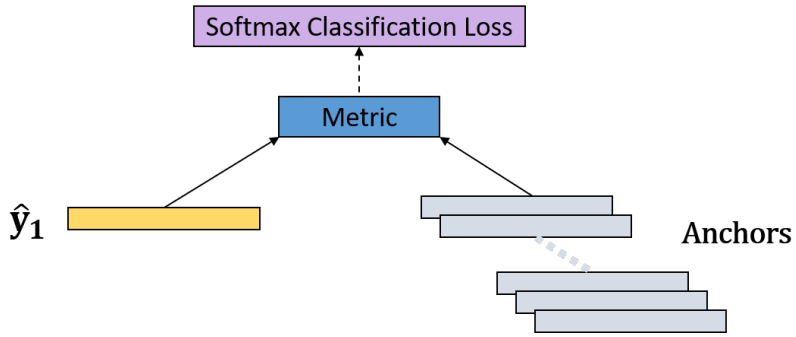


Figure 3.7 Classification-guided metric learning. Each input image passes through the same deep architecture in Fig. 3.2 and is represented by a unit vector after normalization. Instead of examining the similarities between each other, input instances are compared to the anchors of all the identities in the training dataset. With the help of a softmax classification, images are drawn towards the anchor of its identity while pushed away from the others. In the meantime, anchors also memorize the neighborhood context of processed instances.

branches is substituted with the list of anchors which is maintained and updated through the entire training phase.

The goal of optimization is minimizing the distance (or maximizing the similarity) between an instance and its corresponding anchor, keeping it away from other anchors. In direct metric learning, only the neighborhood relationship within the current mini-batch is considered. With our design, the neighborhood information of the entire mini-batch history is extracted and preserved by the anchors and their relative positions. As optimization goes on, the anchors map out all the clusters on the embedding space as a guidance for incoming instance mini-batches.

To optimize the model with anchors, we can either use a pairwise loss with hard instance mining, or simply classify each instance to its identity based on the learned metric. The latter is more intuitive and easier to implement using a classifier like softmax. An input instance is compared to the anchors of all the identities and the softmax classifier is established upon the obtained similarities. Before finally giving the definition of a cross-entropy loss, the normalized directional metrics in Eq. (3.6) need to be rewrite as similarities (radial measurement) in alignment with softmax classifier:

$$\begin{cases} \hat{s}_{geo} = 1 - \frac{2}{\pi} d_{geo} \\ \hat{s}_{L_2} = 1 - d_{L_2} \\ \hat{s}_{cos} = s_{cos} \\ \hat{s}_{tri} = 2s_{tri} - 1 \end{cases} \tag{3.7}$$

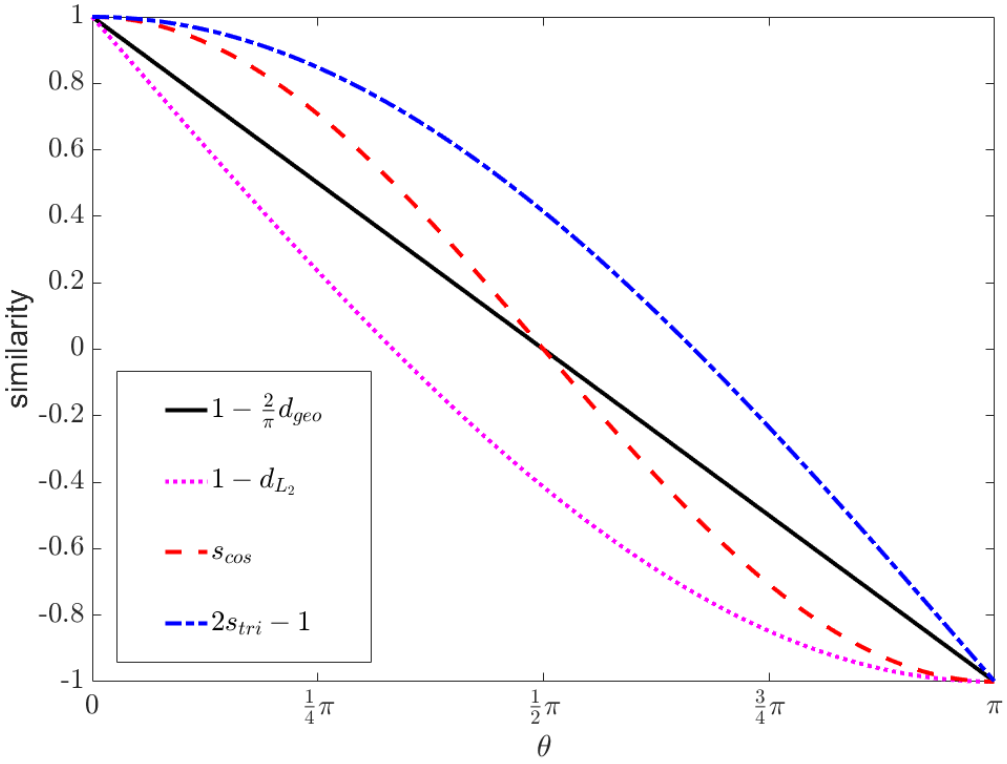


Figure 3.8 Comparison of normalized similarities. Distances on the tangential direction shown in Fig. 3.5 are converted here onto the radial direction. As the included angle in radian θ of the two vectors grows, the similarities now descend.

Based on the normalized similarities $\{\hat{s} \in [-1,1]\}$, we propose the metric learning-based softmax classifier for any input instance:

$$p_k = \frac{\exp(\lambda \hat{s}_k)}{\sum_{k'=1}^{K} \exp(\lambda \hat{s}_{k'})} \tag{3.8}$$

where p_k is the softmax probability of the input instance having identity k among all the K identities in the training set. It is directly determined by the similarities $\{\hat{s}_{k'} | k' = 1,2,\cdots,K\}$ between the instance and all the anchors. Given that the entry $\hat{s}$ is limited to $[-1,1]$ rather than $(-\infty,\infty)$, a constant $\lambda > 0$ is introduced to regularize the significance of similarities. It regulates the distribution of identity clusters on the embedding space just like the margin in direct metric learning approaches, and influences further the re-identification performance. The determination of λ is pertinent to both the chosen metric and K, the total number of identities. We will discuss more on λ with experimental results in Section 3.5.

The classifier is trained using SGD optimization with mini-batches of instances. For an iteration of optimization, every instance in the training batch B has its ground-truth identity

label $k^* \in \{1, 2, \cdots, K\}$. A standard cross-entropy loss derived from the softmax classifier Eq. (3.8) is employed to learn from the batch B:

$$\hat{L}_{cls} = -\sum_{B} \log p_{k^*} \tag{3.9}$$

During testing, the classifier is discarded since identities are different in the testing domain. The learned embedding is preserved for object re-identification. Experiments show a remarkable improvement over direct metric learning after trained with this classification loss. Much less oscillations are observed while re-identification accuracy has seen a significant boost.

3.4 Re-identification in mono-camera video tracking

Our Siamese network and its classification-guided isotope are designed for object tracking in mono-camera videos. Hence, some strategies are chosen to accelerate or facilitate the data association in tracking tasks. In this section, we interpret several design details by comparing our proposition with state-of-the-art re-identification or other kind of appearance models applied in multiple object tracking.

3.4.1 Comparison with verification-modeled structures

As mentioned in 2.4, object re-identification can be formulated in many ways. In multiple object tracking, re-identification serves for appearance-based data association and is usually formulated as a binary classification/verification task. Given a pair of detected instances in different frames, re-identification models are supposed to predict whether the pair is positive. For this reason, unlike learning an embedding, some tracking algorithms manage to train an explicit binary classification model using verification losses [31, 17, 102].

For instance, one of the most successful multiple object tracking algorithms [102] employed "stacked" networks [56, 17, 102] for re-identification, as shown in Fig. 2.9. Unlike standard Siamese network with limited conjoinment, confluence of the two branches of the stacked nets happens much earlier, even at the input stage. The two branches are concatenated along the image (RGB) channel axis. Their information is exchanged through the deep neural network. The earlier the "stack" happens, the better its performance is [56]. With a softmax classifier, stacked nets can realize pairwise verification. Auxiliary information is also often stacked together: associated optical flow maps [56] or body part detectors [101, 102] serve as an alignment guidance and contribute a lot to pairwise comparison.

However, such early-stage and dense connection between branches results in asymmetric outputs. Given two input images $\mathbf{x}_1$ and $\mathbf{x}_2$, the prediction values for $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{x}_1, \mathbf{x}_2)$ may differ from each other, which can be annoying.

A more serious problem is their huge computational cost in multiple object tracking tasks. On the one hand, there is no more weight-sharing between the two branches. Inter-branch information exchange requires layers (no matter dense layers or convolutional layers) to conduct fully connected operations along the stacked axis, which doubles the number of parameters in the network. On the other hand, using cross-image information from both inputs at an early stage means more pair-specific computation. Both images in a pair need to go through the network together to obtain a prediction for this specific pair, and little computation can be reused when one of the two elements is paired up with a third input.

Such a waste of computation makes applying stacked models in multiple object tracking prohibitively expensive, given the fact that many objects may be observed and detected in every frame and a long time interval needs to be investigated. For instance, a tracking algorithm maintains a temporal window of T frames, and let $t = 1, 2, \cdots, T$ denote all the frames within the window. The t-th frame contains n_t observations, i.e. detected object instances to examine. The entire number of pairs is

$$N_{stack} = \sum_{t=1}^{T-1} \sum_{t'=t+1}^{T} n_t \cdot n_{t'} \tag{3.10}$$

For a completely stacked network where all computation is pair-specific, all these pairs must pass through the entire deep architecture. On the contrary, pairwise comparison of Siamese structures that learn image embeddings happens at the very end of the framework, and is often negligible compared to the immense computation of the deep networks. The number of pass-throughs is merely

$$N_{Siam} = \sum_{t=1}^{T} n_t \tag{3.11}$$

Overheads of the two strategies are not on the same order of magnitude at all. Hence, we stick to the strict Siamese fashion instead of stacked ones, no matter how much performance boost the latter might provide.

3.4.2 Comparison with classification-modeled structures

There exist other classification models that contribute to the achievement of state-of-the-art tracking results. Kim et al. [53] revisited MHT with deep convolutional features. Each observation is represented by a feature vector extracted with a pretrained deep convolutional

network. Every potential object has its own linear regressor which indicates whether an observation belongs to it (positive) or not (negative). Each regressor is a linear projection from feature vector space to a scalar response, independent of each other. An ideal linear regressor responds to positive observations with 1, and negative ones with -1.

The training of linear regressors is online: observations in a new frame not only are examined by regressors of all track hypotheses but also serve as their training data. Each hypothetical object incorporates a possible observation (sometimes none) into its positive instances while disperses the rest. Its regressor will then be adjusted to minimize the square error of its responses to all observations after the update.

This online strategy results in robust discriminative appearance model. After a short sequence of training, the online model is usually steady enough against drifting. Also, CNN features prove to be powerful for fine-grained inner-category recognition. The multiclass recognition problem is factorized into several binary regression/classification tasks, and hence, low-dimensional (e.g. 256-D) feature vector space can work well.

However, there are also some drawbacks. The convolutional network for feature extraction is trained apart from the data association task. The feature vector space is not optimized for the inner-category linear regressors, which may also be improved by introducing nonlinearity. As a result, this appearance model is outperformed by our classification-guided metric learning model when used for tracking (see Chapter 4).

Beyond the models already applied in tracking tasks, there are more classification models for re-identification [31, 59]. Contrary to these approaches, classification in our model is not realized with an affixed and adjuvant branch whose knowledge cannot be transferred to other domains. Our classification-guided metric learning structure has nearly no extra overhead than the vanilla one, except for the negligible maintenance cost of a matrix containing all the anchors.

3.4.3 Auxiliary information for re-identification

Auxiliary information can be very useful for object re-identification, which has been proven in previous work [56, 78, 101, 102]. Finer-grained recognition results provide explicit guidance of pattern alignment for further comparison. Keypoint detection is a good way of aligning important object parts. Pixel-level segmentation may also be a helpful auxiliary information provider. Segmentation results, e.g. masks [39], can be directly "stacked" with original input images to eliminate irrelevant background. Another possible approach is multitask learning with segmentation mask branch, as in [39].

There are also some category-aware or task-specific tricks to improve the re-identification performance. For instance, human is the most studied category among others in re-identification.

Human body part detector [78] is a successful and widely-adopted guidance for re-identification and tracking [78, 101, 102]. Techniques like re-ranking [121] can substantially boost the ranking accuracy.

In addition, a large training dataset helps a lot in training deep and sophisticated networks. Data augmentation techniques like random erasing [122] are also an important approach of training more robust re-identification models.

In this dissertation, we deal with multiple object tracking problems. MOT Challenges [67], the most famous multiple object tracking benchmark, encourage tracking algorithms without additional information other than the provided detection results. Therefore, we keep our re-identification model and its training as simple as possible and leave the perfection of re-identification algorithm for future research. In this chapter, all the re-identification is realized without auxiliary information.

3.5 Experiments and discussion

In this section, we present some experimental results and discussions related to our metric learning-based re-identification model design. All experiments are conducted on the CUHK03 re-identification benchmark [58]. The benchmark has two subsets of images: a "detected" set containing roughly aligned object instances which are automatically detected and resized and a "labeled" set constructed by refining the alignment of detected objects via human expert labeling. In this section, we only report experimental results on the "detected" set, which is conventionally considered more challenging and conforms the real application of re-identification.

Under its latest training/testing split protocol, CUHK03 consists of 767 identities totaling 7365 instances for training and 700 identities for testing, with a subdivision of 1400 query instances and 5332 gallery instances. Re-identification is formulated here as a ranking problem. Although the design is more suitable for pairwise verification or classification, our proposed model achieves comparable ranking results to the state-of-the-art approaches. First rank accuracy (Rank@1) and mean average precision (mAP) on its testing set are the official evaluation metrics of this benchmark. We also plot cumulative match characteristic (CMC) curves for evaluation, especially when comparing different strategies and designs.

3.5.1 Influence of classification guidance

First, we compare the two structures proposed in this chapter, the vanilla Siamese network and its classification-guided variation. To compare both structures, we use the same

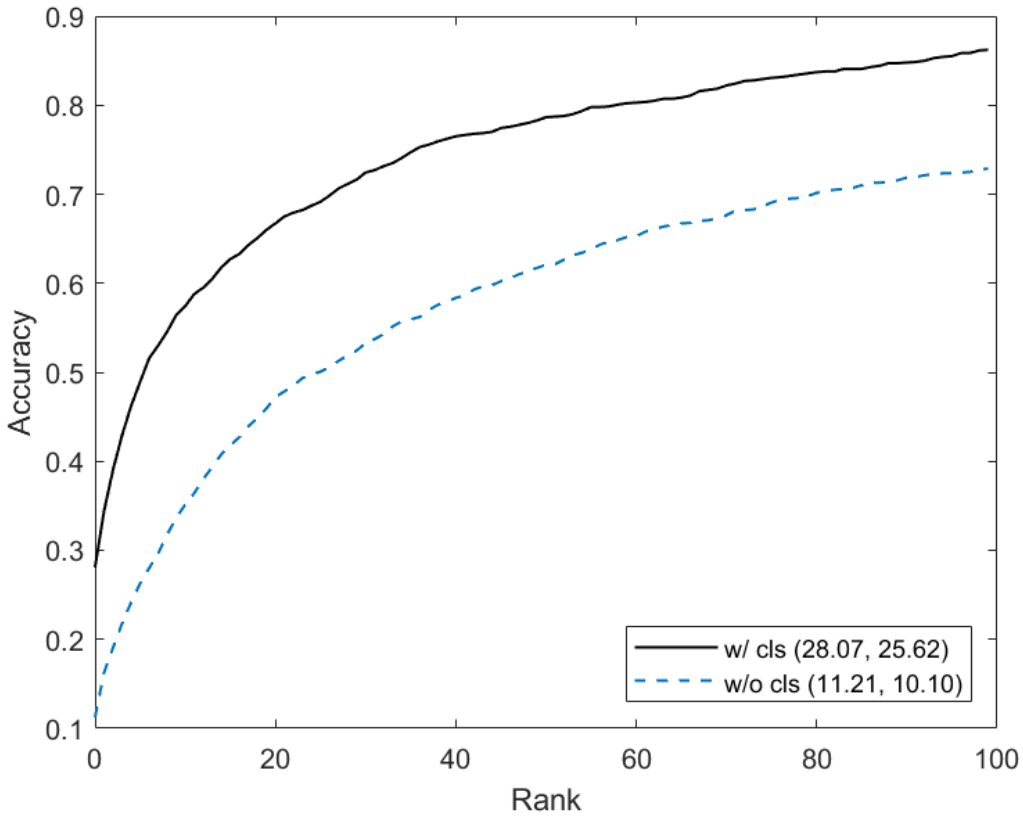


Figure 3.9 Illustration of the influence of classification guidance. CMC curves of the two structures (with and without classification guidance) are plotted according to the re-identification ranking results. X-axis stands for rank and Y-axis stands for re-identification accuracy. The same configuration set is applied for both structures: ResNet50 as backbone with average pooling as head, cosine similarity as metric, embedding space dimension equal to 256.

configurations of backbone net, head net, metric, embedding space dimension, etc. Only the parts relating to losses like the margin $\mu = 0.3$ and the regularizer $\lambda = 5$ are set separately to their own optimums.

The influence of classification guidance is conspicuous. For simplicity, we only present their comparison under a set of configurations (ResNet50 with average pooling, cosine similarity, 256-dimensional embedding space) and plot their CMC curves in Fig. 3.9. Similar patterns can be observed for other configurations.

The rank@1 (28.07%) and mAP (25.62%) of classification-guided structure is much higher than the vanilla version. It proves the effectiveness of the appended softmax classifier. Besides, training the former is much easier and faster. In the rest of this section, we only report the experimental results of the version with classification guidance.

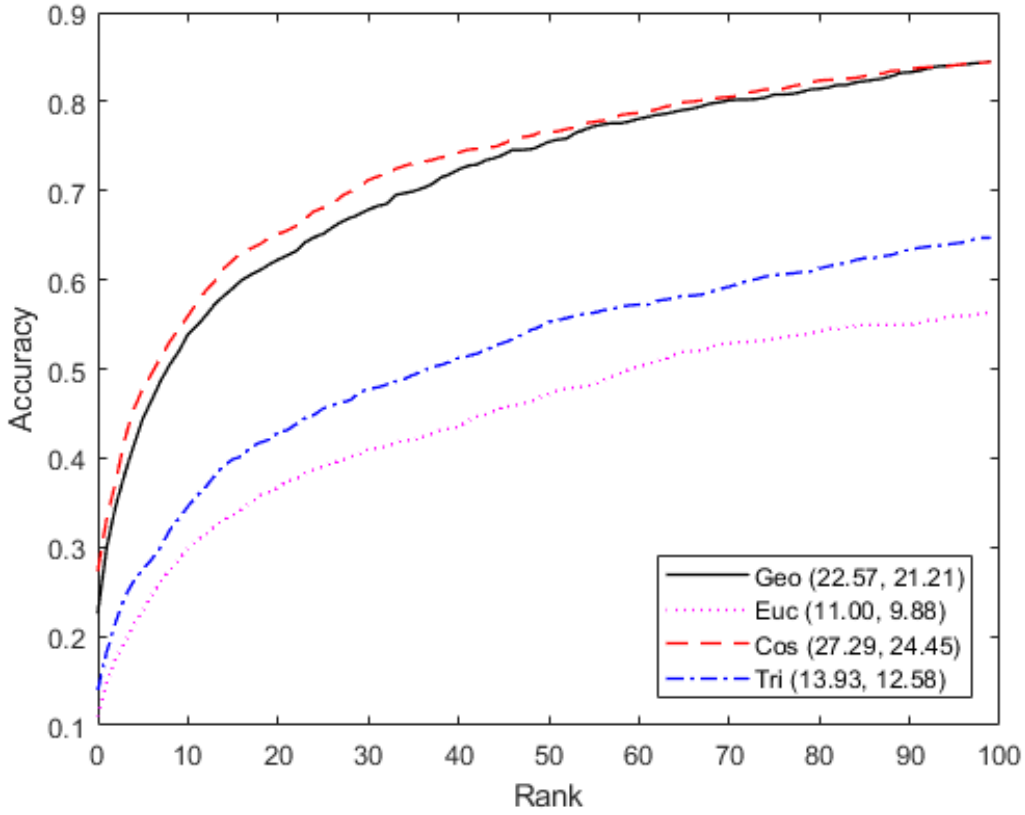


Figure 3.10 Performance of various metrics illustrated by CMC curves. X-axis stands for rank and Y-axis stands for re-identification accuracy. All the results are obtained under the same hyperparameter set: ResNet50 with a 7×7 average pooling as deep architecture, 128-dimensional embedding space and $\lambda = 5$.

3.5.2 Comparison of metrics

As mentioned in Section 2.4.2, metric choice is paramount for the quality of learned embedding. We compare the embedding learned under the 4 directional metrics. Since we report the results of the classification-guided structure, the normalized similarities defined in Eq. (3.7) are involved. All the other hyperparameters are identically set as follows: ResNet50 as backbone with a 7×7 average pooling as head, mapped to a 128-dimensional target space and measured under $\lambda = 5$.

The CMC curves as well as the values of rank@1 and mAP are shown in Fig. 3.10. Apparently, cosine similarity and geodesic distance perform remarkably better than triangular similarity and Euclidean distance. Particularly, since geodesic distance is the arccosine of cosine similarity, their performances only have a minor difference. However, the other two metrics suffers from a constant inefficiency no matter what hyperparameters we choose. This

phenomenon is hard to explain. A possible theory is that the root sum squared in Euclidean distance and triangular similarity is harder for gradients to backpropagate than the inner product in cosine similarity. The theory is merely a guesswork and has not been proved by additional experiments.

Given the performance contrast of these metrics, we incline to employ cosine similarity metric learning in further work. Indeed, cosine similarity not only is easy to implement via matrix multiplication, but also has a compact and concise distributive property, which is useful to model a cluster of vectors (see Chapter 4).

3.5.3 Determination of regularizer λ

In this paragraph, we will discuss the functionality of the regularizer λ. Unlike common softmax classifier where entries vary from $(-\infty, \infty)$, the similarities are normalized to $[-1, 1]$. In this case, the softmax probability by its classic definition (without regularizer), i.e. the Eq. (3.8) with $\lambda = 1$, may have a very narrow range, especially when the number of classes/identities K is large. For example, CUHK03 benchmark has $K = 767$ identities for training. Ideally, a positive instance-anchor pair with a cosine similarity equal to 1 has a softmax classification probability approximately equal to $0.0096 << 1$. If the fact that vectors in negative pairs are often perpendicular after training is taken into consideration, the maximum probability will further drop to 0.0035. It is far from enough to learn a one-hot encoding via softmax classification.

Hence, we introduce the multiplier λ to sharpen the difference between positive and negative pairs. The greater λ is, the more sensitive our classification model is to similarities. Correspondingly, the distribution within each identity will be more concentrated. In other words, λ indirectly controls the desired margin μ between negative pairs. We investigated here three different λ values, 3, 5 and 10, whose performances are illustrated in Fig. 3.11. The experiments are conducted with hyperparameters as follows: ResNet50 followed by a 7×7 average pooling, cosine similarity and 128-dimensional embedding space.

As the curves suggest, λ value is pertinent to the re-identification performance. The model trained under $\lambda = 5$ outperforms the other two for the $K = 767$ classification. The determination of λ depends on the similarity as well, because different metrics have different similarity values when the included angle $\theta = \frac{\pi}{2}$, for perpendicular pairs. Empirically, we choose λ according to the following condition:

$$-\log \frac{\exp(\lambda)}{\exp(\lambda) + (K-1)\exp(\lambda \hat{s}_{\theta=\frac{\pi}{2}})} \approx 2 \tag{3.12}$$

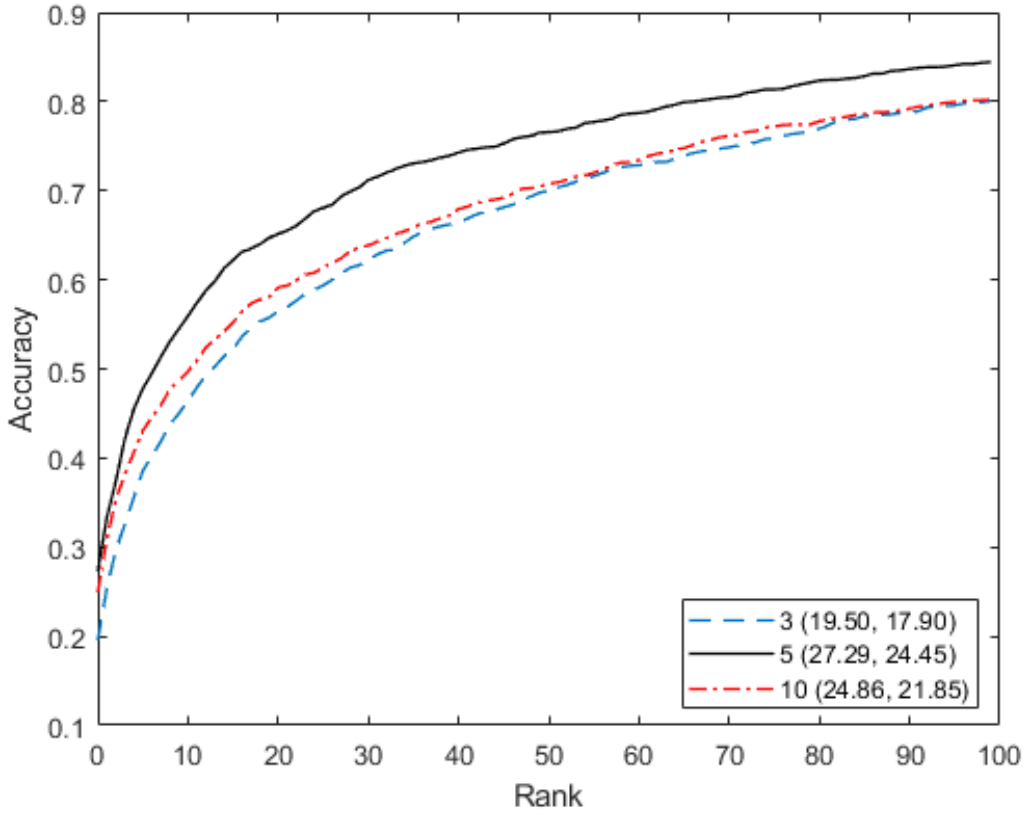


Figure 3.11 Illustration of the significance of the regularizer λ. X-axis stands for rank and Y-axis stands for re-identification accuracy. Three CMC curves respectively representing $\lambda = 3, 5, 10$ are compared with each other. All the results are obtained under the same hyperparameter set: ResNet50 with a 7×7 average pooling as deep architecture, 128-dimensional embedding space under cosine similarity.

In our work, for our re-identification models using cosine similarity metric learning trained on CUHK03, we keep $\lambda = 5$.

3.5.4 Influence of embedding space dimensionalities

As reviewed in Section 2.4.2, metric learning usually realizes a dimension reduction by mapping the input onto a lower-dimensional target space. Regarding to the influence of the target space dimensionality, we also conduct some experiments. We select three different dimensionalities that are chosen by many algorithms: 128, 256 and 512. Their ranking performance results are shown in Fig. 3.12. No noticeable difference can be observed if random error is taken into consideration. We can draw a conclusion that embedding space

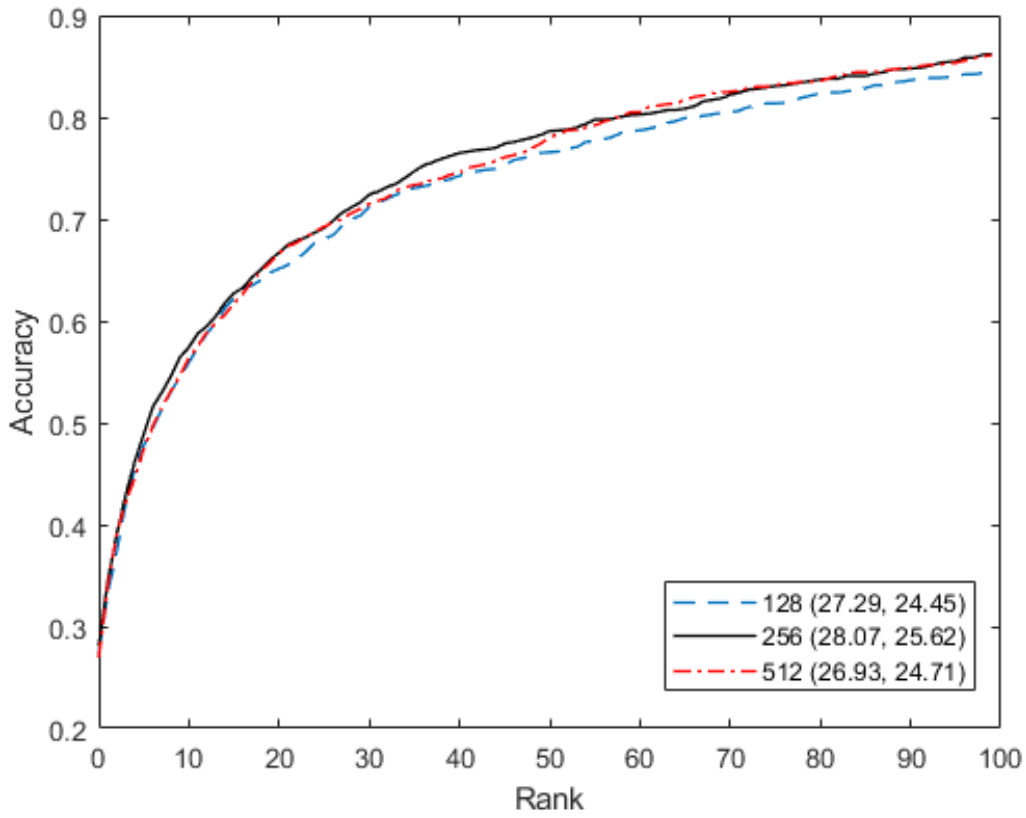


Figure 3.12 CMC curves of different embedding space dimensionalities. X-axis stands for rank and Y-axis stands for re-identification accuracy. All the results are obtained under the same hyperparameter set: ResNet50 with a 7×7 average pooling as deep architecture, under cosine similarity.

dimensionality has little influence on re-identification as long as it is within a reasonable range.

On the other hand, a lower dimensionality means less computation during the application of the learned metric. Given that the performance of the cosine similarity on the 128-dimensional target space is slightly weaker than the others, we choose to map image inputs onto a 256-dimensional space in multiple object tracking tasks (Chapter 4), as a balance between performance and overhead.

3.5.5 Various head networks

As mentioned above, the chosen deep architectures profoundly influence the performance of re-identification algorithms. In this section, we also compare different deep architectures,

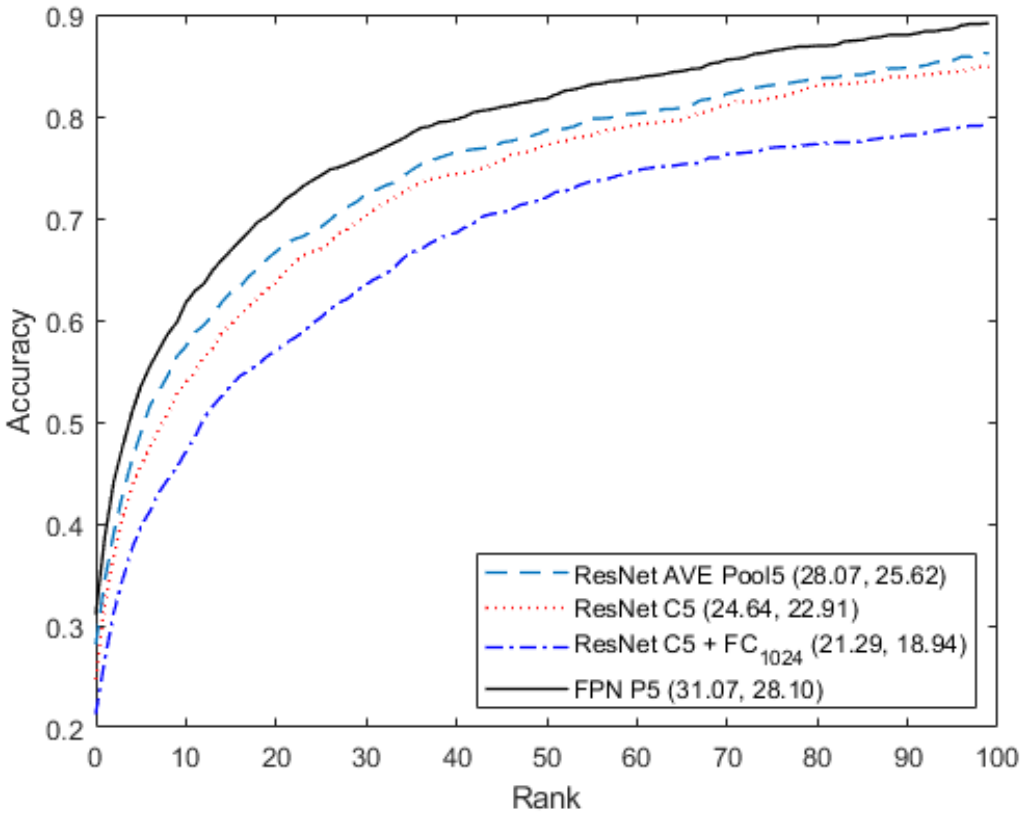


Figure 3.13 Comparison among various head structures. CMC curves are drawn with X-axis standing for rank and Y-axis for re-identification accuracy. Four head nets are introduced between the final feature map $C5$ of ResNet50 and a 256-dimensional embedding space with cosine similarity: a 7×7 average pooling layer, $C5$ itself, an additional dense layer, and two convolutional layers as the FPN $P5$ [61].

especially how all the parts of deep networks affect the re-identification model. We first study the influence of head subnetworks in this paragraph.

In Fig. 3.13, we list the ranking results of four different head nets following the same ResNet50 network. From the simplest to the most complex, we have global average pooling layer following the final feature map $C5$ as in [42], FPN $P5$ (two consecutive convolutional layers with dimension reduction) as in [61], the feature map $C5$ without additional layers, and a 1024-dimensional fully-connected layer. The output of each head is mapped onto a 256-dimensional target space by a dense layer to learn an appropriate embedding under the cosine similarity.

As Fig. 3.13 shows, the embedding learned with FPN $P5$ head outperforms the others. One observation is that the performance is pertinent to the parameter scales of these head nets.

The average pooling layer simply summarizes all the channels of the feature map $C5$ without additional parameters. Its performance is acceptable and comparable to that of $C5$ itself. However, the multi-layer perceptron structure (with an extra 1024-dimensional dense layer) abundantly increases the head parameter scale that exceeds the requirement of the training dataset. The re-identification model becomes hard to train and its performance significantly degenerates.

The FPN $P5$ head, on the contrary, provides an appropriate parameter set while explicitly preserves the local spatial information. Hence, it boosts the reasoning of our re-identification model. In its tracking application, we choose the model learned with this head net.

3.5.6 Stride variation of feature maps

As previously discussed, feature maps of different strides compared to the original input image yield object recognition on different scales. Although many recent studies suggest finer-grained feature maps of smaller strides are helpful to better object detection and segmentation (see Section 2.3), our investigation on this topic draws a different conclusion.

We conducted experiments similar to those in the previous subsection. With a ResNet50 backbone and a 256-dimensional embedding space, we compare embeddings derived from different feature maps under the cosine similarity. As shown in Fig. 3.14, we compare the last two feature maps $C5$ and $C4$ with strides of 32 and 16, respectively. We employ the same head net in FPN [61] where after upsampling, the final feature map and the second last feature map $C4$ are overlaid together as a $P4$ layer. Also, we append two consecutive convolutional layers on the ResNet50 $C4$ feature map, which gives the same head structure as FPN $P5$. We denote such architecture as FPN $C4$ and compare its performance directly with the others (FPN $P4$ and $P5$).

As the CMC curves show, FPN $C4$ falls behind in terms of re-identification accuracy. Besides, FPN $P4$ barely gains from the second last feature map of a smaller stride. A possible explanation is that all the input images are resized to 224×224 in our implementation and the last feature map is accurate enough. On the other hand, the larger number of neurons on the shallower feature map may cause a performance drop in FPN $C4$ facing insufficient training data.

For this reason, we simply choose the FPN $P5$ feature map to learn our appearance model of the tracking algorithm in Chapter 4.

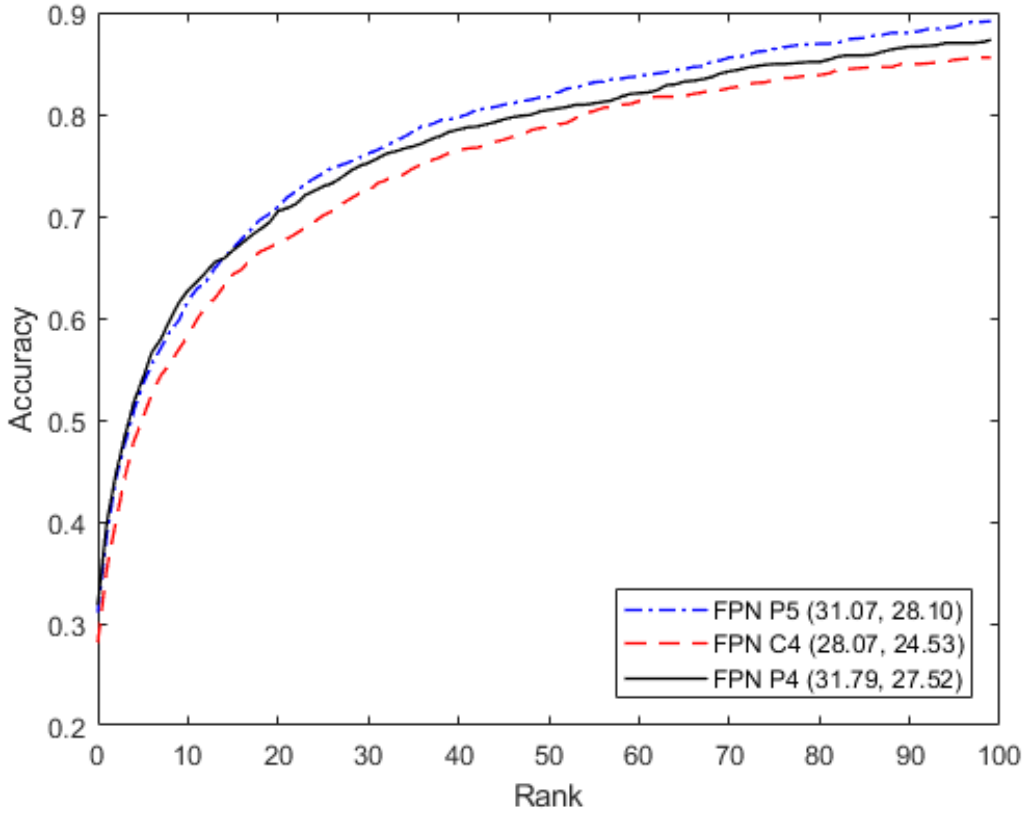


Figure 3.14 Influence on re-identification accuracy of feature map strides. CMC curves are drawn with X-axis standing for rank and Y-axis for re-identification accuracy. Embeddings on a 256-dimensional space under cosine similarity are learned with feature maps of different strides in ResNet50. The FPN heads are described in Subsection 3.5.6 and for more details, please refer to [61].

3.5.7 Different backbone networks

In re-identification tasks, the choice of deep backbone networks is also important. The previous discussion in this section exposes the problem of insufficient training data in current re-identification tasks. Therefore, shallower and simpler architecture may be easier to train. Some recent re-identification algorithms use their own simplified network designs. Contrary to this fashion, we stick to canonical deep networks pre-trained on ImageNet, e.g. VGG16 and ResNet50.

The result comparison between ResNet50 and VGG16 is illustrated in Fig. 3.15. Besides the FPN $P5$ structure, their original reasoning heads (average pooling and max pooling, respectively) are also contrasted. VGG16 nets have a clear advantage over ResNet50 in our experiments no matter which head structure is used. It can be concluded that the obvious

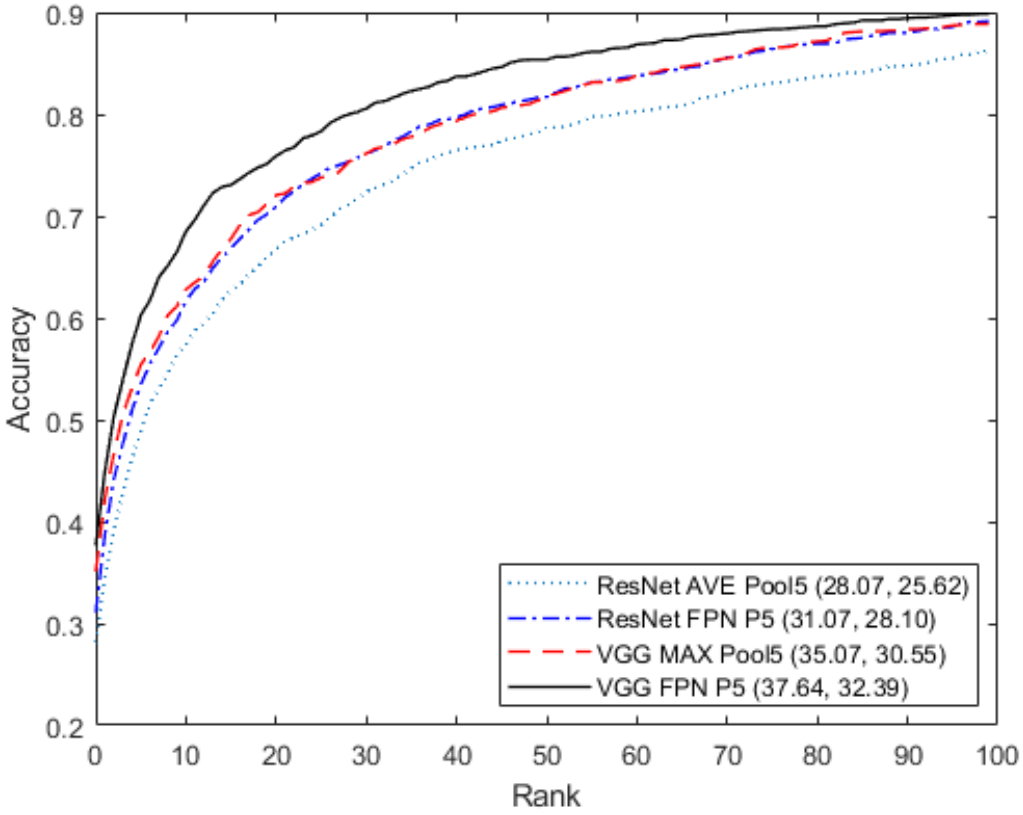


Figure 3.15 Performance of different backbone networks in the proposed re-identification method. CMC curves of VGG16 and ResNet50 are contrasted. X-axis stands for rank and Y-axis stands for re-identification accuracy. Experiments are conducted with the same hyperparameter set: a 256-dimensional embedding space under cosine similarity, only the last feature maps of both backbone networks are considered.

distinction comes from different backbone network architectures. VGG16 is more suitable in our re-identification design given the training set of the CUHK03 benchmark.

3.5.8 Comparison with the state of the art

Finally, we compare our model with state-of-the-art methods in re-identification field. Even trained with pairwise loss functions, our re-identification model achieves comparable performance to state-of-the-art sophisticated triplet loss-driven ones. Table 3.1 summarizes the results of some algorithms published on the CUHK03 benchmark. Only the evaluation on the detected subsets is reported.

Our method is not specifically designed for ranking but more like a binary classifier realizing verification tasks. It is natural that it is not outstanding enough, especially compared

Table 3.1 Results on the MOT16 benchmark compared with leader algorithms

Method	Rank@1	mAP
BOW+XQDA [119]	6.36%	6.39%
LOMO+XQDA [60]	12.8%	11.5%
IDE [120]	21.3%	19.7%
IDE+XQDA+RR [121]	34.7%	37.4%
Proposed	37.64%	32.39%
DPFL [18]	40.7%	37.0%
TriNet [44]	50.5%	46.47%
TriNet+Re [122]	55.5%	50.74%
TriNet+RR+Re [122]	64.43%	64.75%

to those cutting-edge methods based on triplet losses which are more suitable for ranking and using bespoke techniques like re-ranking (denoted by RR in Table 3.1). However, our method is the simplest to implement. No sophisticated triplet mining nor additional re-ranking training is required. On the other hand, our re-identification model already outperforms those applied in multiple object tracking tasks.

The key to overcoming occlusion problem is the association of spatially and temporally separated objects. The lifted multi-cut algorithm [102] showed the importance of a robust model for long-range association. By comparing re-identification accuracy with "stack" net in LMP, we prove our deep Siamese network a qualified substitute.

As described in [102], we train our network on CUHK03 [58] and Market-1501 [119], as well as 5 sequences in MOT16. MOT15 is not included because it has many identical sequences with MOT16. The pairwise comparison accuracy analysis is done on 2 sequences in MOT16 training set, denoted as MOT16-02 and MOT16-11. Positive and negative detection pairs are sampled with a ratio 1:4 in all time intervals. The verification accuracy, i.e. the ratio of correctly classified pairs, is used to measure the data association ability. For more details, please refer to [102].

As shown in Fig. 3.16, Our method achieved commensurate or even slightly better accuracy, even at longer time span. Besides, the pairwise comparison is as simple as an inner product. It consumes much less time than passing through the partial or entire deep network.

3.6 Conclusion

In this chapter, we investigated and compared some metric learning architectures for object re-identification. Recent work suggests that direct metric learning is not effective

nor efficient, since it is not task-oriented. Besides, direct manipulation of distance on the embedding space under mini-batch training ignores the neighborhood context among batches. To overcome these problems, more task-related modification has been made. Extra per-identity classification branches independent of the metric learning part are introduced in parallel as guidance. Moreover, some algorithms abandon the metric learning framework and use only task-specific structures and losses instead of an explicit embedding. However, such modification often provokes huge load of extra overheads caused by non-reusable computation.

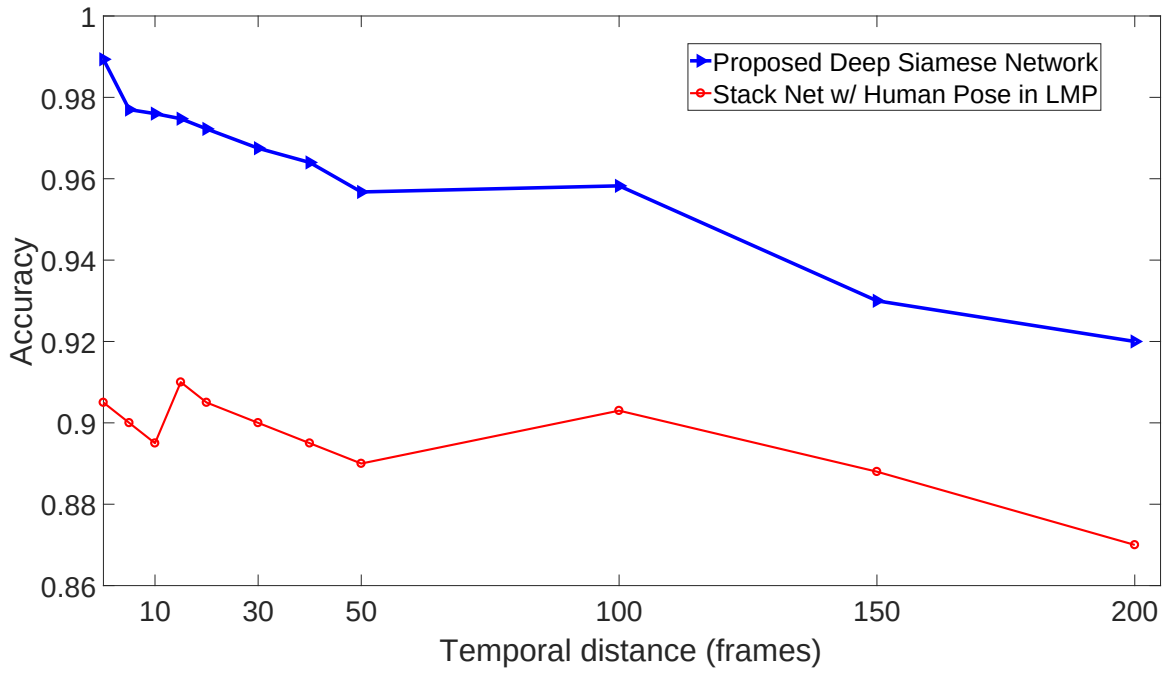
Hence, we propose in this section a simple but effective per-identity classification guidance. Contrary to redundant branches, the classification is based directly on the embedding space of direct metric learning. Each identity is represented by its cluster center on the embedding space. Instead of directly comparing two input images, we compare the feature vector of an image with cluster centers of all the identities. The metrics defined on the embedding space are no longer controlled directly by a distance loss function, but by a per-identity classifier. The design of still learning an explicit embedding guarantees easy knowledge transfer, as well as negligible overheads.

We conducted a survey on the influences of various metric learning architectures and strategies, including the introduction of the classification guidance summarized above. Some interesting preliminary conclusions are drawn from experimental results. Specifically, the proposed classification guidance brings about improvements in both learning accuracy and training efficiency.

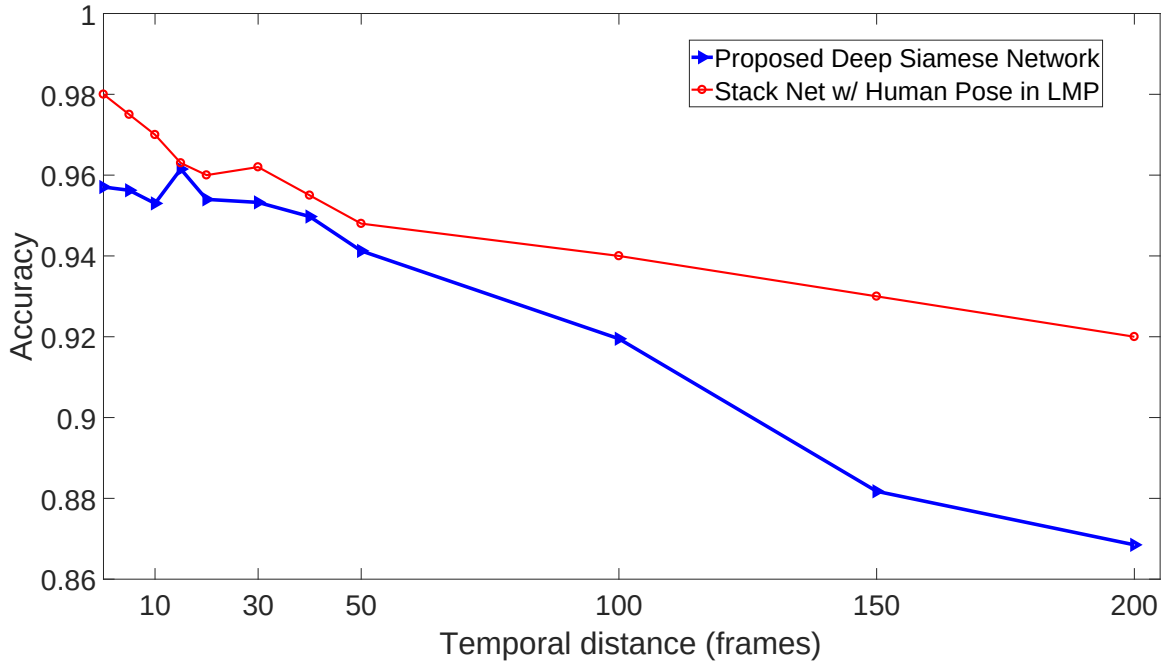
Most trending object re-identification benchmarks evaluate only the ranking performances of re-identification algorithms. Our proposed algorithm is designed in the first place to serve as an appearance model in data association in multiple object tracking, which is formulated as a multi-class classification problem. For this reason, our re-identification model is guided with a per-identity classification loss rather than a ranking-targeted one. It is understandable that our best ranking performance is not comparable to the state-of-the-art algorithms, especially those using triplet losses and tricks specifically designed for ranking tasks (e.g. re-ranking). Despite this, our classification-guided re-identification model makes an obvious improvement over the direct metric learning and its variants with redundant task-oriented parts, especially when applied in tracking tasks.

Many interesting and promising improvements can be investigated in the future. For example, the idea of loose inner-cluster distribution raised in triplet losses can be borrowed. An upper limit can be imposed on all the entries of the per-identity classifier to prevent a condensed distribution within every cluster. Data augmentation techniques like random

erasing [122] can also be introduced in our research. Cross-dataset learning may improvement the performance as well.



(a) MOT16-02



(b) MOT16-11

Figure 3.16 Accuracy of pairwise comparisons on the MOT16-02 (a) and the MOT16-11 (b) sequences. The classic deep Siamese network without the guidance of identity classification (blue lines) is as good as the "stack" net with human body part detectors (red lines) in LMP [102], whatever the temporal distance is.

Chapter 4

Metric Learning for Multiple Object Tracking

4.1 Introduction

Multiple object tracking is a challenging task compared to traditional single target tracking. The former requires maintaining the identities of objects in addition to estimating their locations. Two different kinds of object recognition are therefore needed. In cluttered real-world scenes, objects should be detected out of background, which involves inter-category recognition. At the same time, intra-category differentiation of object instances among video frames is necessary to guarantee the legibility and independence of each trajectory. Features, objectives, training data and strategies of both kinds of recognition differ from or even contrast each other. Hence, most multiple object tracking approaches decompose the task as two separate parts. A standalone detector passes through all the frames to locate all objects of interest, without using cross-frame information. The detection results through frames are then associated together, relying on spatiotemporal information-involved intra-category recognition. The two-stage scheme is called "tracking-by-detection", as briefly reviewed in subsection 2.5.1. Considering object detection as an independent and outsourcing module, tracking is in fact about designing fast and robust data association algorithms.

The popularization of tracking-by-detection scheme is based on the progress made in object detection field, especially with the help of deep learning. Appearance models for data association [53, 102, 22] also benefit from deep neural networks. In this chapter, we introduce our re-identification model proposed in Chapter 3 as a successful appearance model for tracking, after explanation of trending data association approaches. Additionally, we report a research on the collaboration between appearance and motion models. A dynamic

coupling of both kinds of models can further improve the tracking performance, as the experimental results suggest.

4.2 Data association formulation in tracking-by-detection

Data association is the core of tracking-by-detection algorithms. Given the detection results, multiple object tracking is formulated as a problem of linking observations of the same object in all frames into an independent and disjoint trajectory, as reviewed in Chapter 2. Various optimization algorithms based on graph models, e.g. network flow [116], continuous energy minimization [68], multiple hypothesis tracking [53] and (lifted) subgraph multicut [101, 102], have been designed to solve the data association problem. In this section, we recall in detail the graph model used in multiple hypothesis tracking [53], on which the tracking algorithm proposed in this chapter is based. Some comparison with other graph models, especially those in multicut approaches.

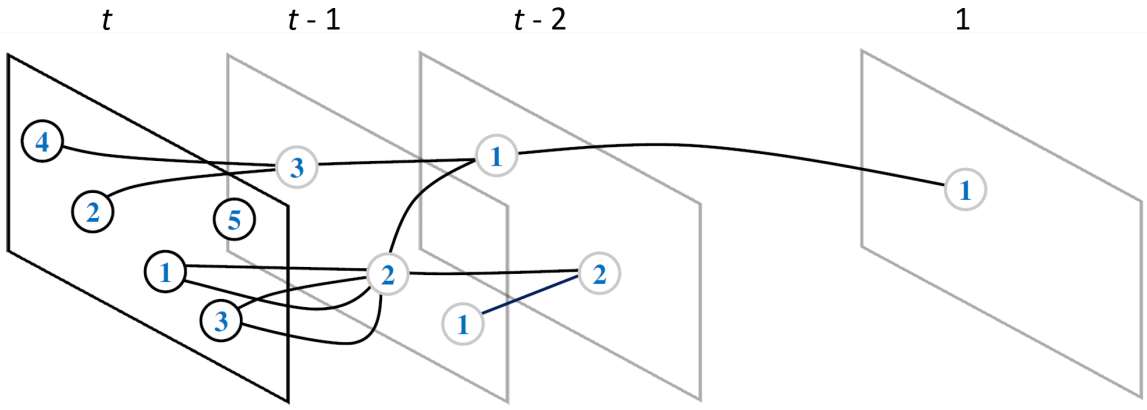
4.2.1 Data association graph models

In many tracking-by-detection algorithms, data association is formulated with graph models containing pre-detected objects. Every object observation is represented by a node in the graph. Usually, the nodes are weighed by their detection confidence. An edge is assigned to any pair of different observations to summarize the connection between them. Various clues from motion/dynamic to appearance have been investigated to model the connection. Tracking is then modeled as an optimization problem with respect to the weighted graph.

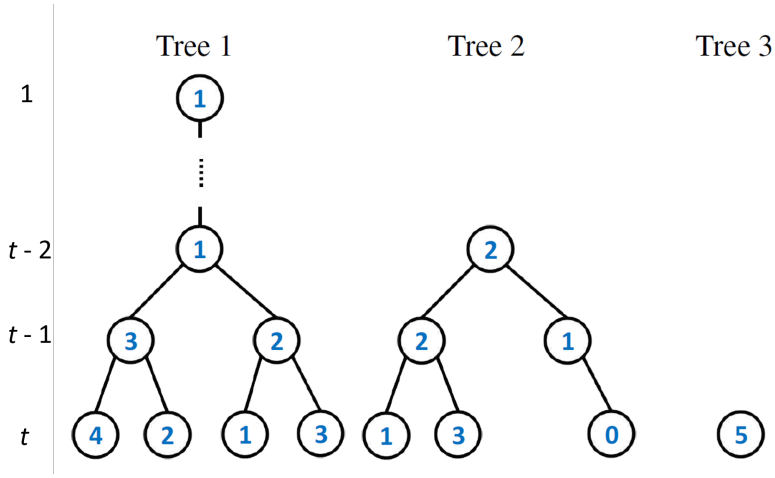
Ideally, an object can be observed in detection results of a frame for no more than once and data association should be dealing with a multipartite graph. Unfortunately, imperfect detections often leave inaccurate observations that cannot be eliminated by non-maximum suppression. On the other hand, trajectories of different objects may temporarily converge during occlusions, incurring lack of observations. Selection or exclusion rules are therefore needed to prevent redundant or merged trajectories.

A popular solution is modeling the tracking as a minimum cost/maximum weight disjoint paths problem [116, 77, 92, 53]. Mutual exclusion is included in the weighing of edges. Techniques like maximum weighted independent set (MWIS) are utilized to guarantee the independence among paths/tracks. Usually, nodes are consecutively linked through frames.

Multiple hypothesis tracking [53] is a representative algorithm among these disjoint paths approaches. As shown in Fig. 4.1(a), tracking is chronologically realized frame by frame. Hence, its data association model has a layered structure. Observations in a frame



(a) Layered graph model



(b) Forest interpretation

Figure 4.1 An example of graph model used in multiple hypothesis tracking, adapted from [53].

are considered mutually exclusive. In multiple hypothesis tracking, the tracking graph is interpreted as a forest, where each tree contains all the plausible tracks or hypotheses starting with the same node and therefore is rooted, as illustrated in Fig. 4.1(b). Since the path from the root to a certain leaf is unique by the connected acyclic definition of trees, leaves of a tree correspond to all its hypotheses. Each detection in the first frame is treated as the root (and the only leaf at the same time) of a new tree. In any other frame $t > 1$, true positive detections can be either the first observations of new objects, or successors of the track sequences established in frame $t-1$. The second case is where association takes place.

The forest $\mathscr{F}^{t-1}$ obtained after frame $t-1$ consists of multiple trees, and each tree $\mathscr{T}_i^{t-1} \in \mathscr{F}^{t-1}$ has multiple leaves (and track hypotheses correspondingly) $\mathscr{L}_i^{t-1}$. The observations $\mathscr{O}^t$

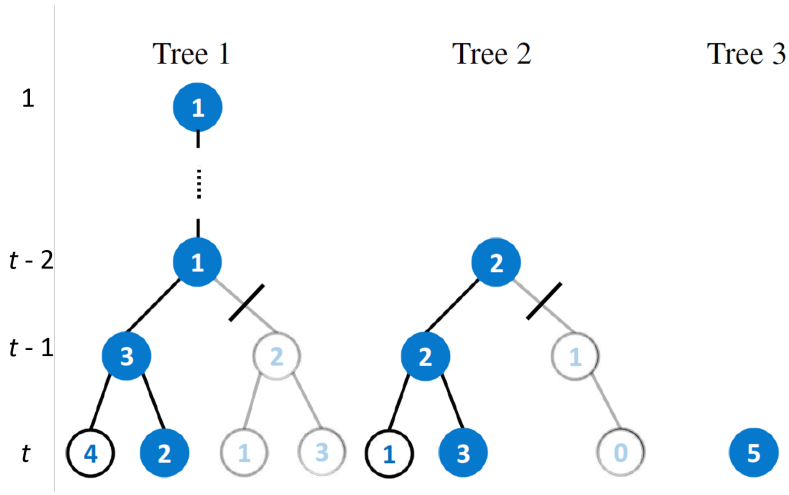
in frame t are to be associated amongst the track hypothesis set $\mathscr{L}^{t-1} = \bigcup_i \mathscr{L}_i^{t-1}$. Concretely, the combination between every leaf $l_i^{t-1} \in \mathscr{L}^{t-1}$ and every new observation $o_j^t \in \mathscr{O}^t$ is examined by tracking models. In another word, every edge (l^{t-1}, o^t) between the two frames is weighed during the association. Some loose spatiotemporal or appearance-based thresholds are applied to reduce the sizes of the trees. Only feasible associations are kept while the other edges are cut off (or weighed as $-\infty$). Although only the edges between two consecutive frames are weighed, their weights are not necessarily determined solely by their vertices, i.e. leaves and new observations. Actually, the ancestors of a leaf are often taken into account to model the corresponding hypothesis [53, 22]. Details of weighing edges will be discussed in the next subsection.

The objective of data association is to find the most probable trajectory for each object from the forest while maintaining the disjointness among trajectories. With respect to the weighted trees, clique techniques are introduced to solve the maximum weighted independent set (MWIS) problem in [53]. From the obtained independent subdivisions of the graph, optimal trajectories can be easily deduced.

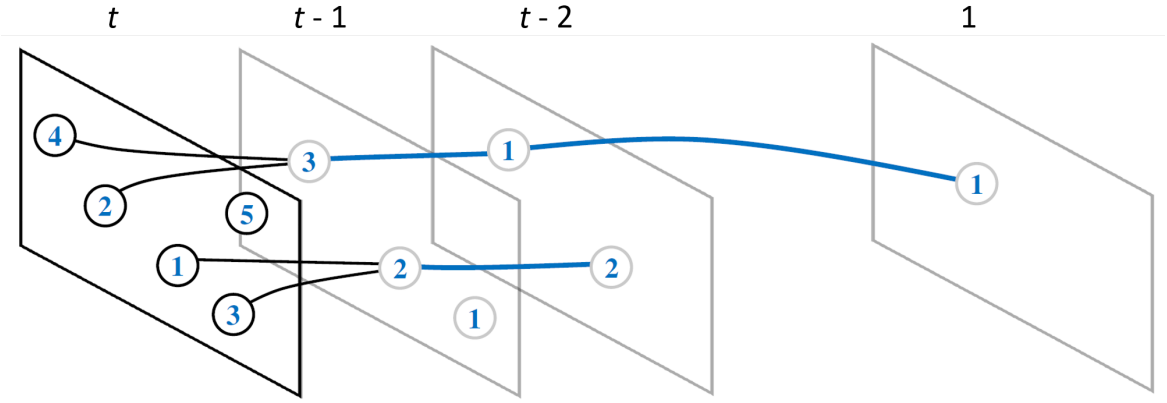
However, hypothesis trees grow exponentially in terms of node numbers. Even after node selection with proper thresholds, the growth is still too fast and causes search space explosion when solving the MWIS problem. In practice, optimal trajectories are calculated every time that a frame is processed. A tree is repeatedly trimmed to keep the optimal hypothesis at the time as its trunk, as shown in Fig. 4.2. Suboptimal branches are cut off unless they are recently forked. The furcation is limited within a temporally sliding window whose length is often set to 5 frames as a tradeoff between tracking accuracy and efficiency. The small window delays the output of confirmed tracks and therefore, multiple hypothesis tracking is a family of noncausal algorithms.

Multiple hypothesis tracking [53] is a successful example of maximum weighted disjoint paths approaches. Such formulation encourages finding as many trajectories as possible. Sometimes, imperfect detection algorithms leave behind inaccurate observations that cannot be eliminated by simple non-maximum suppression as noises. When the same noise is constantly observed in a long enough time span, it can be considered as a redundant track which will be evaluated as false positive for duplication. Even though such noisy observations are connected to the true positive track, they are prone to form their own clique as MWIS formulation suggests.

To address this issue, Tang et al. [100, 101, 6, 102] modeled data association as a minimum cost multicut problem. Instead of finding a subset of hard positive edges, data association is formulated as removing a subset of hard negative edges. Ambiguous observations



(a) Intermediate trimming



(b) Trimmed tree

Figure 4.2 Illustration of trimming the forest shown in Fig. 4.1, adapted from [53].

are inclined to be absorbed by the true positive track rather than split out as an independent track during optimization under the new formulation.

This strategy is claimed to perform better than optimal disjoint paths searching approaches [100–102]. Unfortunately, we have not been able to compare it with multiple hypothesis tracking algorithm using the same set of tracking models in this thesis, due to its patent application and the usage of closed-source commercial software. On the other hand, the multicut approaches demand knowledge of the full graph (or at least a large part of it) before any track can be acquired, compared to small time intervals in path growth approaches. As a result, multicut-based algorithms are slower and require more memory, confirmed by the results on the benchmark MOT Challenge 2016 [1].

1. https://motchallenge.net/results/MOT16

Nevertheless, the tracking performance improvement brought by our re-identification appearance model is the focus of this thesis. In this chapter, we prove that with proper modification, multiple hypothesis tracking can be significantly boosted and even outperform the lifted multicut tracking algorithm [102], the leader of all tracking algorithms published on MOT Challenge.

4.2.2 Edge weights in tracking graphs

As previously discussed, data association is formulated as an optimization problem with respect to a weighted graph model. Besides the various graph models and optimization methods described above, how to weigh the graph is also crucial to the success of data association. Under the tracking-by-detection framework, nodes are weighed directly by their detection confidence scores in most cases. Edge are left to be weighed by tracking models in the data association phase. For most modern multiple object tracking algorithms, both appearance and motion models contribute to the edge weights.

In multiple hypothesis tracking [53], each track hypothesis has its own tracking score S. After associating a new observation in frame t, the hypothesis updates its score by adding the weight of the included edge. In coherence with the denotation in [53], we define the edge weight as the tracking score increment:

$$\Delta S = \alpha_{mot}\Delta S_{mot} + \alpha_{app}\Delta S_{app} \tag{4.1}$$

with subscripts $_{mot}$ and $_{app}$ indicating the terms yielded from motion and appearance models, respectively. Constants α_{mot} and α_{app} are manually chosen as a regularizer between models so that they contribute with a proper ratio. The determination of weight ΔS of an edge is often related to the probability of its pair of vertices being positive. The larger ΔS is, the more likely the instance pair represented by the edge is to be positive. Displacement, scale variation, and other spatiotemporal relationships between the nodes are exploited by motion models to output the motion-based score ΔS_{mot}. Meanwhile, models like deep matching [109, 101] and re-identification [102] evaluate the similarity in appearance of both nodes. The scores ΔS_{mot} and ΔS_{app} are separately calculated in nearly all the trending data association algorithms and summed together with the constant coefficients. Motion and appearance models are mechanically bound together, and seldom collaborate with each other in an active way.

As pointed out in the previous subsection, the weight of an edge is not necessarily determined by its vertices. For instance, most disjoint paths search approaches establish their tracking models on all the examined observations. Many algorithms [8, 53] build and maintain a model for each object along the tracking process. As the trajectory of an object

grows, new observations are compared to its model, rather than directly to all the confirmed instances in its tracking history. Such strategy may save computational time when dealing with complex pairwise comparison. Specifically, in multiple hypothesis tracking [53], a hypothesis is represented by the path from the root to its corresponding leaf. A motion model predicts the hypothetical position of the object in the next frame by examining the sequence of nodes on the path. Not only the leaf but also all its ancestors are labeled as positive examples to learn an object-specific appearance model.

Some tracking algorithms may employ more than one model in each kind. For instance, [102] solicited at the same time deep matching [109] and an inefficient but complex verification-modeled re-identification for appearance guide (discussed also in subsection 3.4.1). For simplicity and conciseness, we keep the expression in Eq. (4.1): outputs of the same kind of models can be regrouped and aggregated as a single score, given their commutative and associative combinations.

In this chapter, we come up with a new appearance model based on our re-identification algorithms in Chapter 3. By inheriting the motion model from [53] and replacing its appearance model with our own, we prove that tracking models, especially well-designed appearance models, are paramount to robust tracking results. As cosine similarity metric learning is the core of our re-identification framework, we can come up with a simple appearance model for each object to tracking without losing any information of the instances in its cluster.

4.3 Appearance model with learned cosine similarity

In this section, we introduce our appearance model. Like the original linear regression appearance model in [53], we establish a model for each hypothesis based on its historical observations. Cosine similarity between the model and a new observation to associate is sent to a softmax classifier whose likelihood output determines the edge weight and further influences the inference of data association. The update of models after associating observations in the current frame is much easier than that in [53], thanks to our model design.

4.3.1 Track hypothesis representation

Our appearance model is based on cosine similarity metric learning, which has been investigated in Chapter 3. Detected object instances are mapped onto an n-dimensional embedding space. The similarity between two detections is measured by the cosine of

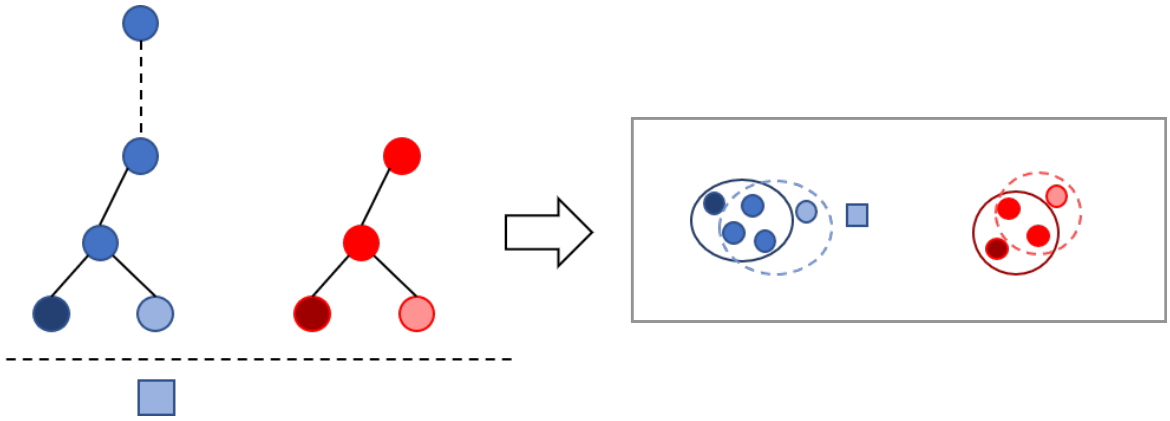


Figure 4.3 Illustration of the proposed appearance tracking model. Left: two trimmed trees in the forest shown in Fig. 4.2. Trees are distinguished by colors, one in blue and the other in red. Each tree contains the active track hypotheses of an object. Branches of a tree are distinguished by different saturations. Tree nodes depicted by circles are associated instances while the new observation to associate is depicted by a rectangle. Right: feature vectors on the embedding space of all the instances. Each hypothesis is represented by an instance cluster. Each object has its own independent instance set.

the angle between their feature vectors $\mathbf{y}_1$ and $\mathbf{y}_2$. Here, we recall the definition of cosine similarity, which has already been listed in Eq. (3.2), for clarity:

$$s_{cos} = \hat{\mathbf{y}}_1^T \hat{\mathbf{y}}_2 \tag{4.2}$$

$\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_1$ are unit vectors in the directions of $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively.

The similarity can be integrated in multiple object tracking frameworks after some transformation. For multicut approaches [102], the similarities can be either directly used as edge weights, or passed through a binary classifier. In this section, we only focus on the application of cosine similarity on common tracking approaches researching disjoint paths, e.g. multiple hypothesis tracking (MHT) [53]. In such approaches, the paths grow frame by frame. At a certain time, detections in the frame to process are compared to track hypotheses rather than independent observations. In the original MHT algorithm, the appearance of each track hypothesis is represented by a linear regression model trained on all examined detections. Our proposed appearance model is much simpler: each track hypothesis is seen as an instance cluster and represented by the center of its cluster. As the definition of tree suggests, the corresponding cluster of a hypothesis is unique. The cosine similarity between an unassociated detection and the cluster center summarizes its relationship to the corresponding hypothesis.

Before describing the details of our model, consider a generic clustering problem on a n-dimensional linear space. For an arbitrary cluster $\mathscr{C} = \{\mathbf{y}\}$, its cluster center is defined as the point $\mathbf{c}$ which has the shortest mean squared distance to all the instances in the cluster:

$$\mathbf{c} = \underset{\mathbf{v}\in\mathbb{R}^n}{\arg\min} \frac{1}{|\mathscr{C}|} \sum_{\mathbf{y}\in\mathscr{C}} d^2(\mathbf{y},\mathbf{v}) \tag{4.3}$$

where $|\mathscr{C}|$ is the cardinality of the cluster. The distance d is usually defined as Euclidean distance, and the solution of the equation above is as simple as shown below:

$$\mathbf{c} = \frac{1}{|\mathscr{C}|} \sum_{\mathbf{y}\in\mathscr{C}} \mathbf{y} \tag{4.4}$$

Eq. (4.4) says the center is the mean vector among the cluster.

In our appearance model for object tracking, we borrow the same idea of cluster center. The definition is revised under cosine similarity instead of squared Euclidean distance. The solution in Eq. (4.4) no longer fits in this situation. The main reason is that the magnitude of the mean vector $\mathbf{c}$ is not necessarily equal to 1. Unless all the vectors in the cluster are aligned, the mean vector become shorter as more instances are added into the track hypothesis. Consequently, a hypothesis may not be able to ingest new instances once it is too long and its mean vector has a length beneath the similarity threshold.

Therefore, we amend the problem by introducing a constraint that a cluster center should also be on the manifold of object feature vectors. Specifically, our learned embedding space for re-identification in Chapter 3 is a unit hypersphere, where all cluster centers should be found. The definition in Eq. (4.3) now can be rewritten as below:

$$\hat{\mathbf{c}} = \underset{\substack{\mathbf{v}\in\mathbb{R}^n \\ \text{subject to } \|\mathbf{v}\|=1}}{\arg\min} \frac{1}{|\mathscr{C}|} \sum_{\mathbf{y}\in\mathscr{C}} s_{cos}(\hat{\mathbf{y}},\mathbf{v}) \tag{4.5}$$

Since inner product of vectors is associative, the solution of Eq. (4.5) is

$$\begin{aligned} \hat{\mathbf{c}} &= \underset{\substack{\mathbf{v}\in\mathbb{R}^n \\ \text{subject to } \|\mathbf{v}\|=1}}{\arg\min} \left(\frac{1}{|\mathscr{C}|} \sum_{\mathbf{y}\in\mathscr{C}} \hat{\mathbf{y}} \right)^T \mathbf{v} \\ &= \underset{\substack{\mathbf{v}\in\mathbb{R}^n \\ \text{subject to } \|\mathbf{v}\|=1}}{\arg\min} \mathbf{c}^T\mathbf{v} \\ &= \frac{\mathbf{c}}{\|\mathbf{c}\|} \end{aligned} \tag{4.6}$$

where $\mathbf{c}$ is the mean vector defined in Eq. (4.4). By the proposed definition, the center of any cluster is exactly the direction vector of its mean, which is intuitive and easy to compute. In

our tracking algorithm, the new cluster center in Eq. (4.6) stands for the appearance model of a hypothesis.

In the meantime, the update of a model is merely absorbing the new observation into the cluster. Since only the inference and update of the models are only based on the cluster center, the unit mean vector $\hat{\mathbf{c}}$ and the cardinality $|\mathscr{C}|$ will suffice the maintenance of our appearance model. To conclude, our model is easy and fast to learn and update while demands very little memory.

4.3.2 Edge weights under softmax classification

In the original work [53], the weight of an edge depends only on the new observation and the hypothesis itself. Different hypotheses in the forest have no direct influence on each other in terms of edge weight determination. If an observation is suitable for multiple hypotheses, its final assignment is confirmed only via the competition among cliques in the MWIS problem. Analogous to [53], we consider the edge weights related to the likelihoods of the edges being positive, i.e. the to-be-associated observation has the same identity as the hypothetically tracked object. However, we believe that the likelihoods depend closely on all the hypotheses. A new detection belongs either to one of the tracked objects already modeled in the graph, or to a newly observed object. Thus, we explicitly model the competition over an instance among hypotheses as a classification problem and make the likelihoods visible in the edge weights.

This formulation is the same classification-guided re-identification as in Section 3.3. An observation $o \in \mathscr{O}^t$ in frame t is now compared with the instance clusters of all the hypotheses established $\mathscr{L}^{t-1}$ in the previous frame. Let $\hat{\mathbf{y}}$ denote the unit feature vector of observation $o \in \mathscr{O}^t$ after mapped onto the embedding space, and $\mathscr{C}$ be the cluster of hypothesis $l \in \mathscr{L}^{t-1}$. As defined in the previous subsection, their cosine similarity is

$$s_{cos}(l,o) = \hat{\mathbf{c}}^T \hat{\mathbf{y}} \tag{4.7}$$

where $\hat{\mathbf{c}}$ is the cluster center of $\mathscr{C}$ as defined in Eq. (4.6), and plays the same role of representing its cluster as the identity anchors proposed in Section 3.3.

Similar to the softmax classification used to guide the re-identification, the cosine similarity in Eq. (4.7) is the direct input of the classifier. It is very important to remark the fact that all the hypotheses in the forest $\mathscr{F}^{t-1}$ are not mutually independent. Hypotheses in $\mathscr{L}^{t-1}$ often share the same trunk while disperse with their own branches. However, a standard softmax classifier takes mutually exclusive events as entries. Ideally, each entry stands for the true positive trajectory of an independent object, rather than a bunch of hypotheses. For-

tunately, the MWIS problem of the graph is solved at every step to prune the tracking forest. Independent sets in the solution belong to mutually exclusive objects. Assume N independent sets are obtained from the forest of frame $t-1$, hypotheses $\mathscr{L}^{t-1}$ can then be correspondingly divided into N independent subsets: $\mathscr{L}^{t-1} = \bigcup_{i=1}^{N} \mathscr{L}_i$, and $\mathscr{L}_i \cap \mathscr{L}_{i'} = \emptyset, \forall i \neq i'$.

Each subset contains all the hypotheses of an object and is therefore an entry in the observation classifier. In our design, the association between an hypothesis subset and the new observation o is featured by its most similar hypothesis to o. The intuition is that non-maximum weighted paths are to be trimmed, and only the most likely hypothesis in the tree will be kept. Therefore, we choose the maximum similarity between o and hypotheses in the subset $\mathscr{L}_i$ as the optimal association at that moment, even though it might not be the globally optimal choice:

$$s_{cos}(\mathscr{L}_i, o) = \max_{l' \in \mathscr{L}_i} \{s_{cos}(l', o)\} \tag{4.8}$$

Now we can finally describe our modified softmax classifier. For a hypothesis $l \in \mathscr{L}^{t-1}$ and an observation $o \in \mathscr{O}^t$, we define the probability of their association being feasible based on their appearance similarity as below:

$$p(l,o) = \frac{\exp(\lambda s_{cos}(l,o))}{\exp(\lambda s_0) + \sum_{i=1}^{N} \exp(\lambda s_{cos}(\mathscr{L}_i, o))} \tag{4.9}$$

where λ is the same constant as in Eq. (3.8) used for classification-guided re-identification training and should be kept in consistence with the learned mapping of metric learning. Each object identity contributes a term in the denominator of Eq. (4.9). Unlike re-identification tasks discussed in Chapter 3, object identities in tracking tasks are not provided *a priori*. We include a term with a prechosen constant $s_0 < 1$ for newly appearing objects. When the observation o is not similar to any of the N tracked objects, our appearance model suggests the creation of a new track hypothesis tree. s_0 works as a soft similarity threshold and should be determined according to the learned embedding.

The classification decision is not instantly made in multiple hypothesis tracking. Instead, the softmax probabilities are stored as edge weights. The appearance-based weight ΔS_{app} of the edge (l,o) is defined as the log likelihood ratio between the probability in Eq. (4.9) and the null hypothesis, as in the original MHT algorithm [53]:

$$\Delta S_{app}(l,o) = \ln p(l,o) - \ln c_1 \tag{4.10}$$

where the constant c_1 is the same null hypothesis probability as in [53].

Our appearance model defined in Eq. (4.10) can be directly integrated into the MHT framework by replacing the old one. Along with its original motion model, the total edge weight ΔS in Eq. (4.1) determines whether the association represented by the edge is feasible. The full track score of a track hypothesis

$$S = \sum_{\tau=t_0}^{t} \Delta S^{\tau} \tag{4.11}$$

is the basis of track tree trimming and path selection, with t_0 indicating the frame where the track starts.

Overall, we only substitute the appearance model in MHT with our deep learning-based re-identification model. Experimental results show a significant improvement in terms of tracking accuracy compared to the original version. For more discussion on experiments, please refer to Section 4.5.

4.4 Online appearance-motion coupling

In the previous section, we introduced our appearance model. Besides the major contribution, we also investigated the data association strategies in multiple object tracking. An interesting observation is that state-of-the-art tracking algorithms have not well studied the relationship between different kinds of models. As we mentioned in Subsection 4.2.1, appearance and motion models or multiple models of the same type are mechanically combined by a naive weighted sum. The weights or coefficients of the models are determined offline and empirically. In this section, we prove that the collaboration of models is far from optimal by introducing a simple but effective online coupling between appearance and motion models.

Most data association approaches consist of two phases alternately staged at each step: an inference phase where the up-to-date tracking models examine all the potential associations when given a new frame, and an update phase where the models are adjusted according to the feasible associations selected in the inference phase. Appearance and motion models are closely interrelated in both phases and we propose an improvement for each. The modifications are based on our appearance model proposed above. In the meantime, the idea can be extended to other models, motion-based ones included.

4.4.1 Online model credibility for inference

During inference phase, each kind of tracking models may encounter difficulties of its own. For example, occlusions often cause missing or partial detection results, which may

Figure 4.4 Example of adjacent similar looking people in MOT16 benchmark [67]. They are difficult for appearance models to distinguish. Once their trajectories cross, motion models are more robust for preventing mismatches or trajectory merging.

further render motion models disabled. In this case, many tracking algorithms including MHT [53] use a dummy instance as placeholder. On the other hand, despite the remarkable progress made in appearance-based re-identification, similar-looking objects may still be indistinguishable, even by human experts. Motion models are indispensable in such situations. It is important to know when and where a tracking model is in trouble so that its questionable output can be flagged down. In this subsection, we propose a simple online detection method of appearance model failure along with its countermeasure.

In MHT approaches, an object has a list of possible tracks and only one hypothesis is finally kept. The objective of MHT is finding the optimal (maximum weight) path mainly based on the tracking score in Eq. (4.11). In other words, it explicitly requires object track tree to select from the candidate detections in each frame. Data association assures the uniqueness of object observation.

If looking from the other direction, a detection can only be observed in the path of one single object at most. The requirement of unique identity should also be satisfied. Nevertheless, the second requirement has not been well modeled and is implicitly achieved through concurrences among objects when a detection has more than one plausible assignment. It often happens when multiple similar-looking detections exist in the vicinity. Overlapping partial detections caused by occlusions may also be considered similar by appearance models. Such concurrences can induce tracking errors like mismatches and trajectory merging, especially when appearance models are given high credibility even if they are incapable of identity classification.

For this reason, we monitor online the credibility of the appearance model in our algorithm. When the appearance model has ambiguity in classification, it is considered less reliable and should contribute less to the combination of tracking scores. For an observation o in

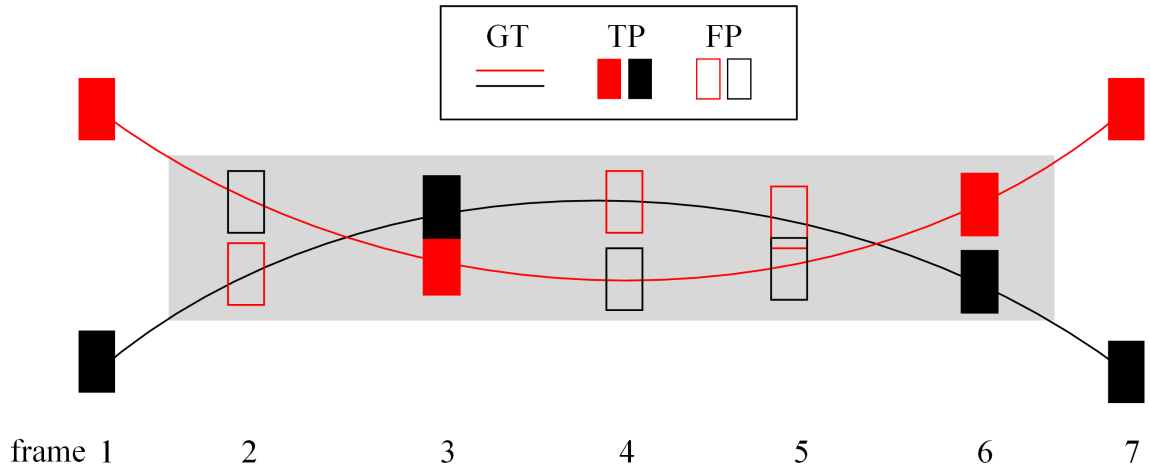


Figure 4.5 Illustration of hypotheses concurrences. It happens when black and red trackers have comparable scores (often caused by appearance model failures facing similar looking objects) for the same observation. The hypothesis-observation association can be unstable, which results in local mismatches and false positive trajectory parts. Such failures of the appearance model can be detected by own online credibility design and the motion model can provide robuster tracks, depicted by the solid lines. (In both Fig. 4.5 and 4.6, GT stands for ground truth, TP for True Positive, FP for False Positive and Occ. for Occlusion.)

frame t, the credibility c_{app} of its appearance-based classification is the greatest classification probability among all hypotheses:

$$c_{app}(o) = \max_{l \in \mathscr{L}^{t-1}} p(l,o) \tag{4.12}$$

It is designed to encourage a one-hot classification result, which implies no noticeable ambiguity. The more evenly matched concurrences among hypotheses exist, the more unreliable the appearance model is.

The combination between appearance and motion models is now established online. The credibility $c_{app} \in (0,1)$ of appearance model is introduced into Eq. (4.1) as a variable weight regularizer:

$$\Delta S = (2 - c_{app})\alpha_{mot}\Delta S_{mot} + c_{app}\alpha_{app}\Delta S_{app} \tag{4.13}$$

This new online model coupling is designed to relay the responsibility of decision making between models. With properly chosen coefficients α_{mot} and α_{app}, the appearance model can be more decisive if it is credible. Upon the failure of the appearance model, the motion model will step in for further arbitration.

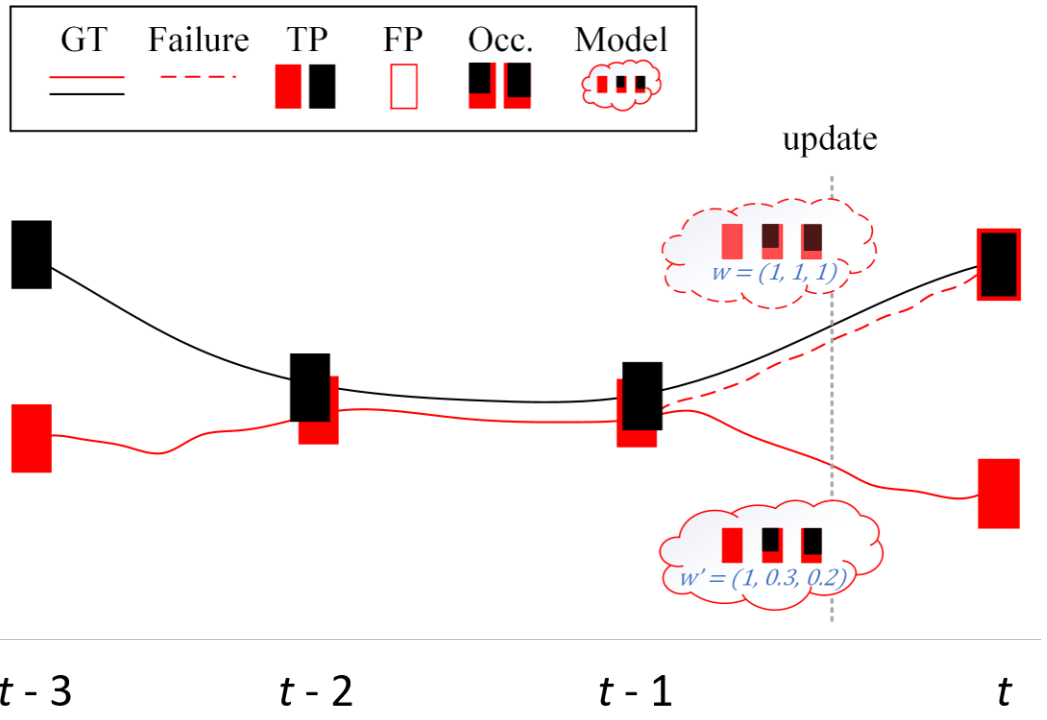


Figure 4.6 Tracker drift caused by noisy observations during model update. Partially occluded observations (illustrated by overlapping bounding boxes) may cause inconsistency of object appearance when given the same weight as benign instances (weight vector $\mathbf{w}$). That may lead to invalid appearance models and finally track fragments or even trajectory merging (see dotted red lines). We force an online noise pruning based on similarity scores, to minimize or eliminate the influence of outlier observations (the lower model with $\mathbf{w}'$)

4.4.2 Online model update

In most disjoint path selection approaches, their motion and appearance models are updated once observations in a new frame are associated. For example, a hypothesis in the original MHT algorithm [53] absorbs all the detections in the latest frame into its appearance model: the approved detection, if there exists one, is added into the positive set while the rest is treated as negative. The linear regressor of the hypothesis is then adjusted to fit the modification.

However, all to-be-absorbed observations are not always benign. Due to occlusions, especially those spanning long-duration and long distance, as well as abrupt object or/and camera movements, tracking algorithms often struggle to fill in the blanks of missing targets by introducing ambiguous observations into the track, under the guidance of motion models. Partially occluded, imprecise or even negative detections of poor appearance resemblances are accepted only because of their advantageous locations. On the other hand, observations are

universally given the identical weight in the model update process. This means false positive and false negative observations contribute as much as true positives and true negatives to the appearance model update. Without eliminating the noises, the model will be contaminated and even dysfunctional. The inconsistency in object appearance incurred by vulnerable model update strategies will then lead to mismatches, track fragments and trajectory merging (see Fig. 4.6).

Indeed, such failures are actually not rare in multiple object tracking tasks where occlusions are common. Noisy observations of the occluded object can be more similar to the overlying object in terms of appearance. After a long-term occlusion, trackers of the overlapping objects may comprise similar objects until the occluded tracker gradually drifts towards its occluder. Such tracker drift problems have been noticed for a long time [48, 21] but not been very well addressed. In this part, we propose a simple, intuitive but effective solution, based on our appearance model. The idea is to entrust our deep metric learning-based re-identification model with appearance-based noise suppression. Observations are properly weighted before getting absorbed into the appearance model so that noises have little influence on the model evolution.

Mathematically, at time t, multiple hypothesis tracking [53] manages to infer the association between remaining hypotheses $\mathscr{L}^{t-1}$ after pruning and the new observations $mathcal O^t$ in frame t. For any hypothesis $l^{t-1} \in \mathscr{L}^{t-1}$ starting from frame t_0, its appearance model is defined as the center $\hat{\mathbf{c}}^{t-1}$ of its observation cluster $\mathscr{C}^{t-1}$. By the former definition given in Subsection 4.3.2, the cluster $\mathscr{C}^{t-1}$ is a set or bag comprising the unit feature vectors $\{\mathbf{y}^\tau | \tau = t_0, \cdots, t-1\}$ of all the corresponding observations $\{o^\tau | \tau = t_0, \cdots, t-1\}$ absorbed into the hypothesis l frame by frame. All the elements are coequal: the set has no order or weights. Assuming an observation $o^t \in \mathscr{O}^t$ is chosen for the hypothesis l after the inference, the model simply incorporates the feature vector $\mathbf{y}^t$ of o^t into the cluster $\mathscr{C}^{t-1}$. We now have the updated model $\mathscr{C}^t = \{\mathbf{y}^\tau | \tau = t_0, \cdots, t\}$.

However, the inference of association is realized under the influence of appearance and motion models which may contradict each other sometimes. As a result of tradeoff between both kinds of tracking models, the inferred association in each frame should be scrutinized other than totally accepted during appearance model update. We use the similarity between the observation o and the hypothesis l to weigh the influence of their association on the appearance model update. The root of a new hypothesis tree has a unit weight. In any following frame, the weight is online determined using the cosine similarity. Accordingly, the center of the cluster defined in Eq. (4.5) and Eq. (4.6) should be reformulated as below:

$$\begin{aligned}\hat{\mathbf{c}}^* &= \underset{\substack{\mathbf{v}\in\mathbb{R}^n \\ \text{subject to } \|\mathbf{v}\|=1}}{\arg\min} \frac{1}{\sum_{\tau=t_0}^{t} w^\tau} \sum_{\tau=t_0}^{t} w^\tau s_{cos}(\hat{\mathbf{y}}^\tau, \mathbf{v}) \\ &= \frac{\sum_{\tau=t_0}^{t} w^\tau \hat{\mathbf{y}}^\tau}{\left\|\sum_{\tau=t_0}^{t} w^\tau \hat{\mathbf{y}}^\tau\right\|}\end{aligned} \tag{4.14}$$

where the weight w^τ is limited in $[0,1]$:

$$w^\tau = \begin{cases} 1, & \tau = t_0 \\ \max(0, s_{cos}(l^{\tau-1}, o^\tau)), & t_0 < \tau \leq t \end{cases} \tag{4.15}$$

and the cosine similarity s_{cos} is now based on the center $\hat{\mathbf{c}}^*$ of the hypothesis cluster at that time under the new definition in Eq. (4.14).

The update of the model $\hat{\mathbf{c}}^*$ and the inference based on it with cosine similarity are alternately performed as usual, except for the online determination of weights for associations. Outliers introduced by motion models now have less influence on object appearance models. Besides, pairwise similarities are already calculated during the inference phase and ready to be used for model update. The online update design has therefore no extra overhead. Experiments show a nontrivial improvement over the original update method in terms of tracking accuracy.

4.5 Experiments and discussion

Table 4.1 Ablation comparison of online model coupling strategies on the MOT16 benchmark

Method	MOTA	MOTP	MT	ML	FP	FN	IDsw	Frag
Baseline	44.3	76.0	14.9%	42.3%	9316	91331	883	850
Baseline+U	45.4	76.3	15.4%	43.2%	5735	92954	794	779
Baseline+U+C	**46.0**	76.3	15.5%	42.6%	5124	92697	693	759

Experiments are conducted on MOT16 benchmark [67] to demonstrate the effectiveness of our contributions. The re-identification model designed in Chapter 3 is pre-trained on CUHK03 detected training set [58] and then fine-tuned on MOT16 training set for tracking result evaluation. The proposed tracking algorithm inherits the exact implementation [2] of MHT [53] under MATLAB. The codes related to the appearance model are altered with

2. http://rehg.org/mht/

our design. The rest of the implementation including tracking hyperparameters irrelevant to the appearance model remains untouched so that we have an ablation comparison between different appearance models.

Our algorithm based on MHT surpasses not only the original method but also the best published algorithm (LMP). Here, we recall the evaluation metrics applied in most MOT benchmarks. Multiple Object Tracking Accuracy (MOTA) is the most important metric and focuses on all kinds of matching failures in all the frames. It is defined as below:

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{4.16}$$

where m_t, fp_t and mme_t denote the numbers of failures in frame t like misses (or false negatives, FN), false positives (FP), mismatches (or ID switches, IDsw), respectively, which are also often individually listed as evaluation metrics, while g_t is the number of groundtruths. A higher MOTA indicates a lower association failure ratio. Multiple Object Tracking Precision (MOTP) calculates the average localization precision of all the objects. Ratios of mostly tracked (MT) and lost (ML) objects are also recorded. The number of all fragments (Frag) is used to roughly evaluate the ability of recovering missing objects.

Table 4.2 Results on the MOT16 benchmark compared with leader algorithms

Method	MOTA	MOTP	MT	ML	FP	FN	IDsw	Frag
MHT_DAM v1 [53]	42.9 ± 8.9	76.6	13.6%	46.9%	5668	97919	499	659
MHT_DAM v2 [53]	45.8 ± 8.9	76.3	16.2%	43.2%	6412	91758	590	781
JMC [101]	46.3 ± 9.0	75.7	15.5%	39.7%	6373	90914	657	1114
LMP [102]	48.8 ± 9.8	79.0	18.2%	40.1%	6654	86245	481	595
Proposed	**49.4** ± 8.4	75.9	19.1%	39.4%	6281	85384	679	823

Also, we demonstrate the influence of our online model coupling strategies via an ablation comparison. Models are trained only on MOT16 training set, no other datasets involved.

The tracking result of the original method in [22] (without any online coupling) is set as baseline. A test only with online appearance update mechanism (Section 4.4) is then carried out (denoted as "baseline + U"). The last experiment includes both contributions (denoted as "baseline + U + C").

An interesting observation is that nearly all the tracking-by-detection approaches entirely split object detection from tracking. Detached from each other, the two modules cannot benefit from each other. We also designed a region-based tracking architecture that train the detection and the re-identification tasks together within a single neural network. Although

the current performance is not satisfactory enough due to the lack of training data, it remains a promising track for future improvement. The formulation of the region-based tracking will be detailed in Appendix B.

4.6 Conclusion

In this chapter, we described our novel formulation of data association. During tracking, objects detected in the upcoming frame or frames should be associated with established track hypotheses. Previous work considers this process as a one-way candidate seeking for hypotheses: each hypothesis chooses from the pool of detected objects its best fit by examining the appearance similarity and the geometric feasibility. However, the selection in the reverse direction, or the identity uniqueness of observations, is constantly ignored in the literature. When a detected object fits simultaneously multiple hypotheses of different identities, such an ambiguity of observation identity has not been well modeled before our work and is left to be solved via concurrences among hypotheses during decision making phase, whose results can be counterintuitive and ambiguous. In this thesis, we believe that a more natural and efficient way is to explicitly model this bidirectional selection by formulating data association as a multi-class classification problem rather than a bunch of binary classification problems. We adopt the per-identity classification-guided re-identification model in Chapter 3 as an appearance model for data association. The identity of each observation is determined via incorporating its classification entries among all the established identities into data association scores. The same concept of cluster center for identities is also inherited from Chapter 3 to calculate the similarity between an observation and a track hypothesis.

The cooperation between appearance and motion models in data association is another interesting topic that can significantly influence the tracking performance but has scarcely been studied. Each kind of models is vulnerable to a certain set of tracking issues. A mechanical combination without adjusting with the issues between both kinds of models can be dangerous. By simply exploiting intermediate outputs of our appearance model, we can automatically and rapidly detect data association model failures and apply an online adjustment. Experimental results show that tracking performance benefits from this online coupling between appearance and motion models. Our tracking algorithm obtained the best multiple object tracking results.

The algorithm proposed in this chapter still needs perfection and more validation on different benchmarks and data association frameworks. The appearance model is not accurate enough with a large margin to be filled. Meanwhile, a more robust re-identification algorithm can be applied in every frame as a better non maximum suppression technique. With

Figure 4.7 An example of tracking results of the sequence MOT-01. With the proposed online appearance and motion models coupling, targets can be recovered after a long period of disappearance. For instance, the person with ID=1 (red bounding box) is consistently tracked, even enduring an incessant occlusion of more than 60 frames.

mutually exclusive observations in each frame, the disadvantage of minimum cost disjoint paths approaches compared to multicut ones will be insignificant. Now with a reliable and fast algorithm based on disjoint paths searching, a robust online tracking algorithm becomes possible and promising.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

Multiple object tracking, i.e. simultaneously tracking multiple objects in the scene, is an important but challenging visual task. Objects need to be accurately detected from background and distinguished from each other to avoid erroneous trajectories. Since remarkable progress has been made in object detection field, "tracking-by-detection" approaches are widely adopted in multiple object tracking research. Objects in all frames are detected in advance and tracking reduces to an association problem: linking detections of the same object through frames into trajectories.

Most tracking algorithms employ both motion and appearance models for data association. For multiple object tracking problems where exist many objects of the same category, a fine-grained discriminant appearance model is paramount and indispensable. In this thesis, we propose an appearance-based re-identification model using deep similarity metric learning to deal with multiple object tracking in mono-camera videos. Two main contributions are reported in this dissertation:

First, we reviewed the state of the art of object re-identification, realizing intra-category recognition which is required in multiple object tracking. Specifically, we thoroughly investigated the application of pairwise metric learning in this field. A deep Siamese network is employed to learn a proper end-to-end mapping from input images to a discriminant embedding space. Different metric learning configurations using various metrics, loss functions, deep network structures, etc. are experimented and compared, in order to determine the best re-identification model for tracking. With an intuitive and simple classification design, the proposed model achieves satisfactory re-identification results, which are comparable to state-of-the-art approaches using triplet loss when evaluated on benchmarks like CUHK03.

Our approach is easy and fast to train and the learned embedding can be readily transferred onto the domain of tracking tasks.

Second, we integrated our proposed re-identification model in data association as appearance guidance for multiple object tracking. For each object to be tracked in a video, we establish an identity-related appearance model based on the learned embedding for re-identification. Similarities among detected object instances are exploited for identity classification, which determines the tracking result along with motion models. Besides, we also investigated the collaboration and interference between appearance and motion models. Contrary to most existing tracking algorithms that bind both kinds of models via a simple sum of their scores, we propose an online model coupling to further improve the tracking performance. When a model fails in front of ambiguous tracks, the other takes over the data association. Experiments on Multiple Object Tracking Challenge benchmark prove the effectiveness of our modifications, with a state-of-the-art tracking accuracy.

5.2 Perspectives

For future work, there are many tracks for the betterment of our multiple object tracking algorithm.

On the one hand, our re-identification model is far from perfect. A more thorough study on this subject can be conducted in the future. The models proposed in Chapter 3 can be upgraded and extended. Deep architectures can be optimized. Random erasing [122], a data augmentation technique that generates incomplete instances to simulate occluded objects, can be introduced in the training of our model along with other beneficial tricks. Recent popular re-identification benchmarks like CUHK03 [58] and Market-1501 [119] evaluate the algorithms by their ranking capabilities. However, pairwise verification accuracy is a more appropriate metric in the design of a re-identification model for multiple object tracking, which will be added in our evaluation system. Besides, the simplicity of verification threshold determination should also be considered in our future research.

On the other hand, the integration of appearance-based re-identification model in multiple object tracking remains an interesting research topic. More tracking frameworks including lifted multicut tracking [102] should be examined to validate the generalization capability of our appearance model. Its application in online tracking approaches are also meaningful as it may help our region-based tracking design discussed in Appendix B. The influence of hyperparameters should also be examined on MOT Challenge benchmark and other datasets. Since more accurate object detection algorithms like Faster R-CNN [84] have been set as

standard detectors in recent benchmarks, the hyperparameters and tracking strategies need to be revised.

Beyond the tracking-by-detection scheme with unrelated detection and tracking, the two tasks may benefit from each other. A single deep architecture that detects and re-identifies objects can be very interesting and meaningful. We already designed a region-based tracking model by appending a new re-identification stage after detection. The details are described it in Appendix B. Its implementation needs improvement for better tracking performance. Recurrent structures can be introduced into the architecture as well to train a motion-based model at the same time. The actual bottleneck lies in the lack of training data for identity classification in multiple object tracking benchmarks. Nonetheless, the region-based tracking is promising, and we are looking forward to achieving a breakthrough in future work.

Some cutting-edge domains may also provide promising ideas for improving the tracking performance. For example, attention learning may concretely help object re-identification. Recent re-identification algorithms try to employ auxiliary information to steer the attention of the model onto salient parts. Such attention-aware info is usually engineered by calibrating certain regions or parts. We think the attention networks may be a better and more elegant solution to learning "what to learn". Another potential research track is recurrent neural networks (RNN). There are some applications of RNN in MOT domain, but none of them is very successful. We tried to directly passing the feature maps across frames in RNN, which seems to be promising. We will revisit this direction in our future research. Meanwhile, the recognition of small objects remains a challenge not only for detection but also for tracking. They are hard to re-identify. Current motion or appearance models cannot prevent them from drifting. For multiple object tracking, we still need to find better solutions for occlusions of such small objects.

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

[2] Abdel-Hakim, A. E. and Farag, A. A. (2006). Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1978–1983. IEEE.

[3] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282.

[4] Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face recognition with local binary patterns. In *European conference on computer vision*, pages 469–481. Springer.

[5] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2037–2041.

[6] Andres, B., Fuksová, A., and Lange, J.-H. (2015). Lifting of multicuts. *CoRR, abs/1503.03791*, 3.

[7] Babenko, B., Yang, M.-H., and Belongie, S. (2009). Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE.

[8] Babenko, B., Yang, M.-H., and Belongie, S. (2011). Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632.

[9] Barkan, O., Weill, J., Wolf, L., and Aronowitz, H. (2013). Fast high dimensional vector multiplication face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1960–1967.

[10] Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720.

[11] Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE.

[12] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688.

[13] Butt, A. A. and Collins, R. T. (2013). Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1846–1853.

[14] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

[15] Carreira, J. and Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3241–3248. IEEE.

[16] Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135.

[17] Chen, W., Chen, X., Zhang, J., and Huang, K. (2017a). A multi-task deep network for person re-identification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[18] Chen, Y., Zhu, X., and Gong, S. (2017b). Person re-identification by deep learning multi-scale representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2590–2600.

[19] Chiba, N. and Nishizeki, T. (1985). Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223.

[20] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

[21] Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B., and Yu, N. (2017). Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4836–4845.

[22] Cuan, B., Idrissi, K., and Garcia, C. (2018). Deep siamese network for multiple object tracking. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE.

[23] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[24] Daugman, J. G. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169.

[25] Dehghan, A., Tian, Y., Torr, P. H., and Shah, M. (2015). Target identity-aware network flow for online multiple target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1146–1154.

[26] Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., and Fei-Fei, L. (2012). Imagenet large scale visual recognition competition 2012 (ilsvrc2012). *See net. org/challenges/LSVRC*.

[27] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

[28] Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model.

[29] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

[30] Garcia, C. and Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11):1408–1423.

[31] Geng, M., Wang, Y., Xiang, T., and Tian, Y. (2016). Deep transfer learning for person re-identification. *arXiv preprint arXiv:1611.05244*.

[32] Gheissari, N., Sebastian, T. B., and Hartley, R. (2006). Person reidentification using spatiotemporal appearance. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1528–1535. IEEE.

[33] Gilbert, A. and Bowden, R. (2006). Tracking objects across cameras by incrementally learning inter-camera colour calibration and patterns of activity. In *European conference on computer vision*, pages 125–136. Springer.

[34] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

[35] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[36] Girshick, R., Iandola, F., Darrell, T., and Malik, J. (2015). Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446.

[37] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[38] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[39] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[40] He, K. and Sun, J. (2015). Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360.

[41] He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer.

[42] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[43] Heikkilä, M., Pietikäinen, M., and Schmid, C. (2006). Description of interest regions with center-symmetric local binary patterns. In *Computer vision, graphics and image processing*, pages 58–69. Springer.

[44] Hermans, A., Beyer, L., and Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

[45] Hinton, G. (2012). Neural networks for machine learning coursera video lectures - geoffrey hinton.

[46] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[47] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[48] Hu, W., Li, W., Zhang, X., and Maybank, S. (2014). Single and multiple object tracking using a multi-feature joint sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 37(4):816–833.

[49] Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

[50] Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

[51] Javed, O., Shafique, K., and Shah, M. (2005). Appearance modeling for tracking in multiple non-overlapping cameras. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 26–33. IEEE.

[52] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

[53] Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704.

[54] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[55] Lafferty, J., McCallum, A., Pereira, F., et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

[56] Leal-Taixé, L., Canton-Ferrer, C., and Schindler, K. (2016). Learning by tracking: Siamese cnn for robust target association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40.

[57] Li, F., Kim, T., Humayun, A., Tsai, D., and Rehg, J. M. (2013). Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2192–2199.

[58] Li, W., Zhao, R., Xiao, T., and Wang, X. (2014). Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159.

[59] Li, W., Zhu, X., and Gong, S. (2017). Person re-identification by deep joint learning of multi-loss classification. *arXiv preprint arXiv:1705.04724*.

[60] Liao, S., Hu, Y., Zhu, X., and Li, S. Z. (2015). Person re-identification by local maximal occurrence representation and metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2197–2206.

[61] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

[62] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[63] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

[64] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[65] Lowe, D. G. (1985). Perceptual organization and visual recognition.

[66] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[67] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

[68] Milan, A., Roth, S., and Schindler, K. (2013). Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72.

[69] Minsky, M. and Papert, S. (1969). Perceptrons.

[70] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[71] Ng, A. (2013). Machine learning and ai via brain simulations.

[72] Nguyen, H. V. and Bai, L. (2010). Cosine similarity metric learning for face verification. In *Asian Conference on Computer Vision*, pages 709–720. Springer.

[73] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[74] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

[75] Perera, A. A., Srinivas, C., Hoogs, A., Brooksby, G., and Hu, W. (2006). Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 666–673. IEEE.

[76] Piccardi, M. (2004). Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE.

[77] Pirsiavash, H., Ramanan, D., and Fowlkes, C. C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208. IEEE.

[78] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., and Schiele, B. (2016). Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937.

[79] Prosser, B. J., Gong, S., and Xiang, T. (2008). Multi-camera matching using bi-directional cumulative brightness transfer functions. In *BMVC*, volume 8, page 74. Citeseer.

[80] Prosser, B. J., Zheng, W.-S., Gong, S., Xiang, T., and Mary, Q. (2010). Person re-identification by support vector ranking. In *BMVC*, volume 2, page 6.

[81] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[82] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.

[83] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

[84] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

[85] Ren, S., He, K., Girshick, R., Zhang, X., and Sun, J. (2017). Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481.

[86] Rippel, O., Paluri, M., Dollar, P., and Bourdev, L. (2015). Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*.

[87] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[88] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[89] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[90] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[91] Schwartz, W. R. and Davis, L. S. (2009). Learning discriminative appearance-based models using partial least squares. In *2009 XXII Brazilian symposium on computer graphics and image processing*, pages 322–329. IEEE.

[92] Segal, A. V. and Reid, I. (2013). Latent data association: Bayesian model selection for multi-target tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2904–2911.

[93] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., and Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126.

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

[95] Sirovich, L. and Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524.

[96] Sobel, I. and Feldman, G. (1968). A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272.

[97] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015a). Highway networks. *arXiv preprint arXiv:1505.00387*.

[98] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015b). Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385.

[99] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[100] Tang, S., Andres, B., Andriluka, M., and Schiele, B. (2015). Subgraph decomposition for multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5033–5041.

[101] Tang, S., Andres, B., Andriluka, M., and Schiele, B. (2016). Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision*, pages 100–111. Springer.

[102] Tang, S., Andriluka, M., Andres, B., and Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548.

[103] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

[104] Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558.

[105] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.

[106] Wang, X. (2013). Intelligent multi-camera video surveillance: A review. *Pattern recognition letters*, 34(1):3–19.

[107] Wang, X., Doretto, G., Sebastian, T., Rittscher, J., and Tu, P. (2007). Shape and appearance context modeling. In *2007 ieee 11th international conference on computer vision*, pages 1–8. Ieee.

[108] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244.

[109] Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. (2013). Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392.

[110] Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

[111] Wojke, N. and Bewley, A. (2018). Deep cosine metric learning for person re-identification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 748–756. IEEE.

[112] Yang, F., Choi, W., and Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2129–2137.

[113] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13.

[114] Zajdel, W., Zivkovic, Z., and Krose, B. (2005). Keeping track of humans: Have i seen this person before? In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2081–2086. IEEE.

[115] Zamir, A. R., Dehghan, A., and Shah, M. (2012). Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *European Conference on Computer Vision*, pages 343–356. Springer.

[116] Zhang, L., Li, Y., and Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[117] Zheng, L. (2016). *Triangular Similarity Metric Learning: a Siamese Architecture Approach*. PhD thesis, UNIVERSITE DE LYON.

[118] Zheng, L., Idrissi, K., Garcia, C., Duffner, S., and Baskurt, A. (2015a). Triangular similarity metric learning for face verification. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–7. IEEE.

[119] Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q. (2015b). Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1116–1124.

[120] Zheng, L., Yang, Y., and Hauptmann, A. G. (2016). Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984*.

[121] Zhong, Z., Zheng, L., Cao, D., and Li, S. (2017a). Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1318–1327.

[122] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017b). Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.

Appendix A

Binary operation for vectors with singleton expansion

We explain here the implementation detail of multidimensional matrix expansion for metric computation mentioned in Chapter 3. The expansion allows element-by-element binary operation between two sets $\mathscr{U}$ and $\mathscr{V}$. Any two elements $u \in \mathscr{U}$ and $v \in \mathscr{V}$ originated from different sets can form an ordered pair (u, v). The examination of all pairs involves a full element combination between the two sets, or their Cartesian product $\mathscr{U} \times \mathscr{V}$.

Consider the simplest situation with two sets of real scalars, i.e. $u, v \in \mathbb{R}$. We use two column vectors $\mathbf{u} = [u_1, u_2, \cdots, u_p]^T$ and $\mathbf{v} = [v_1, v_2, \cdots, u_q]^T$ to denote the sets $\mathscr{U}$ and $\mathscr{V}$, where p and q are their cardinalities. If the element-wise binary operation is defined as the scalar product, we obtain a tensor product:

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_q \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_q \\ \vdots & \vdots & \ddots & \vdots \\ u_p v_1 & u_p v_2 & \cdots & u_p v_q \end{bmatrix} \tag{A.1}$$

In most deep learning framework (MATLAB, Caffe [52], PyTorch [74], Tensorflow [1]), it can be realized with matrix multiplication.

On the hand hand, we can define various map functions $f : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$, which are not necessarily bilinear as the tensor product requires. A result analogous to Eq. (A.1) can be obtained by applying the binary operation on the Cartesian product $\mathscr{U} \times \mathscr{V}$:

$$\begin{bmatrix} f(u_1,v_1) & f(u_1,v_2) & \cdots & f(u_1,v_q) \\ f(u_2,v_1) & f(u_2,v_2) & \cdots & f(u_2,v_q) \\ \vdots & \vdots & \ddots & \vdots \\ f(u_p,v_1) & f(u_p,v_2) & \cdots & f(u_p,v_q) \end{bmatrix} \tag{A.2}$$

In MATLAB, it is achieved with an implicit singleton expansion which replicates the matrix along its singleton dimension (size of the dimension equals 1). However, the expansion style has not been well introduced into other deep learning frameworks.

Moreover, we are dealing with sets of n-dimensional feature vectors in the metric learning process described in Chapter 3. Given two sets $\mathscr{U}, \mathscr{V} \subset \mathbb{R}^n$, their elements can be denoted as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_p]^T$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_q]^T$, respectively. A scalar mapping function $f : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ is applied on the Cartesian product of the sets:

$$\begin{bmatrix} f(\mathbf{u}_1,\mathbf{v}_1) & f(\mathbf{u}_1,\mathbf{v}_2) & \cdots & f(\mathbf{u}_1,\mathbf{v}_q) \\ f(\mathbf{u}_2,\mathbf{v}_1) & f(\mathbf{u}_2,\mathbf{v}_2) & \cdots & f(\mathbf{u}_2,\mathbf{v}_q) \\ \vdots & \vdots & \ddots & \vdots \\ f(\mathbf{u}_p,\mathbf{v}_1) & f(\mathbf{u}_p,\mathbf{v}_2) & \cdots & f(\mathbf{u}_p,\mathbf{v}_q) \end{bmatrix} \tag{A.3}$$

In our algorithms, the function f is set as a metric function among the distances and similarities defined in Eq. (3.1) and Eq. (3.2). To accelerate the calculation of Eq. (A.3) via parallel computation on GPUs, a specific module conducting singleton expansion should be added to the deep learning frameworks. The "vectors" $\mathbf{U}$ and $\mathbf{V}$ are saved as multidimensional matrices in Caffe [52], we implemented a multidimensional matrix expansion suggested by Eq. (A.3).

During the training phase of pairwise loss-guided metric learning, $\mathscr{U}$ and $\mathscr{V}$ contain all the vectors in the mini-batches at each iteration. The training label is also a $p \times q$ matrix that can be calculated by Eq. (A.2). A binary sameness/comparison operation is applied on the ground-truth identities of object instances in the two sets. Usually, the sets are forced to be identical to reduce the feature extraction computation. The diagonals of the metric matrix and the label matrix are ignored under this circumstance. When the classification loss is employed (see Section 3.3), the first set $\mathscr{U}$ is loaded with the anchors of all the identities in the training dataset rather than instance feature vectors.

Appendix B

Region-based tracking

In all the tracking-by-detection algorithms discussed in Chapter 4, our proposed modified algorithms included, object detection is totally separated from the tracking phase. On the one hand, it is demanded by tracking benchmarks like Multiple Object Tracking Challenge [67] where detections are provided for all the algorithms. The idea is to evaluate only their tracking phases. On the other hand, given the different objectives of inter-category detection and intra-category re-identification, simultaneously solving both tasks may bring about difficulties. In this appendix, we provide a feasible multitask approach to combine the two phases together so that they benefit from each other.

As reviewed in Chapter 2, region-based approaches with the help of deep convolutional features dominates the field of object detection. Faster R-CNN [84], one of the most successful algorithms in the region-based detection family, is in fact realized by concatenating two classification networks [34]. The design is based on the same insight as in the cascade of weak classifiers [105]: search space of an image is so vast that literally exhaustive search is prohibitively time consuming and infeasible if every potential pose is scrutinized by a sophisticated classifier. A solution is using weak but simple classifiers to comb the search space, which is either sparsely down-sampled (top-down) or heuristically trimmed with low-level features (bottom-up). More sophisticated and powerful classifiers are then applied on the propositions of weak classifiers. Such hierarchical structure saves a lot of computation and time. The energy functions are also calculated hierarchically.

Specifically, the region proposal phase in Fast R-CNN [34] and Faster R-CNN [84] realizes a binary classification of objectness. This first subtask is conducted on the quantized search space $\mathscr{R}^{(1)}$. Each sampled region $r^{(1)}) \in \mathscr{R}^{(1)}$ (also called an "anchor" in [84]) has two attributes: an appearance-based feature vector $\mathbf{a}^{(1)}$ and a position vector $\mathbf{p}^{(1)}$ indicating its location. Usually, the position of a region is defined by the coordinates of the top-left point along with its height and width: $\mathbf{p}^{(1)} = [p_x^{(1)}, p_y^{(1)}, p_w^{(1)}, p_h^{(1)}]$. A classifier $C^{(1)}$ is trained

to examine all the regions in $\mathscr{R}^{(1)}$ and predicts whether they are roughly accurate object bounding boxes. If the answer for a region $r^{(1)}$ is positive, a regressor $R^{(1)}$ will output a refined location of this region. The negative regions are omitted in this phase. The subtask is therefore trained with an energy function that reflects both classification and regression aspects:

$$J^{(1)} = \frac{1}{\left|\mathscr{R}^{(1)}\right|} \sum_{r^{(1)} \in \mathscr{R}^{(1)}} \left[L_{cls}^{(1)}(r^{(1)}) + L_{loc}^{(1)}(r^{(1)}) \right] \tag{B.1}$$

where $\left|\mathscr{R}^{(1)}\right|$ denotes the cardinality of the set of anchors. During training, positions of the regions in $\mathscr{R}^{(1)}$ are compared to the bounding boxes of all the objects in the ground truth to obtain their overlapping ratios. For a region $r^{(1)}$, the ground-truth object having the highest overlapping ratio with it becomes its training objective for both objectness classification and location regression, unless the highest ratio is below a pre-chosen threshold, in which case the region is marked as negative. The binary label $\ell^{(1)\in\{0,1\}}$ (0 standing for negative and 1 for positive) and the desired location $\mathbf{p}^*$ are used to guide the training of the classifier and the regressor.

In Faster R-CNN [84], the region proposal network (RPN) employs a binary softmax cross entropy loss for appearance-based classification:

$$L_{cls}^{(1)}(r^{(1)}) = L_{Softmax}\left(C^{(1)}(\mathbf{a}^{(1)}),\ \ell^{(1)} \right) \tag{B.2}$$

A smooth L_1 loss [34]

$$L_{SmoothL_1}(p) = \begin{cases} 0.5|p|^2, & \text{if } |p| < 1 \\ |p| - 0.5, & \text{otherwise} \end{cases} \tag{B.3}$$

is applied on each of the independent elements of the location vector:

$$L_{SmoothL_1}(\mathbf{p}) = \sum_i L_{SmoothL_1}(p_i) \tag{B.4}$$

where p_i is the i-th element of the vector $\mathbf{p}$. The regression loss is then defined as below:

$$L_{loc}^{(1)}(r^{(1)}) = \begin{cases} L_{SmoothL_1}\left(R^{(1)}(\mathbf{a}^{(1)}, \mathbf{p}^{(1)}) - \mathbf{p}^* \right), & \text{if } \ell^{(1)} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{B.5}$$

By now, the first subtask, region proposal, is entirely modeled with the first-level energy function $J^{(1)}$.

The proposed regions are not necessarily perfect since the classifier and regressor in RPN are weak and light-weighted, as the hierarchical design requires. A Fast R-CNN [34] is thus appended right after the RPN. The second subtask is nearly the same as region proposal except for some minor differences. In terms of deep architecture, Fast R-CNN is much more complex and powerful than RPN. The search space $\mathscr{R}^{(2)}$ in this subtask is much smaller than the first one, thanks to the negative region omission in RPN. Only the refined positive regions are left for further examination. On the other hand, the classifier $C^{(2)}$ in the Fast R-CNN is category-aware. A region is classified to its exact category. The objectness label is replaced by a category label $\ell^{(2)} \in \{0, 1, \cdots, N\}$ with N categories and 0 here stands for non-object. Besides, the regressor $R^{(2)}$ is now category-specific. Each category has its own bounding box regressor which can be denoted with the category label. The second subtask is mathematically modeled as below:

$$J^{(2)} = \frac{1}{\left|\mathscr{R}^{(2)}\right|} \sum_{r^{(2)} \in \mathscr{R}^{(2)}} \left[L_{cls}^{(2)}(r^{(2)}) + L_{loc}^{(2)}(r^{(2)}) \right] \tag{B.6}$$

where

$$L_{cls}^{(2)}(r^{(2)}) = L_{Softmax}\left(C^{(2)}(\mathbf{a}^{(2)}),\ \ell^{(2)} \right) \tag{B.7}$$

and

$$L_{loc}^{(2)}(r^{(2)}) = \begin{cases} L_{SmoothL_1}\left(R_{\ell^{(2)}}^{(2)}(\mathbf{a}^{(2)}, \mathbf{p}^{(2)}) - \mathbf{p}^{*} \right), & \text{if } \ell^{(2)} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{B.8}$$

Some other losses can also be added into the energy function $J^{(2)}$. For example, Mask R-CNN [39] creates a new branch of instance segmentation.

With objects well detected and classified at category level, the remaining work for a good tracking model is the intra-category object re-identification. Compared to objectness and category, object identity is a third level of object recognition. For this reason, we subjoin a Siamese re-identification network using deep metric learning, which is introduced in Chapter 3, after the Fast R-CNN for each category. The current implementation of region-based detection approaches often takes a single input image. The structure with classification fashion in Section 3.3 is more suitable for the combination of subtasks since it demands no explicit instance contrast but use identity anchors as agents [1]. The energy function $J^{(3)}$ at this level consists of the re-identification/identity classification loss defined in Eq. (3.8).

1. A classic Siamese structure is also feasible in this concatenation, and we managed to implement it in a recurrent neural network style.

The object recognition required by multiple object tracking is hence factorized on three stages of different granularities. The three stages share the same feature map, as done in most region-based detection approaches. The hierarchy allows a gradually increasing complexity at different levels and saves computational resources. The training of the entire structure is modeled as a multitask optimization problem:

$$\min\left(J_1 + J_2 + J_3\right) \tag{B.9}$$

The experimental results of Mask R-CNN [39] suggests that different subtasks may benefit from each other in such a multitask design. With a proper data association algorithm, the learned recognition results can directly yield object trajectories, without splitting detection and tracking phases.

The current tracking performance with this region-based tracking design is not satisfactory enough. The probable cause may be the lack of training data, compared to ImageNet and Microsoft COCO where object detection is trained. More work on the refinement of structure design as well as data augmentation will be conducted in the future.

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : CUAN
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 12/09/2019

Prénoms : Bonan

TITRE : Deep Similarity Metric Learning for Multiple Object Tracking

NATURE : Doctorat

Numéro d'ordre : 2019LYSEI065

Ecole doctorale : INFORMATIQUE ET MATHEMATIQUES (ED512)

Spécialité : Informatique et applications

RESUME :
Le suivi d'objets multiples dans une scène est une tâche importante dans le domaine de la vision par ordinateur, et présente toujours de très nombreux verrous. Les objets doivent être détectés et distingués les uns des autres de manière continue et simultanée. Les approches «suivi par détection» sont largement utilisées, où la détection des objets est d'abord réalisée sur toutes les frames, puis le suivi est ramené à un problème d'association entre les détections d'un même objet et les trajectoires identifiées. La plupart des algorithmes de suivi associent des modèles de mouvement et des modèles d'apparence.

Dans cette thèse, nous proposons un modèle de ré-identification basé sur l'apparence et utilisant l'apprentissage de métrique de similarité. Nous faisons tout d'abord appel à un réseau siamois profond pour apprendre un maping de bout en bout, des images d'entrée vers un espace de caractéristiques où les objets sont mieux discriminés. De nombreuses configurations sont évaluées, afin d'en déduire celle offrant les meilleurs scores. Le modèle ainsi obtenu atteint des résultats de ré-identification satisfaisants comparables à l'état de l'art.

Ensuite, notre modèle est intégré dans un système de suivi d'objets multiples pour servir de guide d'apparence pour l'association des objets. Un modèle d'apparence est établi pour chaque objet détecté s'appuyant sur le modèle de ré-identification. Les similarités entre les objets détectés sont alors exploitées pour la classification. Par ailleurs, nous avons étudié la coopération et les interférences entre les modèles d'apparence et de mouvement dans le processus de suivi. Un couplage actif entre ces 2 modèles est proposé pour améliorer davantage les performances du suivi, et la contribution de chacun d'eux est estimée en continue. Les expérimentations menées dans le cadre du benchmark «Multiple Object Tracking Challenge» ont prouvé l'efficacité de nos propositions et donné de meilleurs résultats de suivi que l'état de l'art.

MOTS-CLÉS :

Suivi d'objets multiples, apprentissage métriques, ré-identification d'objet, apprentissage profond, architecture Siamese

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse : GARCIA, Christophe et IDRISSI, Khalid

Président de jury : Michel PAINDAVOINE

Composition du jury :

Thierry CHATEAU, Alice CAPLIER, Michel PAINDAVOINE, Christophe GARCIA, Khalid IDRISSI