Feature Learning for RGB-D Data

The University Of Sheffield.

Ziyun Cai

Department of Electronic and Electrical Engineering

University of Sheffield

This thesis is submitted for the degree of

Doctor of Philosophy

August 2017

Declaration

This thesis includes parts of the following papers. All these papers were primarily written by Ziyun Cai, as a result of the Ph.D. research. These works have been published or under review:

1. **Z. Cai**, L. Liu, M. Yu and L. Shao. "Latent Structure Preserving Hashing", British Machine Vision Conference, Swansea, UK, Sep. 2015.

2. **Z. Cai**, J. Han, L. Liu and L. Shao. "Rgb-d datasets using microsoft kinect or similar sensors: a survey", published in Multimedia Tools and Applications, pp. 1-43, 2016.

3. L. Shao, **Z. Cai**, L. Liu and K. Lu. "Performance Evaluation of Deep Feature Learning for RGB-D Image/Video Classification", published in Information Sciences, vol. 385, pp. 266-283, 2017.

4. **Z. Cai** and L. Shao. "RGB-D Data Fusion in Complex Space", accepted by IEEE International Conference on Image Processing, Beijing, China, Sep. 2017.

5. **Z. Cai**, Y. Long and L. Shao. "Classification Complexity Assessment For Hyper-parameter Optimization", submitted to IEEE Transactions on Image Processing.

6. **Z. Cai** and L. Shao. "RGB-D Scene Classification via Multi-modal Feature Learning", submitted to IEEE Transactions on Cybernetics.

7. **Z. Cai**, Y. Long and L. Shao. "Adaptive RGB Image Recognition by Visual-Depth Embedding", submitted to IEEE Transactions on Image Processing.

Ziyun Cai
August 2017

Acknowledgements

My deepest gratitude goes first and foremost to my supervisor, Prof. Ling Shao, for his constant encouragement and enthusiastic supervision. He offered a great opportunity to me to research with him, and lighted the path for me into the world of machine learning, pattern recognition and computer vision. I can not reach the current progress without his illuminating and consistent instruction. Meanwhile, I do appreciate for the encouragement from my second supervisor Dr. Wei Liu. His enthusiastic life attitude and working style encourage me a lot. Each regular contact meeting with him made me feel full of power for a further research.

I would like to thank my fellow colleagues: Dr. Simon Jones, Dr. Ruomei Yan, Dr. Di Wu, Dr. Fan Zhu, Dr. Li Liu, Dr. Feng Zheng, Dr. Mengyang Yu, Yang Long, Yawen Huang, Redzuan Bin Abdul Manap, Bo Dong, Shidong Wang, Chirine Riachy, Daniel Organisciak, Yuming Shen, Bingzhang Hu, Yi Zhou and Jie Li. Thank them for their help. In addition, they made a lot of contributions to our group. It is a good memory of working with them. Thanks to Dr. Li Liu, Dr. Fan Zhu and Dr. MengYang Yu for their great help when I just joined our group as a newcomer. I feel grateful to Yang Long and Dr. Feng Zheng who always discussed with me in these three years.

I would like to thank the external examiners Dr. Hubert Shum and Dr. Yonghuai Liu for reviewing my thesis. Their suggestions are invaluable for the improvement of this thesis. I would also like to thank the internal co-ordinator Dr. Xiaoli Chu for the arrangements of the oral examination.

I would like to thank my friends: Qi Hong, Baoling Zhang, Lukai Zheng, Hui Zheng, Shuaida Ji, Tian Feng, Zhizhong Yu, Qian Guo, Lao E, for their support and care. They all bring me joyful experience in my leisure time.

I specially thank Postgraduate Administrator Ms. Hilary J Levesley, PGR Director Dr. Thomas Walther and Head of Department Prof. Geraint W Jewell for their kindness and patience. They are always ready to help when I am in trouble.

Finally, I would like to thank my grandma and loving parents for their enduring love and endless support. They forgive my caprice and fault all the time. I am grateful to my girlfriend Xiao Sun for her love and encouragement all the time.

Abstract

RGB-D data has turned out to be a very useful representation for solving fundamental computer vision problems. It takes the advantages of the color images that provide appearance information of an object and also the depth image that is immune to the variations in color, illumination, rotation angle and scale. With the invention of the low-cost Microsoft Kinect sensor, which was initially used for gaming and later became a popular device for computer vision, high quality RGB-D data can be acquired easily. RGB-D image/video can facilitate a wide range of application areas, such as computer vision, robotics, construction and medical imaging. Furthermore, how to fuse RGB information and depth information is still a problem in computer vision. It is not enough to simply concatenate RGB data and depth data together. A new fusion method could better fuse RGB images and depth images. It still needs more powerful algorithms on this. In this thesis, to explore more advantages of RGB-D data, we use some popular RGB-D datasets for deep feature learning algorithms evaluation, hyper-parameter optimization, local multi-modal feature learning, RGB-D data fusion and recognizing RGB information from RGB-D images: i) With the success of Deep Neural Network in computer vision, deep features from fused RGB-D data can be proved to gain better results than RGB data only. However, different deep learning algorithms show different performance on different RGB-D datasets. Through large-scale experiments to comprehensively evaluate the performance of deep feature learning models for RGB-D image/video classification, we obtain the conclusion that RGB-D fusion methods using CNNs always outperform other selected methods (DBNs, SDAE and LSTM). On the other side, since LSTM can learn from experience to classify, process and predict time series, it achieved better performances than DBN and SDAE in video classification tasks. ii) Hyper-parameter optimization can help researchers quickly choose an initial set of hyper-parameters for a new coming classification task, thus reducing the number of trials in terms of hyper-parameter space. We present a simple and efficient framework for improving the efficiency and accuracy of hyper-parameter optimization by considering the classification complexity of a particular dataset. We verify this framework on three real-world RGB-D datasets. After the analysis of experiments, we confirm that our framework can provide deeper insights into the relationship between dataset classification tasks and hyperparameters optimization,

thus quickly choosing an accurate initial set of hyper-parameters for a new coming classification task. iii) We propose a new Convolutional Neural Networks (CNNs)-based local multi-modal feature learning framework for RGB-D scene classification. This method can effectively capture much of the local structure from the RGB-D scene images and automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities. Experiments are conducted on two popular datasets to thoroughly test the performance of our method, which show that our method with local multi-modal CNNs greatly outperforms state-of-the-art approaches. Our method has the potential to improve RGB-D scene understanding. Some extended evaluation shows that CNNs trained using a scene-centric dataset is able to achieve an improvement on scene benchmarks compared to a network trained using an object-centric dataset. iv) We propose a novel method for RGB-D data fusion. We project raw RGB-D data into a complex space and then jointly extract features from the fused RGB-D images. Besides three observations about the fusion methods, the experimental results also show that our method achieves competing performance against the classical SIFT. v) We propose a novel method called adaptive Visual-Depth Embedding (aVDE) which learns the compact shared latent space between two representations of labeled RGB and depth modalities in the source domain first. Then the shared latent space can help the transfer of the depth information to the unlabeled target dataset. At last, aVDE matches features and reweights instances jointly across the shared latent space and the projected target domain for an adaptive classifier. This method can utilize the additional depth information in the source domain and simultaneously reduce the domain mismatch between the source and target domains. On two real-world image datasets, the experimental results illustrate that the proposed method significantly outperforms the state-of-the-art methods.

Table of contents

List of figures

List of tables

Chapter 1

Introduction and Literature Review

1.1 Introduction

In the past decades, there has been abundant computer vision research based on RGB images [2] [243] [45]. However, RGB images usually only provide the appearance information of the objects in the scene. With this limited information provided by RGB images, it is extremely difficult, if not impossible, to solve certain problems such as the partition of the foreground and background having similar colors and textures. Additionally, the object appearance described by RGB images is not robust against common variations, such as illuminance change, which significantly impedes the usage of RGB based vision algorithms in realistic situations. While most researchers are struggling to design more sophisticated algorithms, another stream of the research turns to find a new type of representation that can better perceive the scene. RGB-D image/video is an emerging data representation that is able to help solve fundamental problems due to its complementary nature of the depth information and the visual (RGB) information. Meanwhile, it has been proved that combining RGB and depth information in high-level tasks (*e.g.*, image/video classification) can dramatically improve the classification accuracy [248] [247].

The core of the RGB-D image/video is the depth image, which is usually generated by a range sensor. Compared to a 2D intensity image, a range image is robust to the variations in color, illumination, rotation angle and scale [51]. Early range sensors (such as Konica Minolta Vivid 910, Faro Lidar scanner, Leica C10 and Optech ILRIS-LR) are expensive and difficult to use for researchers in a human environment. Therefore, there is not much follow-up research at that time. However, with the release of the **low-cost** 3D Microsoft Kinect sensor[1] on 4th November 2010, acquisition of RGB-D data becomes cheaper and

[1] http://support.xbox.com/en-US/browse/xbox-360/accessories/Kinect, Kinect for Xbox 360.

easier. Not surprisingly, the investigation of computer vision algorithms based on RGB-D data has attracted a lot of attention in the last few years.

RGB-D images/videos can facilitate a wide range of application areas, such as computer vision, robotics, construction and medical imaging. Since a lot of algorithms are proposed to solve the technological problems in these areas, an increasing number of RGB-D datasets have been created as well to verify the algorithms. The usage of publicly available RGB-D datasets is not only able to save time and resources for researchers, but also enables a fair comparison of different algorithms.

In this thesis, we firstly briefly review the background, hardware and software information about Microsoft Kinect, recent important RGB-D datasets, four deep learning models and the detailed dataset introduction used in our research in the following sections in this chapter. Then we solve the problems in RGB-D areas: performance of deep learning models in RGB-D datasets, hyper-parameter optimization, RGB-D data fusion and recognizing RGB information from RGB-D data in Chapter 2, 3, 4, 5 and 6. More specifically, the evaluation and developed new methods about RGB-D data for solving these five different problems are as follows:

Summary of Remaining Chapters

Chapter 2: Classification Performance of Deep Learning Models on RGB-D Image/Video Datasets. Since Deep Neural Networks for image/video classification have obtained much success in various computer vision applications and high-quality RGB-D data can be easily acquired and used to enhance computer vision algorithms [94], this chapter aims to investigate how deep learning can be employed for extracting and fusing features from RGB-D data. In this chapter, we choose four prevalent deep learning models (*i.e.*, Deep Belief Networks (DBNs), Stacked Denoising Auto-Encoders (SDAE), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) Neural Networks). Then we conduct extensive experiments on five popular RGB-D datasets including three image datasets and two video datasets. We then present a detailed analysis of the comparison between the learned feature representations from the four deep learning models. In addition, a few suggestions on how to adjust hyper-parameters for learning deep neural networks are made in this chapter. According to the extensive experimental results, we believe that this evaluation will provide insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion.

Chapter 3: Hyper-parameter Optimization via Classification Complexity Assessment. Following the work in Chapter 2, we observe that the performances of many machine learning methods vary significantly with different sets of hyper-parameters especially for com-

plex models, such as DBN, SDAE, CNNs and other deep learning models, which always have tens to hundreds of hyper-parameters. More specifically, achieving the best performance with many machine learning methods depends critically on model hyper-parameter optimization. However, this optimization, which requires strong expertise, is often a "black magic" especially on deep learning models. Currently, the widely used classic methods such as random search, grid search and manual search are computationally expensive and unpractical. They have to face the same challenge about how to choose the initial set of trials from random hyper-parameter permutation and combination. In this chapter, we present a simple and efficient framework for improving the efficiency and accuracy of hyper-parameter optimization by considering the classification complexity of a particular dataset. Through this framework, we can quickly choose an initial set of hyper-parameters that are suitable for a new classification task, thus reducing the number of trials in the hyper-parameter space. Results on six real-world datasets using three representative deep learning models demonstrate the effectiveness of our framework for hyper-parameter optimization.

Chapter 4: Feature Learning for RGB-D Scene Classification. Besides the performance evaluation and hyper-parameter optimization through the utilization of RGB-D datasets, a new Convolutional Neural Networks (CNNs)-based local multi-modal feature learning framework (LM-CNN) for RGB-D scene classification is also proposed. LM-CNN is different from most of the past deep learning methods which are proposed for RGB-D scene classification use global information and directly consider all pixels in the whole image for high-level tasks. Such past deep learning methods cannot hold much information about local feature distribution, and simply concatenate RGB and depth features without exploring the correlation and complementarity between raw RGB and depth images. From the human vision perspective, we recognize the category of one unknown scene mainly relying on the object-level information which includes appearance, texture, shape and depth of each object and the structural distribution of different objects. Based on this observation, constructing mid-level representations with discriminative object parts would generally be more attractive for scene analysis. In this chapter, our proposed LM-CNN for RGB-D scene classification can effectively capture much of the local structure from the RGB-D scene images and automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities. The experimental results on two popular datasets, *i.e.,* NYU v1 depth dataset and SUN RGB-D dataset, show that our method with local multi-modal CNNs outperforms state-of-the-art methods.

Chapter 5: RGB-D Data Fusion in Complex Space. Different from the method which automatically learns a fusion strategy for the object-level recognition step in Chapter 4, we project the RGB and depth data into a complex space and make the fusion strategy at the

initial stage. Most of the RGB-D fusion methods extract features from RGB data and depth data separately and then simply concatenate them or encode these two kinds of features. Such frameworks cannot explore the correlation between the RGB pixels and their corresponding depth pixels. In this chapter, motivated by the physical concept that range data correspond to the phase change and color information corresponds to the intensity, we propose a novel method for RGB-D data fusion. We first project raw RGB-D data into a complex space and then jointly extract features from the fused RGB-D images. The advantages of the proposed fusion method are verified from three aspects: mutual information and independence, feature distribution and Euclidean KS-distance to uniformity. Meanwhile, we modify the classical SIFT to complex-valued SIFT ($\mathbb{C}$-SIFT) to evaluate our fusion method. Besides, some traditional algorithms and deep learning models can also be generalized for this fusion method. Consequently, the correlated and individual parts of the RGB-D information in the new feature space are well combined. Experimental results on two widely used RGB-D scene datasets show that our proposed RGB-D fusion method can achieve competing performance against the classical fusion methods. Our fusion method is valuable for other researchers who are exploring better features.

Chapter 6: Recognizing RGB Information from RGB-D data. Though the feature learning and RGB-D fusion methods have been proposed in Chapter 4 and Chapter 5, we still need to explore how to recognize RGB images captured by conventional surveillance cameras through leveraging a set of labeled RGB-D data. Recognizing RGB Information from RGB-D data is a promising application, which significantly reduces the cost while can still retain high recognition rates [116] [37] [111]. However, existing methods still suffer from the domain shifting problem due to conventional surveillance cameras and depth sensors are using different mechanisms. In this chapter, we aim to simultaneously solve the above two challenges: 1) how to use the additional depth information in the source domain? 2) how to reduce the data distribution mismatch between the source and target domains? We propose a novel method called adaptive Visual-Depth Embedding (aVDE) which learns the compact shared latent space between two representations of labeled RGB and depth modalities in the source domain first. Then the shared latent space can help the transfer of the depth information to the unlabeled target dataset. At last, aVDE models two separate learning strategies for domain adaptation (feature matching and instance reweighting) in a unified optimization problem, which matches features and reweights instances jointly across the shared latent space and the projected target domain for an adaptive classifier. We test our method on two pairs of datasets for object recognition and scene classification, the results of which demonstrate the effectiveness of our proposed method.

Chapter 7: Conclusion and Future work. In this chapter, we briefly summary the contri-

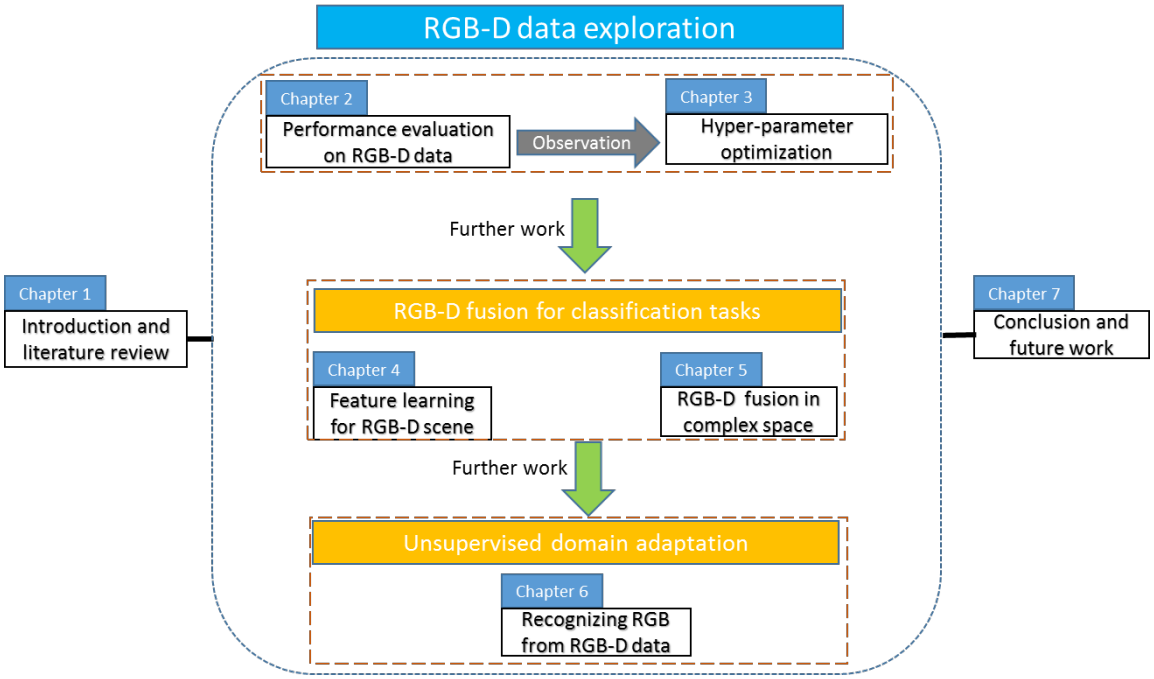


Fig. 1.1 The main studies and the correlation of all chapters carried out in this thesis. According to the RGB-D data exploration, all studies are explored further step by step: performance evaluation and hyper-parameter optimization (Chapters 2, 3), RGB-D fusion for classification tasks (Chapters 4, 5) and unsupervised domain adaptation on RGB-D data (Chapters 6).

butions of above work and discuss the future research directions.

For further explaining on the correlation between chapters and promoting a holistic understanding about this thesis, the main studies and correlation of all chapters are organized into Fig. 1.1.

1.2 A Brief Review of Kinect

In the past years, as a new type of scene representation, RGB-D data acquired by the consumer-level Kinect sensor or other similar sensors has shown the potential to solve challenging problems for computer vision. In this section, we select Kinect sensor as the core of the review. The reasons can be summarized as: 1) The hardware sensor as well as the software package of Kinect are released by Microsoft in November 2010, which makes Kinect sensor as the first released low-cost RGB-D sensor with powerful features. 2) Kinect sensor as the most representative RGB-D camera has a vast of sales until now. 3) Most RGB-D datasets are created in a time range from 2011 to 2017. The comparison of RGB-D dataset-

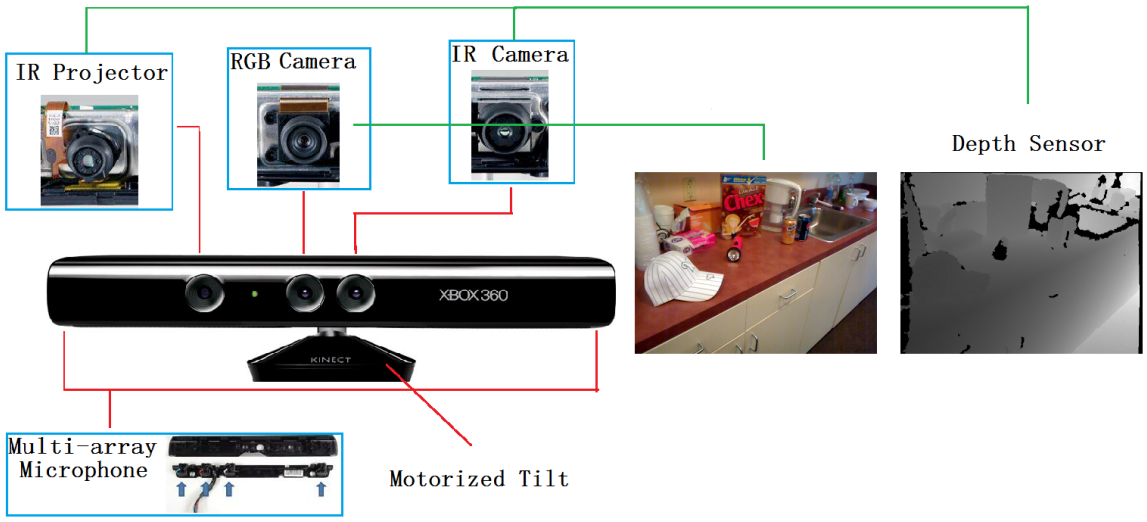


Fig. 1.2 Illustration of the structure and internal components of the Kinect sensor. Two example images from RGB and depth sensors are also displayed to show their differences.

s (from Table 1.4 to Table 1.6) shows that almost all of the datasets choose Kinect v1 or Kinect v2 as the related devices. Meanwhile, all of the selected datasets in our experiments are based on Kinect sensor.

At the beginning, Kinect acts as an Xbox accessory, enabling players to interact with the Xbox 360 through body language or voice instead of the usage of an intermediary device, such as a controller. Later on, due to its capability of providing accurate depth information with relatively low cost, the usage of Kinect goes beyond gaming, and is extended to the computer vision field. This device equipped with intelligent algorithms is contributing to various applications, such as 3D-simultaneous localization and mapping (SLAM) [109] [153], people tracking [190], object recognition [21] and human activity analysis [163] [38], etc. In this subsection, we introduce Kinect from two perspectives: hardware configuration and software tools.

1.2.1 Kinect Hardware Configuration

Generally, the basic version of Microsoft Kinect consists of a RGB camera, an infrared camera, an IR projector, multi-array microphone [138] and motorized tilt. Fig. 1.2 shows the components of Kinect and two example images captured by RGB and depth sensors, respectively. The distance between objects and the camera is ranging from 1.2 meters to 3.5 meters. Here, RGB camera is able to provide the image with the resolution of 640×480 pixels at 30 Hz. This RGB camera also has option to produce higher resolution images (1280×1024 pixels), running at 10 Hz . The angular field of view is 62 degrees horizontal-

ly and 48.6 degrees vertically. Kinect's 3D depth sensor (infrared camera and IR projector) can provide depth images with the resolution of 640×480 pixels at 30 Hz. The angular field of this sensor is slightly different with that of the RGB camera, which is 58.5 degrees horizontally and 46.6 degrees vertically. In the application such as **NUI** (Natural User Interface), multi-array microphone can be available for a live communication through acoustic source localization of Xbox 360. This microphone array actually consists of four microphones, and the channels of which can process up to 16-bit audio signals at a sample rate of 16 kHz. Following Microsoft, Asus launched Xtion Pro Live[2], which has more or less the same features with Kinect. In July 2014, Microsoft released the second generation Kinect: Kinect for windows v2[3]. The difference between Kinect v1 and Kinect v2 can be seen in Table 1.1.

Table 1.1 The difference between Kinect for windows v1 and Kinect for windows v2.

		Kinect for windows v1	Kinect for windows v2
Color	**Resolution**	640×480	1920×1080
	fps	30fps	30fps
Depth	**Resolution**	640×480	512×424
	fps	30fps	30fps
Sensor		Structured Light	Time of Flight
Range		$1.2 \sim 3.5m$	$0.5 \sim 4.5m$
Joint		20 joint / people	25 joint / people
Hand State		Open / closed	Open / closed / Lasso
Number of Apps		Single	Multiple
Body tracking		2 people	6 people
Body Index		6 people	6 people
Angle of View	**Horizontal**	62 degree	70 degree
	Vertical	48.6 degree	60 degree
Tilt Motor		Yes	No
Aspect Ratio		4:3	6:5
Supported OS		Win 7, Win 8	Win 8
USB Standard		2.0	3.0

In general, the technology used for generating the depth map is based on analyzing the speckle patterns of infrared laser light. The method is patented by PrimeSense [74]. For more detailed introductions, I refer to [78].

[2]http://www.asus.com, Asus Corporation, Xtion Pro Live

[3]http://www.xbox.com/en-GB/xbox-one/accessories/kinect, Microsoft Corporation, Kinect v2 for Xbox 360.

1.2.2 Kinect Software Tools

When Kinect is initially released for Xbox360, Microsoft actually did not deliver any SDKs. However, some other companies forecast an explosion in using Kinect and thus provide unofficial free libraries and SDKs. The representatives include CL NUI Platform[4], OpenKinect/Libfreenect[5], OpenNI[6] and PCL[7]. Although most of libraries provide basic algorithmic comments, such as camera calibration, automatic body calibration, skeletal tracking, facial tracking, 3-D scanning and so on, each library has its own characteristics. For example, CL NUI Platform developed by NUI researchers can obtain the data from RGB camera, depth sensor and *accelerometer*. Open Kinect focuses on providing free and open source libraries, enabling researchers to use Kinect over Linux, Mac and Windows. OpenNI is an industry-led open source library which can program RGB-D device for NUI applications. It is not specifically built for Kinect, and it can support multiple PrimeSense 3D sensors. Normally, users need to install SensorKinect, NITE, and OpenNI to control the Kinect sensor, where SensorKinect is the driver of Kinect and NITE is the middleware provided by PrimeSense. The latest version of OpenNI is version 2.2.0.33 until March 2017. The Point Cloud Library (PCL) is a standalone open source library which provides SLAM-related tools such as surface reconstruction, sample consensus, feature extraction, and visualization for RGB-D SLAM. It is licensed by Berkeley Software Distribution (BSD). More details and publications about PCL can be found in [204].

The official version of Kinect for Windows SDK[8] was released in July 2011, which provides a straightforward access to Kinect data: depth, color and disparity. The newest version is SDK 2.0. It can be applied for Windows 7, Windows 8, Windows 8.1 and Windows Embedded 8 with C++, C# or VB.NET. The development environment uses Visual Studio 2010 or higher versions. Regarding the software tool, it mainly contains skeletal tracking, higher depth fidelity, audio processing and so on.

The comparison of Kinect Windows SDK and unofficial SDK, *e.g.*, OpenNI, can be summarized below. The detailed same and difference between the Kinect Windows SDK and unofficial SDK can be seen in Table 1.2.

Kinect Windows SDK:

1) It supports audio signal processing and allows to adjust the motor angle.

2) It provides a full-body tracker including head, feet, hands and clavicles. Meanwhile, some details such as occluded joints are processed meticulously.

[4]http://codelaboratories.com/kb/nui, CL NUI Platform [Online].
[5]https://github.com/OpenKinect/libfreenect/, OpenKinect [Online].
[6]http://www.openni.ru/, OpenNI [Online].
[7]http://www.pointclouds.org/, PCL [Online].
[8]http://www.microsoft.com/en-us/kinectforwindows/, Microsoft Kinect SDK [Online].

3) Multiple sensors can be supported.

OpenNI/NITE library:
1) Commercial use of OpenNI is allowed.
2) Frameworks for hand tracking and hand-gesture recognition are included in OpenNI. Moreover, it automatically aligns the depth image and the color image.
3) It consumes less CPU power than that of Kinect Windows SDK.
4) It supports Windows, Linux and Mac OSX. In addition, streaming the raw Infrared video data becomes possible.

In conclusion, the most attractive advantage of OpenNI is the feasibility for multiple operational platforms. Besides it, using OpenNI is more convenient and can obtain better results for the research of colored point clouds. However, in terms of collection quality of the original image and the technology for pre-processing, Kinect for Windows SDK seems to be more stable. Moreover, Kinect for Windows SDK is more advantageous when requiring skeletal tracking and audio processing.

1.3 RGB-D Benchmark Datasets

Since the Kinect sensor was just released a few years ago, most RGB-D datasets are created in a time range from 2011 to 2017. Different from the traditional RGB datasets, RGB-D dataset not only has RGB data but also depth data. To have a clear structure, we divide the RGB-D datasets into 5 categories depending on the facilitated computer vision applications. More specifically, the reviewed datasets fall into object detection and tracking, human activity analysis, object and scene recognition, SLAM and hand gesture analysis. However, each dataset may not be limited to one specific application only. For example, object RGB-D can be used in detection as well. Table 1.3 illustrates a summary of these 5 categories.

1.3.1 RGB-D Datasets for Object Detection and Tracking

Object detection and tracking is one of the fundamental research topics in computer vision [233] [67]. It is an essential building-block of many intelligent systems. As we mentioned before, the depth information of an object is immune to changes of the object appearance or/and environmental illumination, and subtle movements of the background. With the availability of the low-cost Kinect depth camera, researchers immediately noticed that the feature descriptor based on depth information can help significantly detect and track the object in the real world where all kinds of variations occur. Therefore, RGB-D based object

Table 1.2 Comparison between the Kinect Windows SDK and unofficial SDK.

	Kinect Windows SDK	Unofficial SDK
Supported OS	Windows 7x86/x64	Windows XP/Vista/7x86/x64
	Windows 8, Windows 8.1 and Windows Embedded 8	Windows 8, Windows 8.1 and Windows Embedded 8
		Linux Ubuntu x86/x64
		Mac OS
		Android
Development language	C++, C#	C, C++, C#, Java
Commercial use	No	Yes
Supports for audio and motor/tilt	Yes	No
Supports multiple sensors	Yes	No
Consumption of CPU power	More	Less
Full body tracking	Includes head, hands, feet, clavicles	No head, hands, feet, clavicles
	Calculates positions for the joints, but not rotations	Calculates both positions and rotations for the joints
	Only tracks the full body, no hands only mode	Supports for hands only mode
Supports for Unity3D game engine	No	Yes
Supports for record/ playback to disk	No	Yes
Supports to stream the raw InfraRed video data	No	Yes

detection and tracking have attracted great attention in recent a few years. As a result, many datasets are created for evaluating proposed algorithms [232] [222].

1.3.2 Human Activity Analysis

The usage of RGB-D data opens up more opportunities to solve human activity analysis problems [161] [271]. Algorithms which combine RGB information and depth data can effectively increase the accuracy of activity recognition in a cluttered and illumination changed background.

1.3.3 Object and Scene Recognition

Object and scene recognition is a fundamental problem which aims to provide the information whether the image contains the object [296] [263]. In the real world environment,

Table 1.3 The summary of RGB-D dataset categories. It includes RGB-D People dataset [235], TUM Texture-Less dataset [97], object segmentation dataset (OSD) [202], object discovery dataset [176], Princeton tracking benchmark dataset (PTB) [232], Berkeley 3-D object dataset (B3DO) [119], NYU (New York University) depth V1 and V2 dataset [222], object dataset [144], Biwi head pose dataset [64], UR (University of Rzeszow) fall detection dataset [142], MSRDailyActivity3D (Microsoft research Activity3D) dataset [265], RGB-D person re-identification dataset [11], Kinect FaceDB [180], Big BIRD (Berkeley Instance Recognition dataset) [225], High Resolution Range based Face dataset (HRRFaceD) [172], TUM (University of Technology Munich) dataset [239], ICL-NUIM (Imperial College London and National University of Ireland Maynooth) dataset [95], Microsoft Research Cambridge-12 (MSRC-12) Kinect Gesture dataset [71], Sheffield Kinect Gesture (SKIG) dataset [163] and 50 Salads Dataset [238].

RGB-D benchmark datasets				
Object detection and tracking	***Human activity analysis***	***Object and scene recognition***	***SLAM***	***Hand gesture analysis***
People [235]	Biwi head pose [64]	Object [144]	TUM [239]	MSRC-12 Gesture [71]
TUM Texture-Less [97]	UR fall detection [142]	NYU depth V1 and V2 [222]	ICL-NUIM [95]	SKIG [163]
Object segmentation [202]	MSRDailyActivity3D [265]	B3DO [119]		50 Salads [238]
Object discovery [176]		Person re-identification [11]		
PTB [232]		Kinect FaceDB [180]		
B3DO [119]		Big BIRD [225]		
NYU depth V1 and V2 [222]		HRRFaceD [172]		
Object [144]				

recognizing objects from camera signal has to solve many problems, such as change of illumination, different levels of occlusion, textureless objects and reflection. One of the key challenges in RGB-D based object and scene recognition is how to formulate visual information (from RGB camera) and range information (from depth camera) to a fully descriptive and discriminative descriptor [286].

1.3.4 Simultaneous Localization and Mapping (SLAM)

The general problem of SLAM for both camera trajectory recovering and the map generation from sensor data attracts great attention from scientists in computer vision and robotics [288] [213]. Several datasets and benchmarks have been created for RGB-D based SLAM systems [239] [95]. However, how to provide an optimal up-to-date representation of the map in real-time is still a challenge.

1.3.5 Hand Gesture Analysis

In recent years, the research of hand gesture analysis from RGB-D sensors develops quickly, because it can facilitate a wide range of applications in human computer interaction, human robot interaction and pattern analysis [31] [40]. Compared to human activity analysis, hand gesture analysis does not need to deal with the dynamics from other body parts but only focuses on the hand region. On the one hand, the focus on the hand area only helps to increase the analysis accuracy. On the other hand, it also reduces the complexity of the system, thus enabling real-time applications. Basically, a hand gesture analysis system covers three components: hand detection and tracking, hand pose estimation and gesture classification. Since situations like occlusions, different illumination conditions and skin color affect the results of hand gesture analysis, improving the recognition accuracy of unconstrained human hand motions still needs a lot of efforts.

1.3.6 Comparison of RGB-D datasets

In this subsection, the comparison of RGB-D datasets is conducted from several aspects. For easy access, all the datasets are ordered alphabetically in three tables (from Table 1.4 to Table 1.6). If the dataset name starts with a digital number, it is ranked numerically following all the datasets which starts with English letters. For more comprehensive comparisons, besides these 20 mentioned datasets in Table 1.3, another 26 extra RGB-D datasets for different applications are also added into the tables: Birmingham University Objects, Category Modeling RGB-D [289], Cornell Activity [240] [132], Cornell RGB-

D [131], DGait [22], Daily Activities with occlusions [1], Heidelberg University Scenes [178], Microsoft 7-scenes [221], MobileRGBD [256], MPII Multi-Kinect [241], MSR Action3D Dataset [265], MSR 3D Online Action [284], MSRGesture3D [140], DAFT [86], Paper Kinect[197], RGBD-HuDaAct [188], Stanford Scene Object [127], Stanford 3D Scene [295], Sun3D [274], SUN RGB-D [231], TST Fall Detection [75], UTD-MHAD [36], Vienna University Technology Object [3], Willow Garage [269], Workout SU-10 exercise [186] and 3D-Mask [62]. In addition, we name those datasets without original names by means of creation place or applications. For example, I name the dataset in [178] as Heidelberg University Scenes.

Let us now explain these tables. The first and second columns in the tables are always the serial numbers and the names of the dataset. Table 1.4 shows some features including the authors of the datasets, the year of the creation, the published papers describing the dataset, the related devices, data size and number of references related to datasets. The author (the third column) and the year (the forth column) are collected directly in the datasets or are found in the oldest publication related to the dataset. The cited references in the fifth column contain the publications which elaborate the corresponding dataset. Data size (the seventh column) refers to the size of all information, such as the RGB and depth information, camera trajectory, ground truth and accelerometer data. For a scientific evaluation about these datasets, the comparison of number of citation is added into Table 1.4. A part of these statistical numbers are derived from the number of papers which use related dataset as benchmark. The rest are from the papers which do not directly use these datasets but mention these datasets in their published papers. It is noted that the numbers are roughly estimated. It can be easily seen from the table that the datasets with longer history [131] [239] [144] always have more related references than those of new datasets [289] [127]. Particularly, Cornell Activity, MSR Action3D Dataset, MSRDailyActivity3D, MSRGesture3D, Object RGB-D, People, RGBD-HuDaAct, TUM and NYU Depth V1 and V2 all have more than 100 citations. However, it does not necessarily mean that the old datasets are better than the new ones.

Table 1.5 presents the following information: the intended applications of the datasets, label information, data modalities and the number of the activities or objects or scenes along with the datasets. The intended applications (the third column) of the datasets are divided into five categories. However, each dataset may not be limited to one specific application only. For example, object RGB-D can be used in detection as well. The label information (the forth column) is valuable because it aids in the process of annotation. The data modalities (the fifth column) include color, depth, skeleton and accelerometer, which are helpful for researchers to quickly identify the datasets especially when they work on multi-modal

Table 1.4 The characteristics of the selected 46 RGB-D datasets.

No.	Name	Author	Year	Description	Device	Datasize	Number of citation
1	Big BIRD	Arjun Singh et al.	2014	[225]	Kinect v1 and DSLR	$\approx$ 74G	Unknown
2	Birmingham University Objects	Krzysztof Walas et al.	2014	No	Kinect v2	Unknown	Unknown
3	Biwi Head Pose	Fanelli et al.	2013	[64]	Kinect v1	5.6G	88
4	B3DO	Allison Janoch et al.	2011	[119]	Kinect v1	793M	96
5	Category Modeling RGB-D	Quanshi Zhang et al.	2013	[289]	Kinect v1	1.37G	4
6	Cornell Activity	Jaeyong Sung et al.	2011	[240]	Kinect v1	44G	>100
7	Cornell RGB-D	Abhishek Anand et al.	2011	[131]	Kinect v1	$\approx$ 7.6G	60
8	DAFT	David Gossow et al.	2012	[86]	Kinect v1	207M	2
9	Daily Activities with occlusions	Abdallah DIB et al.	2015	[1]	Kinect v1	6.2G	0
10	DGait	Ricard Borrs et al.	2012	[22]	Kinect v1	9.2G	7
11	Heidelberg University Scenes	Stephan Meister et al.	2012	[178]	Kinect v1	3.3G	24
12	HRRFaceD	Tomás Mantecón et al.	2014	[172]	Kinect v2	192M	Unknown
13	ICL-NUIM	A. Handa et al.	2014	[95]	Kinect v1	18.5G	3
14	Kinect FaceDB	Rui Min et al.	2012	[180]	Kinect v1	Unknown	1
15	Microsoft 7-scenes	Antonio Criminisi et al.	2013	[221]	Kinect v1	20.9G	10
16	MobileRGBD	Dominique Vaufreydaz et al.	2014	[256]	Kinect v2	Unknown	Unknown
17	MPII Multi-Kinect	Wandi Susanto et al.	2012	[241]	Kinect v1	15G	11
18	MSRC-12 Gesture	Simon Fothergill et al.	2012	[71]	Kinect v1	165M	83
19	MSR Action3D Dataset	Jiang Wang et al.	2012	[265]	Similar to Kinect	56.4M	>100
20	MSRDailyActivity3D	Zicheng Liu et al.	2012	[265]	Kinect v1	3.7M	>100
21	MSR 3D Online Action	Gang Yu et al.	2014	[284]	Kinect v1	5.5G	9
22	MSRGesture3D	Alexey Kurakin et al.	2012	[140]	Kinect v1	28M	94
23	NYU Depth V1 and V2	Nathan Silberman et al.	2011	[222]	Kinect v1	520G	>100
24	Object RGB-D	Kevin Lai et al.	2011	[144]	Kinect v1	84G	>100
25	Object Discovery	Julian Mason et al.	2012	[176]	Kinect v1	7.8G	8
26	Object Segmentation	A. Richtsfeld et al.	2012	[202]	Kinect v1	302M	28
27	Paper Kinect	F. Pomerleau et al.	2011	[197]	Kinect v1	2.6G	32
28	People	L. Spinello et al.	2011	[235]	Kinect v1	2.6G	>100
29	Person Re-identification	B. I. Barbosa, M et al.	2012	[11]	Kinect v1	Unknown	37
30	PTB	Shuran Song et al.	2013	[232]	Kinect v1	10.7G	12
31	RGBD-HuDaAct	Bingbing Ni et al.	2011	[188]	Kinect v1	Unknown	>100
32	SKIG	L. Liu et al.	2013	[163]	Kinect v1	1G	35
33	Stanford Scene Object	Andrej Karpathy et al.	2014	[127]	Xtion Pro Live	178.4M	29
34	Stanford 3D Scene	Qian-Yi Zhou et al.	2013	[295]	Xtion Pro Live	$\approx$ 33G	15
35	Sun3D	Jianxiong Xiao et al.	2013	[274]	Xtion Pro Live	Unknown	16
36	SUN RGB-D	S. Song et al.	2015	[231]	Kinect v1, Kinect v2, *etc*.	6.4G	8
37	TST Fall Detection	S. Gasparrini et al.	2015	[75]	Kinect v2	12.1G	25
38	TUM	J. Sturm et al.	2012	[239]	Kinect v1	50G	>100
39	TUM Texture-less	S Hinterstoisser et al.	2012	[97]	Kinect v1	3.61G	26
40	UR Fall Detection	Michal Kepski et al.	2014	[129]	Kinect v1	$\approx$ 5.75G	2
41	UTD-MHAD	Chen Chen et al.	2015	[36]	Kinect v1 and Kinect v2	$\approx$ 1.1G	3
42	Vienna University Technology Object	Aitor Aldoma et al.	2012	[3]	Kinect v1	81.4M	19
43	Willow Garage	Aitor Aldoma et al.	2011	[269]	Kinect v1	656M	Unknown
44	Workout SU-10 exercise	F Negin et al.	2013	[186]	Kinect v1	142G	13
45	3D-Mask	N Erdogmus et al.	2013	[62]	Kinect v1	Unknown	18
46	50 salads	Sebastian Stein et al.	2013	[238]	Kinect v1	Unknown	4

Table 1.5 The characteristics of the selected 46 RGB-D datasets.

No.	Name	Intended applications	Label information	Data modalities	Number of categories
1	Big BIRD	Object and scene recognition	Masks, ground truth poses, registered mesh	Color, depth	125 objects
2	Birmingham University Objects	Object detection and tracking	The model into the scene	Color, depth	10 to 30 objects
3	Biwi Head Pose	Human activity analysis	3D position and rotation	Color, depth	20 objects
4	B3DO	Object and scene recognition Object detection and tracking	Bounding box labeling at a class level	Color, depth	50 objects and 75 scenes
5	Category Modeling RGB-D	Object and scene recognition Object detection and tracking	Edge segments	Color, depth	900 objects and 264 scenes
6	Cornell Activity	Human activity analysis	Skeleton joint position and orientation on each frame	Color, depth skeleton	120+ activities
7	Cornell RGB-D	Object and scene recognition	Per-point object-level labeling	Color, depth, accelerometer	24 office scenes and 28 home scenes
8	DAFT	SLAM	Camera motion type, 2D homographies	Color, depth	Unknown
9	Daily Activities with occlusions	Human activity analysis	Position markers of the 3D joint location from a MoCap system	Color, depth, skeleton	Unknown
10	DGait	Human activity analysis	Subject, gender, age and an entire walk cycle	Color, depth	11 activities
11	Heidelberg University Scenes	SLAM	Frame-to-frame transformations and LiDAR ground truth	Color, depth	57 scenes
12	HRRFaceD	Object and scene recognition	No	Color, depth	22 subjects
13	ICL-NUIM	SLAM	Camera trajectories for each video. Geometry of the scene	Color, depth	2 scenes
14	Kinect FaceDB	Object and scene recognition	The position of six facial landmarks	Color, depth	52 objects
15	Microsoft 7-scenes	SLAM	6DOF ground truth	Color, depth	7 scenes
16	MobileRGBD	SLAM	speed and trajectory	Color, depth	1 scene
17	MPII Multi-Kinect	Object detection and tracking	Bounding box and polygons	Color, depth	10 objects and 33 scenes
18	MSRC-12 Gesture	Hand gesture analysis	Gesture, motion tracking of human joint locations	Color, depth, skeleton	12 gestures
19	MSR Action3D Dataset	Human activity analysis	Activity being performed and 20 joint locations of skeleton positions	Color, depth, skeleton	20 actions
20	MSRDailyActivity3D	Human activity analysis	Activity being performed and 20 joint locations of skeleton positions	Color, depth, skeleton	16 activities
21	MSR 3D Online Action	Human activity analysis	Activity in each video	Color, depth, skeleton	7 activities
22	MSRGesture3D	Hand gesture analysis	Gesture in each video	Color, depth	12 activities
23	NYU Depth V1 and V2	Object and scene recognition Object detection and tracking	Dense multi-class labeling	Color, depth, accelerometer	528 scenes
24	Object RGB-D	Object and scene recognition Object detection and tracking	Auto-generated masks	Color, depth	300 objects and scenes
25	Object Discovery	Object detection and tracking	Ground truth object segmentations	Color, depth	7 objects
26	Object Segmentation	Object detection and tracking	Per-pixel segmentation	Color, depth	6 categories
27	Paper Kinect	SLAM	6DOF ground truth	Color, depth	3 scenes
28	People	Object detection and tracking	Bounding box annotations and a 'visibility' measure	Color, depth	Multiple people
29	Person Re-identification	Object and scene recognition	Foreground masks, skeletons, 3D meshes and an estimate of the floor	Color, depth	79 people
30	PTB	Object detection and tracking	Bounding box covering target object	Color, depth	3 types and 6 scenes
31	RGBD-HuDaAct	Human activity analysis	Activities being performed in each sequence	Color, depth	12 activities
32	SKIG	Hand gesture analysis	The gesture is performed	Color, depth	10 gestures
33	Stanford Scene Object	Object detection and tracking	Ground truth binary labeling	Color, depth	58 scenes
34	Stanford 3D Scene	SLAM	Estimated camera pose	Color, depth	6 scenes
35	Sun3D	Object detection and tracking	Polygons of semantic class and instance labels	Color, depth	254 scenes
36	SUN RGB-D	Object and scene recognition Object detection and tracking	Dense semantic	Color, depth	19 scenes
37	TST Fall Detection	Human activity analysis	Activity performed, acceleration data and skeleton joint locations	Color, depth, skeleton, accelerometer	2 categories
38	TUM	SLAM	6DOF ground truth	Color, depth, accelerometer	2 scenes
39	TUM Texture-less	Object detection and tracking	6DOF pose	Color, depth	15 objects
40	UR Fall Detection	Human activity analysis	Accelerometer data	Color, depth, accelerometer	66 falls
41	UTD-MHAD	Human activity analysis	Accelerometer data with each video	Color, depth, skeleton, accelerometer	27 actions
42	Vienna University Technology Object	Object and scene recognition	6DOF GT of each object	Color, depth	35 objects
43	Willow Garage	Object detection and tracking	6DOF pose, per-pixel labelling	Color, depth	6 categories
44	Workout SU-10 exercise	Human activity analysis	Motion Files	Color, depth, skeleton	10 activities
45	3D-Mask	Object and scene recognition	Manually labeled eye positions	Color, depth	17 people
46	50 salads	Hand gesture analysis	Accelerometer data and labeling of steps in the recipes	Color, depth, accelerometer	27 people

Table 1.6 The characteristics of the selected 46 RGB-D datasets.

No.	Name	Camera move -ment	Multi -Sensors	Condi -tions required	Link
1	Big BIRD	Yes	Yes	Yes	http://rll.berkeley.edu/bigbird/
2	Birmingham University Objects	No	No	Yes	http://www.cs.bham.ac.uk/~walask/SHREC2015/
3	Biwi Head Pose	No	No	No	https://data.vision.ee.ethz.ch/cvl/gfanelli/head_pose/head_forest.html#
4	B3DO	No	No	No	http://kinectdata.com/
5	Category Modeling RGB-D	No	No	No	http://sdrv.ms/Z4px7u
6	Cornell Activity	No	No	No	http://pr.cs.cornell.edu/humanactivities/data.php
7	Cornell RGB-D	Yes	No	No	http://pr.cs.cornell.edu/sceneunderstanding/data/data.php
8	DAFT	Yes	No	No	http://ias.cs.tum.edu/people/gossow/rgbd
9	Daily Activities with Occlusions	No	No	NO	https://team.inria.fr/larsen/software/datasets/
10	DGait	No	No	No	http://www.cvc.uab.es/DGaitDB/Download.html
11	Heidelberg University Scenes	No	No	Yes	http://hci.iwr.uni-heidelberg.de//Benchmarks/ document/kinectFusionCapture/
12	HRRFaceD	No	No	No	https://sites.google.com/site/hrrfaced/
13	ICL-NUIM	Yes	No	No	http://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html
14	Kinect FaceDB	No	No	Yes	http://rgb-d.eurecom.fr/
15	Microsoft 7-scenes	Yes	No	Yes	http://research.microsoft.com/en-us/projects/7-scenes/
16	MobileRGBD	Yes	No	Yes	http://mobilergbd.inrialpes.fr//#RobotView
17	MPII Multi-Kinect	No	Yes	No	https://www.mpi-inf.mpg.de/departments/computer-vision- and-multimodal-computing/research/object-recognition-and-scene- understanding/mpii-multi-kinect-dataset/
18	MSRC-12 Gesture	No	No	No	http://research.microsoft.com/en-us/um/cambridge/projects/msrc12/
19	MSR Action3D Dataset	No	No	No	http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/
20	MSRDailyActivity3D	No	No	No	http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/
21	MSR 3D Online Action	No	No	No	http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/
22	MSRGesture3D	No	No	No	http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/
23	NYU Depth V1 and V2	Yes	No	No	http://cs.nyu.edu/~silberman/datasets/nyu_depth_v1.html http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html
24	Object RGB-D	No	No	No	http://rgbd-dataset.cs.washington.edu/
25	Object Discovery	Yes	No	No	http://wiki.ros.org/Papers/IROS2012_Mason_Marthi_Parr
26	Object Segmentation	No	No	No	http://www.acin.tuwien.ac.at/?id=289
27	Paper Kinect	Yes	No	No	http://projects.asl.ethz.ch/datasets/doku.php?id=Kinect:iros2011Kinect
28	People	No	Yes	No	http://www2.informatik.uni-freiburg.de/~spinello/RGBD-dataset.html
29	Person Re-identification	No	No	Yes	http://www.iit.it/en/datasets-and-code/datasets/rgbdid.html
30	PTB	Yes	No	No	http://tracking.cs.princeton.edu/dataset.html
31	RGBD-HuDaAct	No	No	Yes	http://adsc.illinois.edu/sites/default/files/files/ADSC-RGBD- dataset-download-instructions.pdf
32	SKIG	No	No	No	http://lshao.staff.shef.ac.uk/data/SheffieldKinectGesture.htm
33	Stanford Scene Object	NO	No	No	http://cs.stanford.edu/people/karpathy/discovery/
34	Stanford 3D Scene	Yes	No	No	https://drive.google.com/ folderview?id=0B6qjzcYetERgaW5zRWtZc2FuRDg&usp=sharing
35	Sun3D	Yes	No	No	http://sun3d.cs.princeton.edu/
36	SUN RGB-D	No	No	No	http://rgbd.cs.princeton.edu
37	TST Fall Detection	No	Yes	No	http://www.tlc.dii.univpm.it/blog/databases4kinect
38	TUM	Yes	Yes	No	http://vision.in.tum.de/data/datasets/rgbd-dataset
39	TUM Texture-less	No	No	No	http://campar.in.tum.de/Main/StefanHinterstoisser
40	UR Fall Detection	No	Yes	No	http://fenix.univ.rzeszow.pl/~mkepski/ds/uf.html
41	UTD-MHAD	No	No	No	http://www.utdallas.edu/~kehtar/UTD-MHAD.html
42	Vienna University Technology Object	No	No	No	http://users.acin.tuwien.ac.at/aaldoma/datasets/ECCV.zip
43	Willow Garage	No	No	No	http://www.acin.tuwien.ac.at/forschung/v4r/mitarbeiterprojekte/willow/
44	Workout SU-10 exercise	No	No	Yes	http://vpa.sabanciuniv.edu.tr/phpBB2/vpa_views.php?s=31&serial=36
45	3D-Mask	NO	NO	Yes	https://www.idiap.ch/dataset/3dmad
46	50 salads	No	No	Yes	http://cvip.computing.dundee.ac.uk/datasets/foodpreparation/50salads/

fusion [162] [34] [35]. Accelerometer data is able to indicate the potential impact of the object and starts an analysis of depth information, at the same time, it simplifies the complexity of the motion feature and increases its reliability. The number of the activities or objects or scenes is connected closely with the intended application. For example, if the application is SLAM, we focus on the number of the scenes in the dataset.

Table 1.6 concludes the information, such as whether the sensor moves during the collection process, whether it enables multi-sensor or not, whether it is download restricted, and the web link to the dataset. Camera movement is another important information when the algorithm selects the datasets for its evaluation. The rule in these tables is as follows: if the camera is still all the time in the collection procedure, it is marked "No", otherwise "Yes". The fifth column is related to the license agreement requirement. Most of the datasets can be downloaded directly from the web. However, downloading data from some datasets may need to fill in a request form. Moreover, few datasets are not public. The link to each dataset is also provided which can better help the researchers in related research areas.

1.4 Deep Learning Models

Many successful deep learning methods [24, 100, 146, 257] as efficient feature learning tools have been applied in a large amount of research. The aim of deep nets is to learn high-level features at each layer from the features in low-level. Some methods like (DBNs [100] and SDAE [257]) have something in common: they have two steps in the training procedure. One is unsupervised pre-training and the other is fine-tuning. In the first step, through an unsupervised algorithm, the weights of the network are able to be better than random initialization. This phase can avoid local minimum when doing supervised gradient descent. Therefore, we can consider that unsupervised pre-training is a regularizer. In the fine-tuning step, criterion (the prediction error which uses the labels in a supervised task) is minimized. These two approaches for learning deep networks are shown to be essential to train deep networks. Other methods like CNNs [136] contain more connections than weights. The model itself realizes a form of regularization. The aim of this kind of neural networks is to learn filters, in a data-driven fashion, as a tool to extract features describing inputs. This is not only used in 2D convolution but also can be extended into 3D-CNNs [122]. Up to date, though various successful deep learning methods are proposed in the field of computer vision, we can observe that these methods can be divided into four categories according to the basic methods they are derived from: Restricted Boltzmann Machines (RBMs) [101], CNNs, Auto-Encoder and Recurrent Neural Networks (RNNs). Some representative works in the categorization of deep learning methods can be found in Table 1.7. In the following

subsections, we will briefly introduce four representative classic deep learning methods in the four categories (DBNs, SDAE, CNNs and Long Short-Term Memory (LSTM) Neural Networks). In addition, the selected four introduced classic deep learning methods are used for the classification performance evaluation on RGB-D Image/Video datasets in Chapter 2.

Table 1.7 A categorization of the deep learning methods which includes RBM-based methods, Auto-Encoder-based methods, CNN-based methods and RNN-based methods, and their representative works.

Deep Learning Methods			
RBM-based Methods	**Auto-Encoder-based Methods**	**CNN-based Methods**	**RNN-based Methods**
Deep Belief Networks [100]	Stacked Denoising Auto-Encoders [257]	AlexNet [137]	Long Short-Term Memory [104]
Deep Boltzmann Machines [206]	Sparse Auto-Encoders [200]	Clarifai [287]	Bidirectional RNNs [212]
Deep Energy Models [187]	Contractive Auto-Encoders [203]	SPP [193]	Deep RNNs [87]
		VGG [224]	Echo State Networks [118]
		GoogLeNet [244]	Gated Recurrent Unit Recurrent Neural Networks [44]
			Clockwork RNNs [133]

1.4.1 Deep Belief Networks

As a neural network, Deep Belief Networks (DBNs) stack and train many layers of unsupervised Restricted Boltzmann Machines (RBMs) in a greedy manner which is first introduced in [100]. A deep hierarchical representation of the training data can be extracted by DBNs. DBNs consist of visible layers vector $\mathbf{x}$ and the ℓ hidden layers h^k. Each neuron on the layers is fully connected to all the neurons on the next layer. Through an unsupervised algorithm, the learned weights which are better than random initialization weights are used to initialize a multi-layer neural network and then adjusted to the current task through supervised information for classification. The model of the joint distribution between visible vector $\mathbf{x}$ and hidden layers h^k can be expressed:

$$P(x,h^1,\ldots,h^\ell) = \left(\prod_{k=0}^{\ell-2} P\left(h^k \mid h^{k+1}\right)\right) P\left(h^{\ell-1},h^\ell\right),$$

where x is denoted as h^0. $P\left(h^k \mid h^{k+1}\right)$ is the conditional distribution about the visible units conditioned on the hidden units of the RBM at level $k+1$. $P\left(h^{\ell-1},h^\ell\right)$ is the joint distribution over the visible units and hidden units on the top-level RBM. A schematic representation can be found in Fig. 1.3.

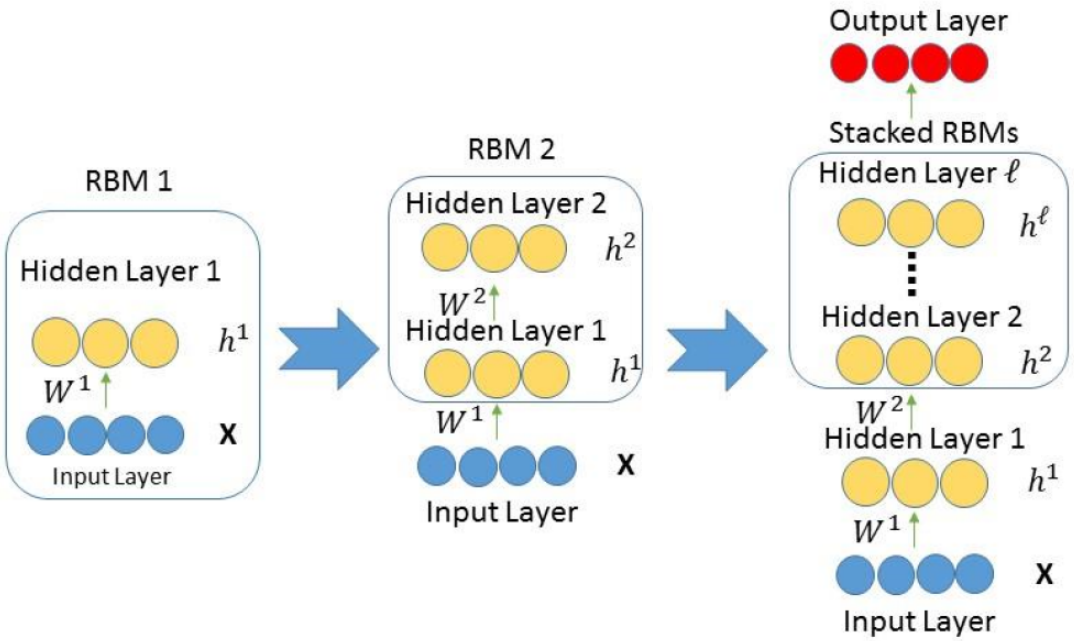


Fig. 1.3 The schematic representation of DBN. It is just an example about DBN structure. In practice, the number of units on each hidden layer is flexible.

The capability of "learning features" in a "layer-by-layer" manner is the greatest advantage of DBNs. From the previous layers, the higher-level features can be learned. These features are believed to be more complicated and can better reflect the information which is contained in the structures of input data. Another advantage of DBNs is that it learns the generative model without imposing subjective selection of filters. Factored RBM is able to learn the filters while learning the feature activities in an unsupervised learning manner. It solves the concern of the legality of the selected filters. Meanwhile, it shows the biological implementation of visual cortex, namely, the receptive fields for cells in the primary visual cortex. However, a well-performing DBN requires a lot of empirically decided hyper-parameter settings, *e.g.*, learning rate, momentum, weight cost number of epochs and number of layers. Inadequate selection of hyper-parameters will result in over-fitting and blow up DBNs. The property of DBNs that is sensitive to the empirically selected parameters has also been proved in our experiments. An improper set of hyper-parameters results in a huge difference from the best performance. To some extent, this disadvantage compromises the potential of DBNs.

DBNs have been applied for generating and recognizing images [14], video sequences [242], motion-capture data [249] and natural language understanding [207].

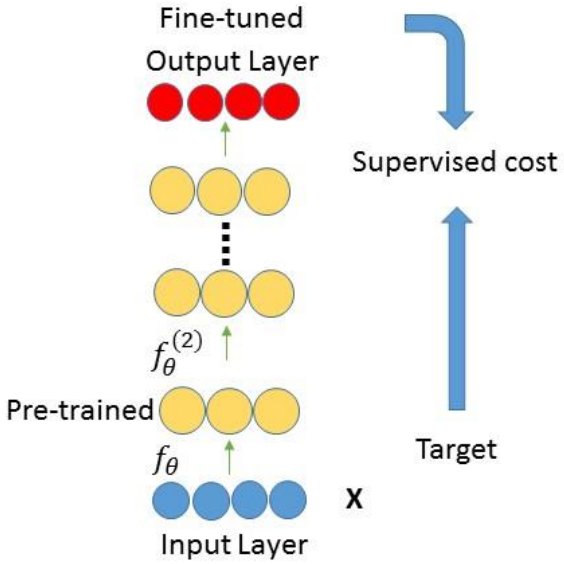


Fig. 1.4 The figure of Stacked Denoising Auto-Encoders includes unsupervised pre-training steps and supervised fine-tuning steps. Through performing gradient descent on a supervised cost, the parameters are fine-tuned to minimize the error with the supervised target.

1.4.2 Stacked Denoising Auto-Encoders

The second model is Stacked Denoising Auto-Encoders (SDAE) [257] which is an extension of Stacked auto-encoder [145]. This model has something in common with DBNs: they have two steps in the training procedure. One is unsupervised pre-training and the other is fine-tuning. SDAE also uses the greedy principle, but stacks denoising auto-encoders to initialize the deep network. An auto-encoder contains an encoder $h(\cdot)$ and a decoder $g(\cdot)$. Therefore, the reconstruction of the input x can be expressed as $Re(x) = g(h(x))$. The reconstruction accuracy is able to be obtained from minimizing the average reconstruction error $loss(x, Re(x))$. The figure of Stacked Denoising Auto-Encoders is shown in Fig. 1.4.

SDAE makes use of different kinds of encoders to transform the input data, which can preserve a maximization of the mutual information between the original and the encoded information. Meanwhile, it utilizes a noise criterion for minimizing the transformation error. We mentioned that DBNs and SDAE have something in common: they have two steps in the training procedure - one is unsupervised pre-training and the other is fine-tuning. The advantage of using auto-encoders as unsupervised building block of the deep architecture is that the training criterion is continuous in the parameters, almost any parametrization of the layers is possible [13]. However, in SDAE, training with gradient descent is slow and hard to parallelize. The optimization of SDAE is inherently non-convex and dependent on its initialization. Besides, since SDAE does not correspond to a generative model, unlike

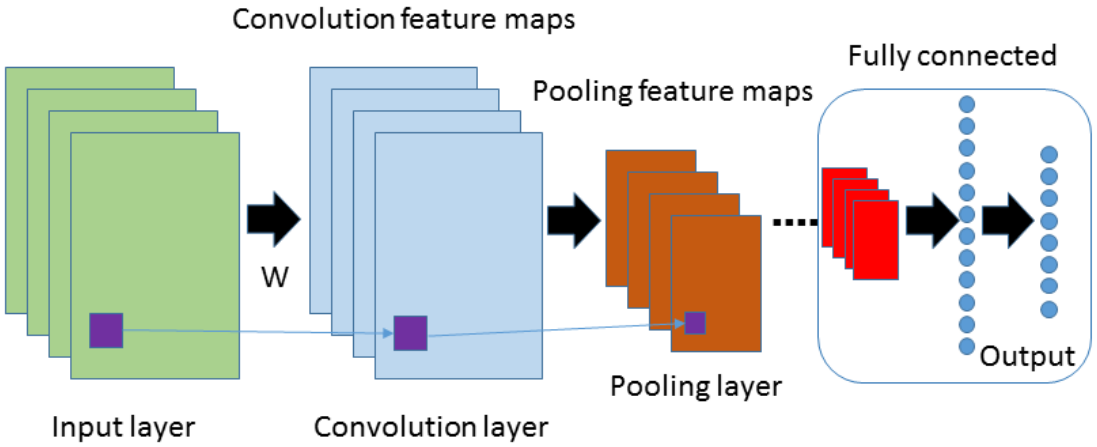


Fig. 1.5 The classical schematic representation of CNN. It includes input layer, convolutional layers, max-pooling layers and output layer. Fully connected part is also presented in this figure.

DBNs which is with generative models, samples cannot be drawn to qualitatively check what are learned.

SDAE is currently applied to many areas such as domain adaptation [80], images classification [275] and text analysis [264].

1.4.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [149] obtained much success in many image processing tasks in past years. A BP-trained CNN [148] set a new MNIST record of 0.39% [198] with no unsupervised pre-training. In 2012, GPU-implementation CNN achieved the best results on ImageNet classification benchmark [136]. CNNs consist of one image processing layer, one or more convolutional layers and max-pooling layers and one classification layer. A classical schematic representation of CNNs can be found in Fig. 1.5. According to a N-classification problem with F training examples and N classes, the squared-error is defined as:

$$E^F = \frac{1}{2}\sum_{f=1}^{F}\sum_{n}^{N}(t_n^f - y_n^f)^2, \tag{1.1}$$

where t_n^f is the value of the n-th dimension about f-th patten's corresponding label, and y_n^f is the n-th output layer unit related to f-th input patten.

The major advantage of CNNs is the use of shared weights. The same filter is utilized

for pixels in the layer, which leads to the reduction of memory footprint and the improvement of result performance. For image classification applications, CNNs utilize relatively little pre-processing, which means that the networks in CNNs are responsible to learn the filters. Another advantage of CNNs is that CNNs do not depend on human effort and prior knowledge for designing features. Besides, compared to traditional neural networks, CNNs are more robust towards variation of input features. The neurons in the hidden layers are connected to the neurons which are in the same spatial area instead of being connected to the nodes in previous layers. Furthermore, when calculating to higher layers, the resolution of the image data will be reduced. However, besides a complex implementation, CNNs have another significant disadvantage that they require very large training data and consume an often impractical amount of time to learn the parameters of the network, which always take several days or weeks. Though the framework for accelerating training and classification of CNNs on Graphic Processing Units (GPUs) has been implemented and performs nearly hundreds of times faster than on the CPU, it is still not enough for real-world applications.

CNNs are considered as one of the most attractive supervised feature learning methods nowadays. CNNs have achieved superior performance for different tasks such as image recognition [244], video analysis [122], Natural language processing [220] and drug discovery [260]. Especially, CNNs based on GoogLeNet increased the MAP (Mean Average precision) of object detection to 0.439 and reduced the classification error to 0.067 [244]. Both of the performances are the best results up to now.

1.4.4 Long Short-Term Memory Neural Networks

Long short-term memory (LSTM) is the extension of recurrent neural network (RNN) architecture which was first proposed in [104] for addressing the vanishing and exploding gradient problems of conventional RNNs. Different from traditional RNNs, when there exist long time lags of unknown size among important events, an LSTM network can classify, predict and process time series from experience. LSTM provides remedies for the RNN's weakness of exponential error decay through adding constant error carousel (CEC) which allows for constant error signal propagation along with the time. Besides, the access to the CEC can be controlled through taking advantages of multiplicative gates.

An LSTM architecture consists of an input layer, an output layer and a layer of memory block cell assemblies. A classical schematic representation of standard LSTM architecture is shown in Fig. 1.6. Fig. 1.6 shows that the memory block assemblies consist of many separate layers: the input gate layer (ι), the memory cell layer (c), the forget gate layer (ϕ) and the output gate layer (ω). The input layer projects all of the connections to each layer. The memory cell layer projects all the connections to the output layer (θ). A diagram of a

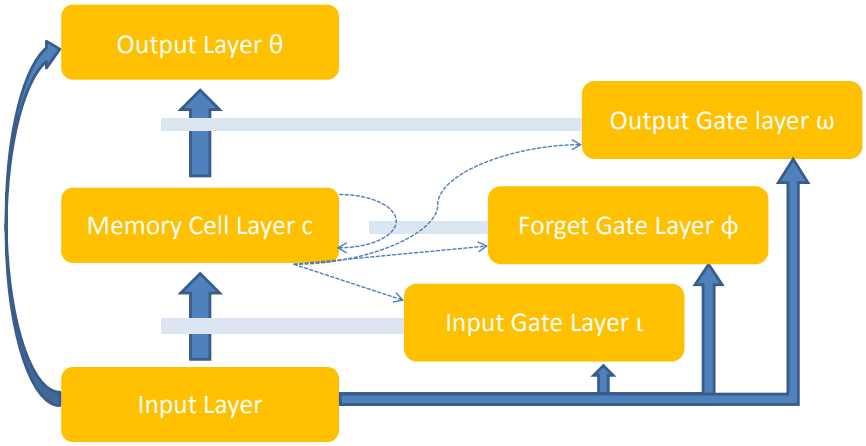


Fig. 1.6 The standard LSTM architecture. The memory block assemblies contain input gates, separate layers of memory cells, forget gates and output gates, in addition to the input and output layers. Blue solid arrows show full all-to-all connectivity between units in one layer. Blue dashed arrows mean connectivity between the units in the two layers which have same index. The light gray bars denote gating relationships.

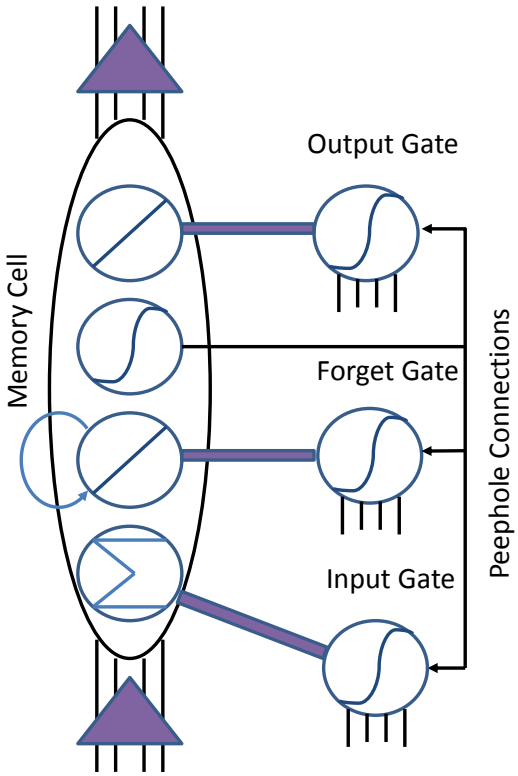


Fig. 1.7 A cross-section of an LSTM network, with a single memory block, and connections from the input layer (bottom) to the output layer (top).

single memory block which consists of four specialized neurons: a memory cell, an input gate, a forget gate and an output gate can be found in Fig. 1.7. The memory cell and the gates receive a connection from every neuron in the input layer. Through gated control, the network can effectively maintain and make use of past observations. The mapping from input sequences $x = (x_1, \cdots, x_T)$ to output sequences $y = (y_1, \cdots, y_T)$ can be computed by the LSTM network through the network unit activations with the equations iteratively from $t = 1$ to T [205]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i), \tag{1.2}$$

$$f_t = \sigma(W_{fx}x_t + W_{mf}m_{t-1} + W_{cf}c_{t-1} + b_f), \tag{1.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c), \tag{1.4}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o), \tag{1.5}$$

$$m_t = o_t \odot h(c_t), \tag{1.6}$$

$$y_t = W_{ym}m_t + b_y, \tag{1.7}$$

where the W terms denote weight matrices, the b terms denote bias vectors, σ is the logistic sigmoid function. i, c, f and o represent the input gate, the cell activation vectors, the forget gate and the output gate respectively. All of them have the same size with the cell output activation vector m. h and g are the cell output and the cell input activation functions respectively. $\odot$ is the element-wise product for vectors.

LSTM can solve the vanishing gradient point problem in RNN. Meanwhile, LSTM has the capability of bridging long time lags between inputs, which can remember inputs up to 1000 time steps in the past. This advantage makes LSTM learn long sequences with long time lags. Besides, it appears that there is no need for parameter fine tuning in LSTM [104]. LSTM can work well over a broad range of parameters such as learning rate, input gate bias and output gate bias. However, in LSTM, the explicit memory adds more weights to each node, and all of these weighs have to be trained. This increases the dimensionality of the task and potentially makes it harder to find an optimal solution.

Applications of LSTM include speech recognition [87], handwriting recognition [88] and human action recognition [8]. Besides, LSTM is also applicable to robot localization [70], online driver distraction detection [270] and many other tasks. Specially, LSTM RNN/HMM hybrids won the best known performance on medium-vocabulary [84] and large-vocabulary speech recognition. Moreover, LSTM-based methods set benchmark records in audio onset detection [173], prosody contour prediction [68] and text-to-speech synthesis [63]. Note that different from DBNs, SDAE and CNNs, LSTM is a sequence learning method which is hardly applied to image classification and object detection. Therefore, in

our comparison experiments in Chapter 2, we only show the performance about LSTM on a gesture recognition dataset (SKIG dataset) and an action recognition dataset (MSRDaily-Activity3D dataset).

1.5 Datasets in Our Research

1.5.1 RGB-D Object Dataset

University of Washington and Intel Labs Settle released this large-scale RGB-D object dataset on June 20, 2011 [144]. It contains 300 common household objects (*i.e.*, apple, banana, keyboard, potato, mushroom, bowl, coffee mug) which are classified into 51 categories. Each object in this dataset was recorded from multiple view angles with resolution of 640×480 at 30 Hz, thus resulting in 153 video sequences (3 video sequences for each object) and nearly 250,000 RGB-D images. Fig. 1.8 illustrates some selected objects from this dataset as well as the examples of RGB-D images. Through WordNet hyponym/hypernym relations, the objects are arranged in a hierarchical structure, which helps many possible algorithms. Ground truth **pose** information and per-frame bounding boxes about all these 300 objects are offered in the dataset. On April 5, 2014, the RGB-D scenes dataset was upgraded to **v**.2, adding 14 new scenes with the tabletop and furniture objects. This new dataset further boosts the research on applications such as category recognition, instance recognition, 3D scene labeling and object pose estimation [143] [20] [19].

To help researchers use this dataset, RGB-D object dataset provides code snippets and software for RGB-D kernel descriptors, reading point clouds (MATLAB) and spinning images (MATLAB) on their website. The performance comparison of different methods tested on this dataset is also reported on the web. RGB-D object dataset is the first large-scale hierarchical multi-view RGB-D dataset, which immediately becomes popular for the evaluation of RGB-D data based methods including CNN-RNN [229], R^2ICA [121], HMP-S [21] and so on. Compared to other object based RGB-D datasets, RGB-D object dataset has more categories, which is used in Chapter 2, Chapter 3 and Chapter 6 for object classification, hyper-parameter optimization and domain adaptation. RGB-D object dataset can be downloaded from their website[9].

[9]http://rgbd-dataset.cs.washington.edu/.

Fig. 1.8 Sample objects from the RGB-D object dataset (left), examples of RGB image and depth image of an object (right top) and RGB-D scene images (right bot).

1.5.2 NYU Depth V1 and V2

Vision Learning Graphics (VLG) lab in New York University created the NYU Depth V1 for indoor-scene object segmentation in 2011. Compared to most works in which the scenes are in a very limited domain [154], this dataset is collected from much wider domains (the background is changing from one to another), facilitating multiple applications. It records video sequences of a great diversity of indoor scenes [222], including a subset of densely labeled video data, raw RGB images, depth images and accelerometer information. On the website, users can find a toolbox for processing data, and suggested training/test splits. Examples of RGB images, raw depth images and labeled images in the dataset are illustrated in Fig. 1.9. Besides the raw depth images, this dataset also provides some pre-processed images on which the black areas with missed depth values have been filled (see Fig. 1.10 for an example). The sampling rate of the Kinect camera is varying from 20 frames per second to 30 frames per second. As a result, there are 108,617 RGB-D images captured from 64 different indoor scenes, such as bedroom, bathroom and kitchen. Every 2 to 3 seconds, frames extracted from the obtained video are processed with dense multi-class labeling. This special subset contains 2347 unique labeled frames.

NYU Dataset V2 [185] is an extension of NYU Dataset V1 and was founded in 2012. This new dataset includes approximately 408,000 RGB images and 1449 aligned RGB-D images with detailed annotations from 464 indoor scenes across 26 scene classes. Obviously, the scale of this dataset is even larger and it is more diversified than NYU dataset V1. The RGB-D images are collected from numerous buildings in three US cities. Meanwhile, this dataset includes 894 different classes about 35,064 objects. Particularly, to identify multiple instances of an object class in one scene, each instance in this scene is given a unique label. The representative research work using these two datasets as the benchmark for indoor

segmentation and classification can be found in [215] [248] [247]. NYU Depth V1 and V2 are the first large-scale indoor-scene based RGB-D datasets, which are immediately considered as the benchmark for the evaluation of RGB-D scene data based methods [262] [124]. Compared to the RGB-D scene datasets which are created before low-cost RGB-D cameras (*e.g.* Kinect) are invented, NYU Depth V1 and V2 have more categories and data, which can help researchers explore more sophisticated algorithms. NYU Depth V1 and V2 are used in Chapter 2, Chapter 3, Chapter 4, Chapter 5 and Chapter 6, which can be downloaded from their websites[10].

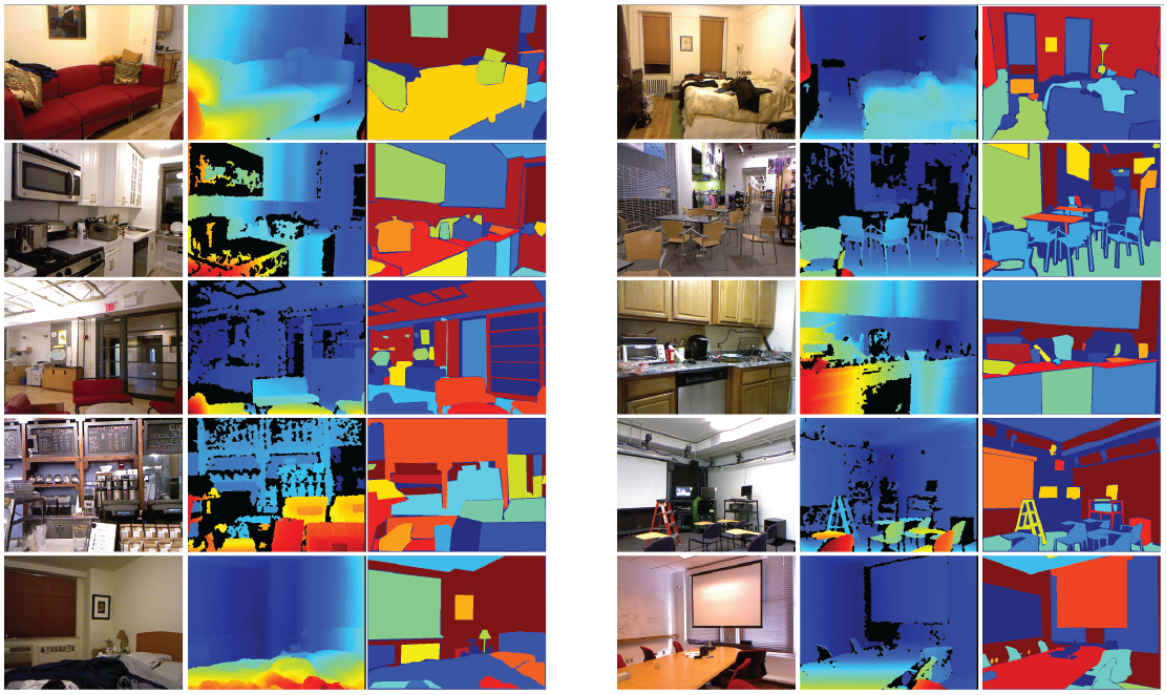


Fig. 1.9 Selected examples of RGB images, raw depth images and class labeled images in NYU dataset.

1.5.3 2D&3D object dataset

2D&3D object dataset includes 18 different categories (*i.e.*, bottles, books, binders, coffee pots, cans, dishes, cups, dish liquids, mice, monitors, pens, perforator, phone, knives, folks, scissors, spoons and drink cartons) with each of them containing 3 to 14 objects resulting in 162 objects. The views of object are recorded every 10 degrees along the vertical axis. Therefore, there are totally $162 \times 36 = 5832$ RGB images and $162 \times 36 = 5832$ depth

[10]http://cs.nyu.edu/~silberman/datasets/nyu_depth_v1.html, http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

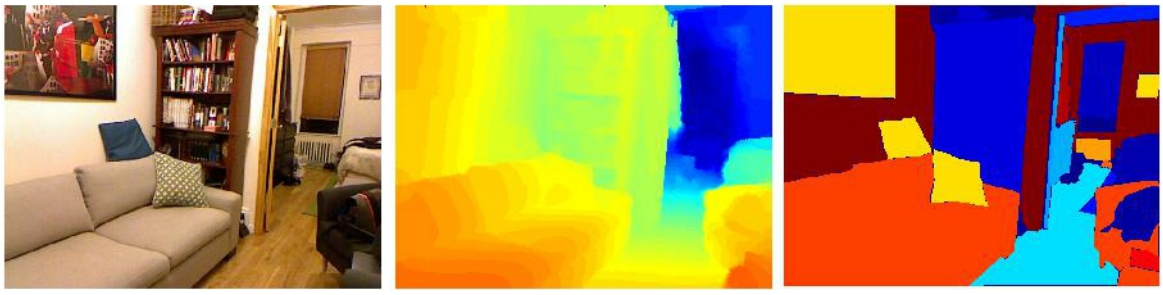


Fig. 1.10 Output of the RGB camera (left), pre-processed depth image (mid) and class labeled image (right) from NYU Depth V1 and V2 dataset.

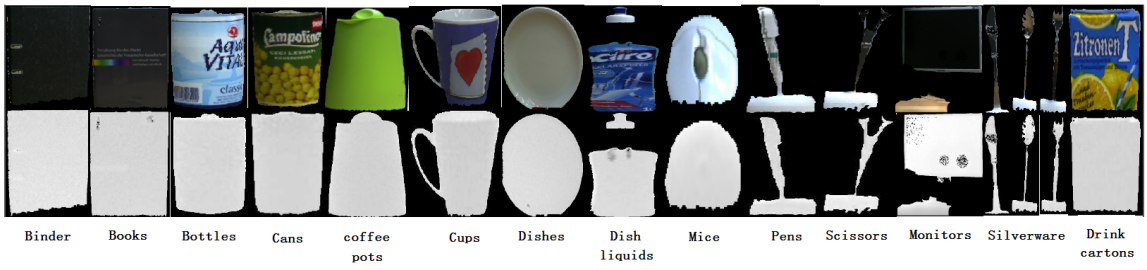


Fig. 1.11 Example images in the 2D&3D Object dataset, which contains 14 object classes (binder, bottles, books, cans, coffee pots, dishes, cups, dish liquids, mice, scissors, pens, monitors, drink cartons and silverware). There are totally 14 paired samples shown in this figure. The Cropped RGB image is shown on the top and the corresponding depth image is on the bottom.

images respectively. Example images from this dataset are given in Fig. 1.11. 2D&3D object dataset is created in 2011 [26] and then widely used in many applications ranging from robotics to computer vision [9] [21]. Compared to the RGB-D object datasets [179] which are created by the past range sensors (*e.g.* Minolta Vivid 910), 2D&3D object dataset has more categories and samples, which can help researchers better explore RGB-D data. 2D&3D object dataset is used in Chapter 2 and Chapter 3, which can be downloaded from their website[11].

1.5.4 Sheffield Kinect Gesture Dataset (SKIG)

SKIG is a hand gesture dataset which was supported by the University of Sheffield since 2013. It is first introduced in [163] and applied to learn discriminative representations. This dataset includes totally 2160 hand-gesture video sequences from six people, 1080 RGB sequences and 1080 depth sequences, respectively. In this dataset, there are 10 categories of

[11] http://www.kyb.tuebingen.mpg.de/research/dep.html.

gestures: triangle (anti-clockwise), circle (clockwise), right and left, up and down, wave, hand signal "Z", comehere, cross, pat and turn around. All these sequences are extracted through a Kinect sensor and the other two synchronized cameras. In order to increase the variety of recorded sequences, subjects are asked to perform three kinds of hand postures: fist, flat and index. Furthermore, three different backgrounds (*i.e.*, wooden board, paper with text and white plain paper) and two illumination conditions (light and dark) are used in SKIG. Therefore, there are 360 different gesture sequences accompanied by hand movement annotation for each subject. Fig. 1.12 shows some frames in this dataset. SKIG is created after the creation of the RGB-D gesture dataset MSRGesture3D [140]. Many researchers choose these two datasets as the benchmarks for the evaluation of RGB-D data based methods [285] [292]. SKIG dataset is used in Chapter 2, which can be downloaded from their website[12].

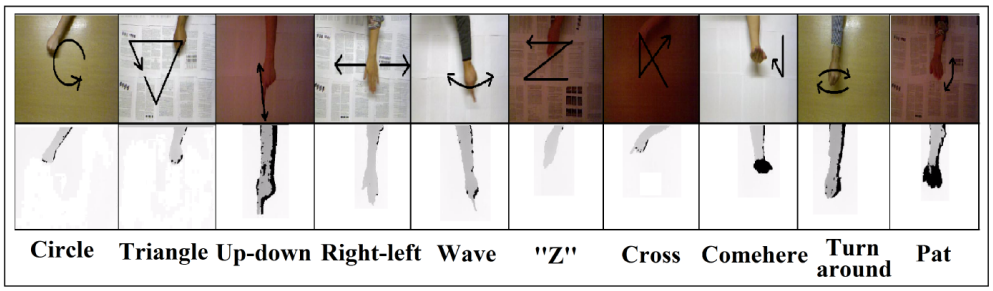


Fig. 1.12 Sample frames from Sheffield Kinect gesture dataset and the descriptions of 10 different categories.

1.5.5 MSRDailyActivity3D Dataset

MSRDailyActivity3D dataset [265] is a daily activity dataset which contains 16 activity types (*e.g.*, drink, eat, play game). There are 10 subjects with each of them performs every activity twice. one is in sitting position and the other is in standing position. Examples of RGB images, raw depth images in this dataset are illustrated in Fig. 1.13. MSRDailyActivity3D is a popular dataset, which has over 700 citations till Aug 2017. Many researchers choose MSRDailyActivity3D dataset as the benchmark for the evaluation of RGB-D data based human activity analysis [192] [168]. MSRDailyActivity3D dataset is used in Chapter 2, which can be downloaded from their website[13].

[12] http://lshao.staff.shef.ac.uk/data/SheffieldKinectGesture.htm.
[13] http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/.

Fig. 1.13 Selected examples of RGB images and raw depth images in MSRDailyActivity3D dataset.

1.5.6 SUN RGB-D dataset

SUN RGB-D dataset [231] is captured by four different sensors (Asus Xtion, Intel RealSense, Kinect v1 and Kinect v2) and contains 10,335 RGB-D images. These images are organized into 19 scene categories such as bathroom, computer room and lecture theatre with more than 80 images in each category. Fig. 1.14 shows some example images from this dataset. Every shown scene is from one of the 19 scene categories. To the best of our knowledge, compared to other RGB-D scene datasets, SUN RGB-D dataset has more categories and samples, which makes it as the largest and most challenging RGB-D scene dataset currently. SUN RGB-D dataset is used in Chapter 4 and Chapter 5, which can be downloaded from their website[14].

1.5.7 Caltech-256 dataset

Caltech-256 dataset [90] contains 256 object categories, for example, “calculator”, “ball”, “coffee mug”, “cereal box”, “Flashlight”, “keyboard”, “mushroom”, “soda can”, “light bulb” and “tomato”. Each category has more than 80 images. It only contains color images. Fig. 1.15 shows some example images from this dataset. Caltech-256 dataset is used in Chapter 6 for the application of recognizing RGB information from RGB-D data, which can share ten common categories with the RGB-D object dataset. Caltech-256 dataset can

[14]http://rgbd.cs.princeton.edu/.

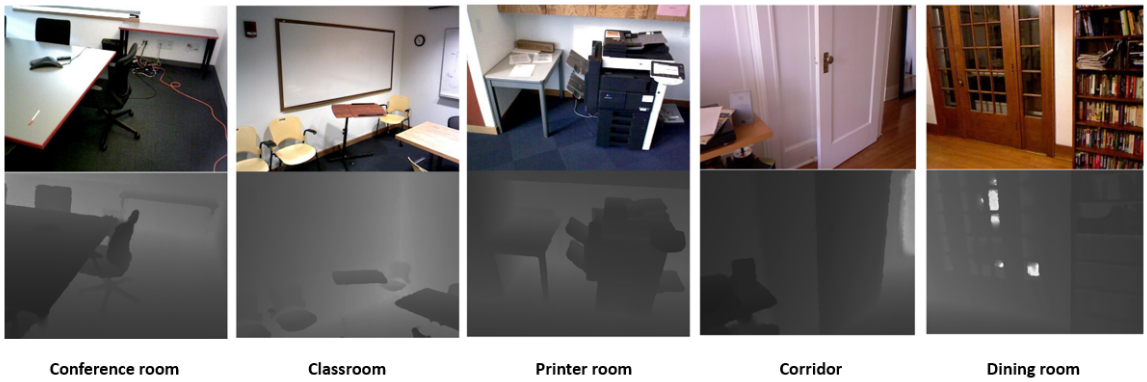


Fig. 1.14 Some example images from the SUN RGB-D dataset. It includes 19 scene categories (conference room, classroom, bookstore, printer room, corridor, dining room and so on). Due to space limitation, we only show 5 paired scene samples in this figure. In each pair, the RGB image is shown on the top and the corresponding depth image is on the bottom.

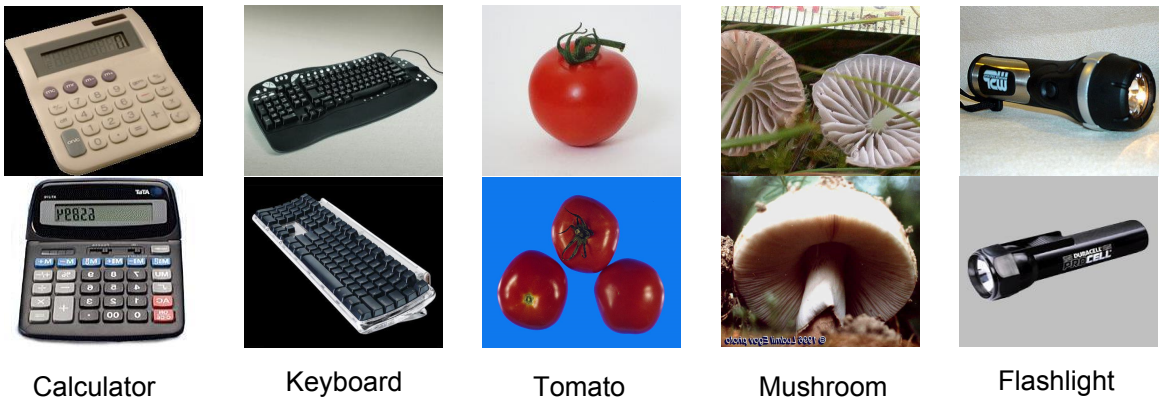


Fig. 1.15 Selected examples of RGB images in Caltech-256 dataset.

be downloaded from their website[15].

1.5.8 Scene-15 dataset

Scene-15 dataset [66] contains 15 scene categories, for example, “bedroom”, “kitchen”, “living room”, “MIT street”, “office” and “MIT forest”. Each category has more than 150 images. It only contains color images. Fig. 1.16 shows some example images from this dataset. Scene-15 dataset is used in Chapter 6 for the application of recognizing RGB information from RGB-D data, which can share four common categories with the NYU

[15]http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

Fig. 1.16 Selected examples of RGB images in Scene-15 dataset.

Depth v1 dataset. Scene-15 dataset can be downloaded from the website[16].

[16]http://www-cvr.ai.uiuc.edu/ponce_grp/data/.

Chapter 2

Classification Performance of Deep Learning Models on RGB-D Image/Video Datasets

2.1 Overview

Learning good feature representations from input data for high-level tasks receives much attention in computer vision, robotics and medical imaging [152, 155, 272, 277]. Image/video classification is a classic and challenging high-level task, which has many practical applications, such as robotic vision [4], image annotation [184, 216] and video surveillance [126, 258]. The objective is to predict the labels of new coming images/videos. Though RGB image/video classification has been studied for many years, it still faces a lot of challenges, such as complicated background, illuminance change and occlusion. With the invention of the **low-cost** Microsoft Kinect sensor, it opens a new dimension (*i.e.*, depth data) to overcome the above challenges. Compared to RGB images, depth images are robust to the variations in color, illumination, rotation angle and scale [51]. Moreover, Deep Neural Networks for high-level tasks obtain great success in recent years. Different from hand-crafted feature representations such as SIFT [167], HOG [53] and STLPC [214], deep learned features are automatically learned from the images or videos. These neural network models improve the state-of-the-art performance on many important datasets (*e.g.*, the ImageNet dataset), and some of them even overcome human performance [261]. Combining the advantages of RGB-D images and Deep Neural Networks, many researchers are making great efforts to design more sophisticated algorithms. However, no single existing approach can successfully handle all scenarios. Therefore, it is important to comprehensively evaluate the

deep feature learning algorithms for image/video classification on popular RGB-D datasets. We believe that this chapter will provide insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion.

The rest of this chapter is organized as follows. In Section 2.2, we give the literature review of this chapter. In Section 2.3, we present the data pre-processing techniques on deep learned features. Section 2.4 describes experimental analysis, results and some tricks on our selected RGB-D datasets. Finally, we draw the summary in Section 2.5.

2.2 Literature Review

2.2.1 Related work to RGB-D information

In the past decades, since RGB images usually provide the limited appearance information of the objects in different scenes, it is extremely difficult to solve certain challenges such as the partition of the foreground and background which have the similar colors and textures. Besides that, the object appearance described by RGB images is sensitive to common variations, such as illuminance change, which significantly impedes the usage of RGB based vision algorithms in realistic situations. Complementary to the RGB images, depth information for each pixel can help to better perceive the scene. RGB-D images/videos provide richer information, leading to more accurate and robust performance on vision applications.

The depth images/videos are generated by a depth sensor. The research of computer vision algorithms based on RGB-D data has attracted a lot of attention in the last few years. Bo et al. [21] presented a hierarchical matching pursuit (HMP) based on sparse coding to learn new feature representations from RGD-D images in an unsupervised way. Tang et al. [246] designed a new feature called histogram of oriented normal vectors (HONV) to capture local 3-D geometric characteristics for object recognition on depth images. In [18], Blum et al. presented an algorithm that can automatically learn feature responses from the image, and the new feature descriptor encodes all available color and depth data into a concise representation. Spinello et al. introduced an RGB-D based people detection approach which combines a local depth-change detector employing HOD and RGB data HOG to detect the people from the RGB-D data in [235] and [236]. In [61], Endres et al. introduced an approach which describes a volumetric voxel representation [273] through optimizing the 3D pose graph using the g^2o [139] framework which can directly be used for robot localization, path planning and navigation [106]. More papers on combining color and depth channels from multiple scenes using RGB-D perception can be found in [250], [219], [158].

2.2.2 Related work to deep learning methods

According to our evaluation, we select four representative deep learning methods including DBNs, SDAE, CNNs and LSTM for our experiments which have been introduced in detail in Chapter 1. These methods have been widely applied in numerous contests in pattern recognition and machine learning. DBN is fine-tuned by backpropagation (BP) without any training pattern deformations which receives much success with 1.2% error rate on the MNIST handwritten digits [101]. Meanwhile, it achieved a good result with an error rate of 26.7% on phoneme recognition on the TIMIT core test set [182]. SDAE was first introduced in [257] as an extension of Stacked auto-encoder (SAE) [145]. BP-trained CNNs [148] achieved a new MNIST record of 0.39% [198]. In 2012, GPU-implemented CNNs achieved the best results on the ImageNet classification benchmark [136]. LSTM won the ICDAR handwriting competition in 2009 and achieved the record of 17.7% phoneme error rate on TIMIT natural speech dataset in 2013. More relevant work and history on these four deep learning methods can be found in [209].

Currently, aiming to obtain more robust features from RGB and depth images/videos, various algorithms based on Deep Neural Networks have been proposed. R. Socher *et al.* presented convolutional and recursive neural networks (CNN-RNN) [229] to obtain higher order features. In CNN-RNN, CNN layers firstly learn low-level translationally invariant features, and then these features are given as inputs into multiple, fixed-tree RNNs. Bai *et al.* proposed subset based sparse auto-encoder and recursive neural networks (Sub-SAE-RNNs) [9] which first train the RGB-Subset-Sparse auto-encoder and the Depth-Subset-Sparse auto-encoder to extract features from RGB images and depth images separately for each subset. These learned features are then transmitted to RNNs to reduce the dimensionality and learn robust hierarchical feature representations. In order to combine hand-crafted features and machine learned features, Jin *et al.* used Locality-constrained Linear Coding (LLC) based spatial pyramid matching to extract hand-crafted features and the CNNs for the machine learned representation [124]. This new feature representation method can obtain the advantages of deep learned features and hand-crafted features. From these above successful methods, we can observe that they are all the extensions of our selected methods (CNNs, DBNs, SDAE or LSTM). Therefore, it is important to explore the performance of our selected methods on different kinds of RGB-D datasets.

Aiming to make a comprehensive performance evaluation, we collect five representative datasets including two RGB-D object datasets [26, 144], an RGB-D scene dataset [223], an RGB-D gesture dataset [163] and an RGB-D activity dataset [265] which can be divided into four categories: object classification, scene classification, gesture classification and action classification. This is the first work to comprehensively focus on the performance of

deep learning methods on popular RGB-D datasets. In our experiments, in order to make the comparison of CNNs, DBNs, SDAE and LSTM under a fair environment, we consider our experiment setting from three aspects: 1) No other pre-training data are included. For example, the pre-trained CNNs model through abundant RGB data can help performance improvement of the RGB-D datasets after the RGB data and HHA-coding depth data fine-tuning. However, not all of our selected four deep learning methods can use other RGB data for pre-training. Therefore, we do not use extra data for pre-training. 2) No other RGB-D coding methods are included. Some particular RGB-D coding methods may not be suitable for other three kinds of deep learned features. Therefore, the design of our experiments is in a traditional way for providing insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion. 3) The fixed time for hyper-parameter adjustment. Different hyper-parameters result in much different performances. For a fair comparison, we take almost the same time to adjust hyper-parameters. The design of our experiments is introduced in detail in Section 2.4. In addition, besides results of the classification accuracies, our evaluation also provides a detailed analysis including confusion matrices and error analysis. Some tricks about adjusting hyper-parameters that we observed during our experiments are also given in this chapter.

2.3 Data Preprocessing on Deep Learned Features

Data preprocessing is an important part of the procedure of learning deep features. In practice, through a reasonable choice of preprocessing steps, it will result in a better performance according to the related task. Common preprocessing methods include normalization and PCA/ZCA whitening. Generally, one without much working experience about the deep learning algorithms will find it hard to adjust the parameters for raw data. When the data is processed in a small regular range, tuning parameters will become easier [48]. However, in the whole process of our experiments, we find that not every dataset is suitable to be either normalized or whitened. Therefore, we will have a test on the dataset and then choose the preprocessing steps according to the situations. Additionally, before we test the algorithms on the datasets, we will first observe properties of the data itself to gain more information which will help us to save more time.

2.3.1 Normalization

General normalization approaches include simple rescaling, per-example mean subtraction and feature standardization. The choice of these methods mainly depends on the data. In our

experiments, since feature standardization is able to set every dimension of raw data to have unit-variance and zero-mean, at the same time, deep features will work with the linear SVM classifier, we choose feature standardization to normalize our data. Therefore, our data is normalized through first subtracting the mean of each dimension from each dimension and then dividing it by its standard deviation.

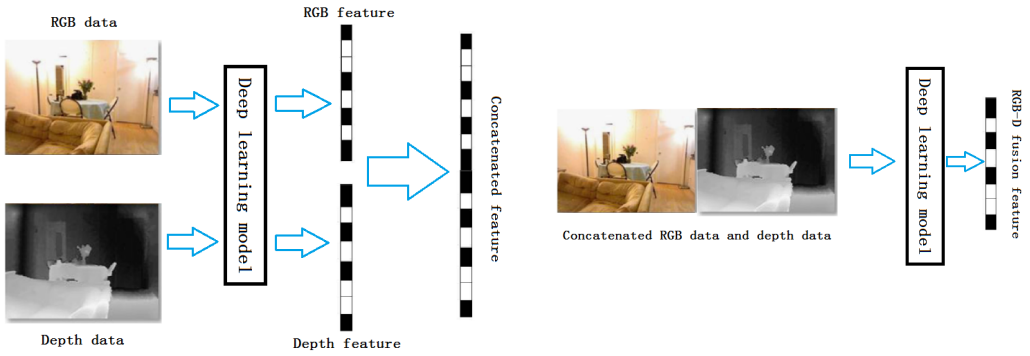
2.3.2 PCA/ZCA Whitening

Following the step of feature standardization, we apply PCA/ZCA whitening to the entire dataset [117]. This is commonly used in deep learning tasks (e.g., [135]). Whitening cannot only make the deep learning algorithm work better but also speed up the convergence of the algorithm. However, in our experiments, for SDAE and DBNs, the results after whitening did not show an obvious improvement. To make the experiments under a fair environment, as long as whitening does not lead to a worse result, we choose to do ZCA whitening to the normalized data. Since we transfer RGB images to grey-scale images to make the data have the stationary property in our experiments and the data has been scaled into a reasonable range, the value of epsilon in ZCA whitening is set large (0.1) for low-pass filtering. More details about PCA/ZCA whitening can be found in [117].

2.4 Experiments on Deep Learning Models

In this section, we evaluate four deep feature learning algorithms (DBNs, CNNs, SDAE and LSTM) on three popular image recognition datasets and two video recognition datasets including 2D&3D object dataset [26], RGB-D object dataset [144], NYU Depth v1 indoor scene segmentation dataset [223], Sheffield Kinect Gesture dataset (SKIG) [163] and MSR-DailyActivity3D dataset [265]. Note that in our experiments, we only show the performance about LSTM on SKIG dataset and MSRDailyActivity3D dataset. In all of these five datasets, we follow the standard setting procedures according to the authors of their respective datasets. Over all of the datasets, we process raw RGB images into grey-scale images and choose the first channel of the depth images as training and test data. According to DBNs, CNNs, SDAE and LSTM, after weights are learned in the deep neural networks, we are able to extract the image or video features from the preprocessed images/videos. Then a linear SVM classifier is trained and tested on the related test sets. To make the results comprehensive, we compare the final results computed on deep features from RGB data only, deep features from depth data only, RGB-D features concatenation and deep features from RGB-D fusion. In RGB-D features concatenation experiments, we concatenate the feature vectors

which are extracted from RGB data and depth data respectively into new vectors. Different from concatenation experiments, according to RGB-D fusion experiments, we firstly concatenate RGB images/frames and relative depth images/frames together, and then extract features from deep learning models. Illustration about these two experimental procedures is shown in Fig. 2.1. Detailed experimental settings, some important parameters, tricks and experiences about adjusting hyper-parameters are shown in the following subsections. All experiments are performed using Matlab 2013b and C++ on a server configured with a 16-core processor and 500G of RAM running the Linux OS.



(a) RGB-D features concatenation (b) Deep features from RGB-D fusion

Fig. 2.1 Illustration about two experimental procedures used in our evaluation work.

2.4.1 2D&3D Object Dataset

We evaluate deep feature learning for object category recognition on the 2D&3D object dataset. For the consistency with the setup in [26], since the low number of examples of classes perforator and phone, our experiments do not include them. Meanwhile, knives, forks and spoons are combined into one category 'silverware'. We choose 6 objects per category for training, and the left are used for testing. If the number of objects in a category is less than 6 (e.g., scissors), 2 objects are added into the test. Since images are cropped in different sizes, we resize each image into 56×56 pixels. We give the final comparison results between neural-network classifier and SVM in Table 2.1.

The hyper-parameters of the DBNs, SDAE and CNNs models are described in Table. 2.2, Table. 2.3 and Table. 2.4. Fig. 2.2 shows confusion matrixes about our three deep learning models across 14 classes on the 2D&3D dataset.

From the comparison results of our experiments about three selected deep learning models on 2D&3D dataset in Table. 2.1, it can be seen that the accuracy of RGB, depth and

Table 2.1 The final comparison results between neural-network classifier and SVM on the 2D&3D object dataset. The second, fourth and seventh columns are the results of RGB test images, depth test images and RGB-D fusion test images on the neural-network classifier separately. The third, fifth, sixth and eighth columns are the results of RGB test images, depth test images, concatenated RGB-D image features and RGB-D fusion test images on SVM separately.

Method	RGB	RGB (SVM)	Depth	Depth (SVM)	RGB-D Concatenation (SVM)	RGB-D fusion	RGB-D fusion (SVM)
DBNs	72.1	74.5	75.7	78.6	**82.3**	78.3	79.1
CNNs	77.3	79.1	81.0	83.5	83.6	82.7	**84.6**
SDAE	73.0	74.5	74.2	75.6	**79.3**	77.6	78.4

Table 2.2 Hyper-parameters about DBNs experiments on the 2D&3D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	2
Units for each layer	100/100/100	100/100/100	100/100
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.009	0.009	0.008
Number of unsupervised epochs	13	13	13
Number of supervised epochs	17	30	24

RGB-D fusion results through SVM outperforms that through the neural-network classifier. In each deep learning method, accuracies of RGB-D concatenation through SVM and RGB-D fusion features through SVM are higher than deep features from RGB data only and deep features from depth data only. In these three methods (DBNs, CNNs and SDAE), CNNs obtain the highest performance (84.6). From the comparison of three confusion matrixes in Fig. 2.2, we can see that our three deep learning models all have the lowest error rates in bottles, cans, coffee pots and cups. Binders, books, pens and scissors have higher error rates. The main reason is that binders and books are similar in shape and color. Pens,

Table 2.3 Hyper-parameters about SDAE experiments on the 2D&3D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	2	2	2
Units for each layer	100/100	100/100	100/200
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.1	0.1	0.1
Number of unsupervised epochs	10	10	15
Number of supervised epochs	10	10	30

Table 2.4 Hyper-parameters about CNNs experiments on the 2D&3D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of convolution layers	2	2	2
Number of sub-sampling layers	2	2	2
Kernel size	5	5	5
Learning rate	0.1	0.06	0.1
Number of epochs	30	60	30

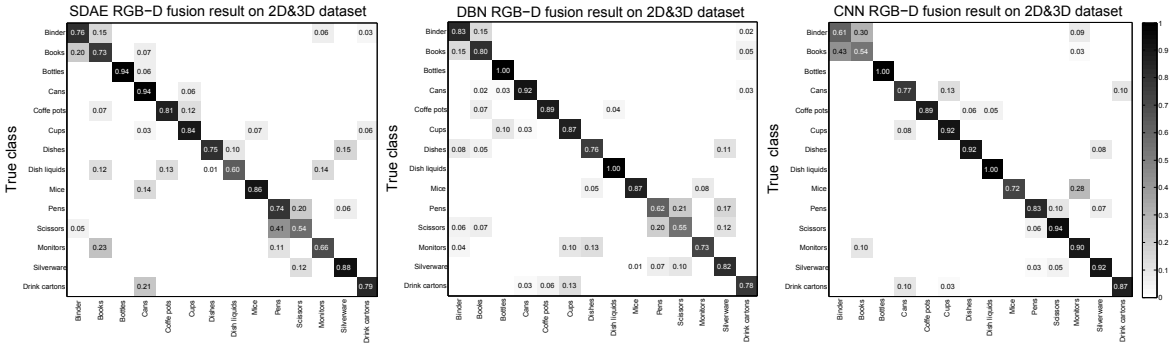


Fig. 2.2 Confusion matrixes about three deep learning models on the 2D&3D dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

scissors and silverware are similar in shape. It is worth to note that the error rates of binders and books in SDAE and DBNs are much lower than that of binders and books in CNNs, and the error rates of pens and scissors in SDAE and DBNs are much higher than that of pens and scissors in CNNs. The error rates of other categories are approximately similar. This interesting phenomenon may be due to the principle of the three different deep learning methods. In addition, it proves that in general SDAE and DBNs are more in common than CNNs.

2.4.2 Object RGB-D Dataset

We test these deep learning algorithms on the second dataset called RGB-D object dataset. Following the setup in [144], we choose to run category recognition experiments by randomly leaving one object out from each category for testing and train the classifiers on the remaining objects. Each image in object RGB-D dataset is resized into 56×56 pixels for consistency with the 2D&3D dataset. Table 2.5 summarizes the comparison between neural-network classifier and SVM.

The hyper-parameters of three deep learning models DBNs, SDAE and CNNs are shown in Table 2.6, Table 2.7 and Table 2.8.

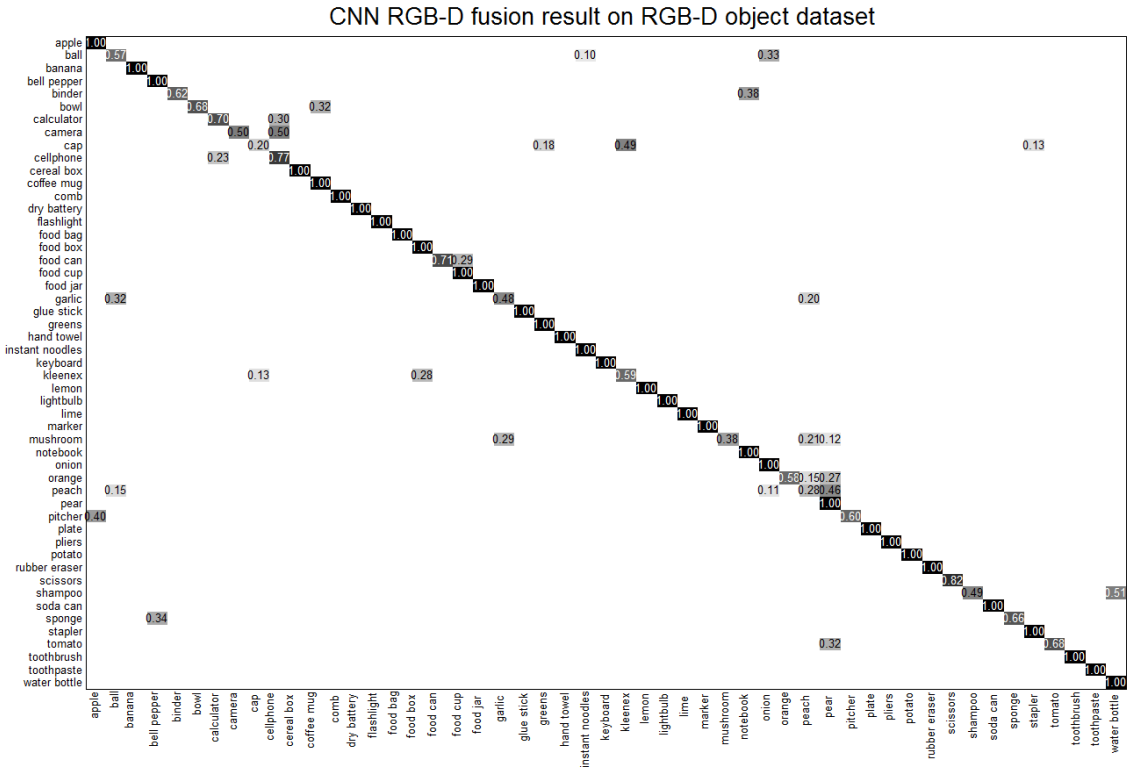


Fig. 2.3 Confusion matrix about CNNs on Object RGB-D Dataset. The labels on the vertical axis express the true classes and the labels on the horizontal axis denote the predicted classes.

Table 2.5 The final comparison results between neural-network classifier and SVM on Object RGB-D dataset. The second, fourth and seventh columns are the results of RGB test images, depth test images and RGB-D fusion test images on the neural-network classifier separately. The third, fifth, sixth and eighth columns are the results of RGB test images, depth test images, concatenated RGB-D image features and RGB-D fusion test images on SVM separately.

Method	RGB	RGB (SVM)	Depth	Depth (SVM)	RGB-D Concatenation (SVM)	RGB-D fusion	RGB-D fusion (SVM)
DBNs	80.9	81.6	75.1	78.6	**84.3**	82.4	83.7
CNNs	82.4	82.5	75.5	78.9	83.4	83.2	**84.8**
SDAE	81.4	82.0	71.9	73.7	82.3	82.6	**84.2**

Table 2.6 Hyper-parameters about DBNs experiments on Object RGB-D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	3
Units for each layer	110/100/20	110/100/20	110/100/20
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.009	0.009	0.009
Number of unsupervised epochs	13	13	13
Number of supervised epochs	8	10	22

As we can see from Table 2.5, CNNs outperform DBNs and SDAE by 0.5% and 0.3%. Due to the limitation of space, we only give the confusion matrix of the best performance (CNNs RGB-D fusion) in our experiments. Fig. 2.3 shows the confusion matrix about CNNs across 51 classes over object RGB-D dataset.

Table 2.7 Hyper-parameters about SDAE experiments on Object RGB-D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	2	2	2
Units for each layer	100/100	130/100	110/200
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.1	0.08	0.05
Number of unsupervised epochs	10	15	15
Number of supervised epochs	15	30	30

Table 2.8 Hyper-parameters about CNNs experiments on Object RGB-D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of convolution layers	2	2	2
Number of sub-sampling layers	2	2	2
Kernel size	5	5	5
Learning rate	0.1	0.06	0.03
Number of epochs	30	60	80

2.4.3 NYU Depth v1

Besides image object classification, we also evaluate these three deep feature learning models on indoor scene classification. NYU Depth v1 dataset consists of 7 different kinds of scene classes totally containing 2347 labeled frames. Since the standard classification protocol removes scene 'cafe' from the dataset, we use the remaining 6 different scenes. It is worth noting that since there are so many objects in one scene and the correlation between images in one scene is low, it makes NYU Depth v1 a very challenging dataset. The baseline when only using RGB images is 55% [223]. Table 2.9 shows the performance comparison between neural-network classifier and SVM on this dataset.

Table 2.9 The performance comparison results between neural-network classifier and SVM on NYU Depth v1 dataset. The second, fourth and seventh columns are the results of RGB test images, depth test images and RGB-D fusion test images on the neural-network classifier separately. The third, fifth, sixth and eighth columns are the results of RGB test images, depth test images, concatenated RGB-D image features and RGB-D fusion test images on SVM separately.

Method	RGB	RGB (SVM)	Depth	Depth (SVM)	RGB-D Concatenation (SVM)	RGB-D fusion	RGB-D fusion (SVM)
DBNs	62.4	66.7	57.3	60.8	68.3	65.5	**70.5**
CNNs	68.4	69.5	56.5	56.9	70.4	70.1	**71.8**
SDAE	65.2	68.4	51.5	55.0	70.3	69.6	**71.1**

The hyper-parameters of DBNs, SDAE and CNNs can be found in Table 2.10, Table 2.11 and Table 2.12. Fig. 2.4 shows confusion matrixes about our three deep learning models across 6 classes over NYU Depth v1 dataset.

As we have mentioned above, NYU depth v1 dataset is very challenging. Therefore, in our three deep learning methods, CNNs achieve the best performance which is only 71.8%. Different from 2D&3D object dataset and object RGB-D dataset, RGB-D fusion through SVM always obtains the higher recognition accuracy (70.5% DBNs, 71.8% CNNs

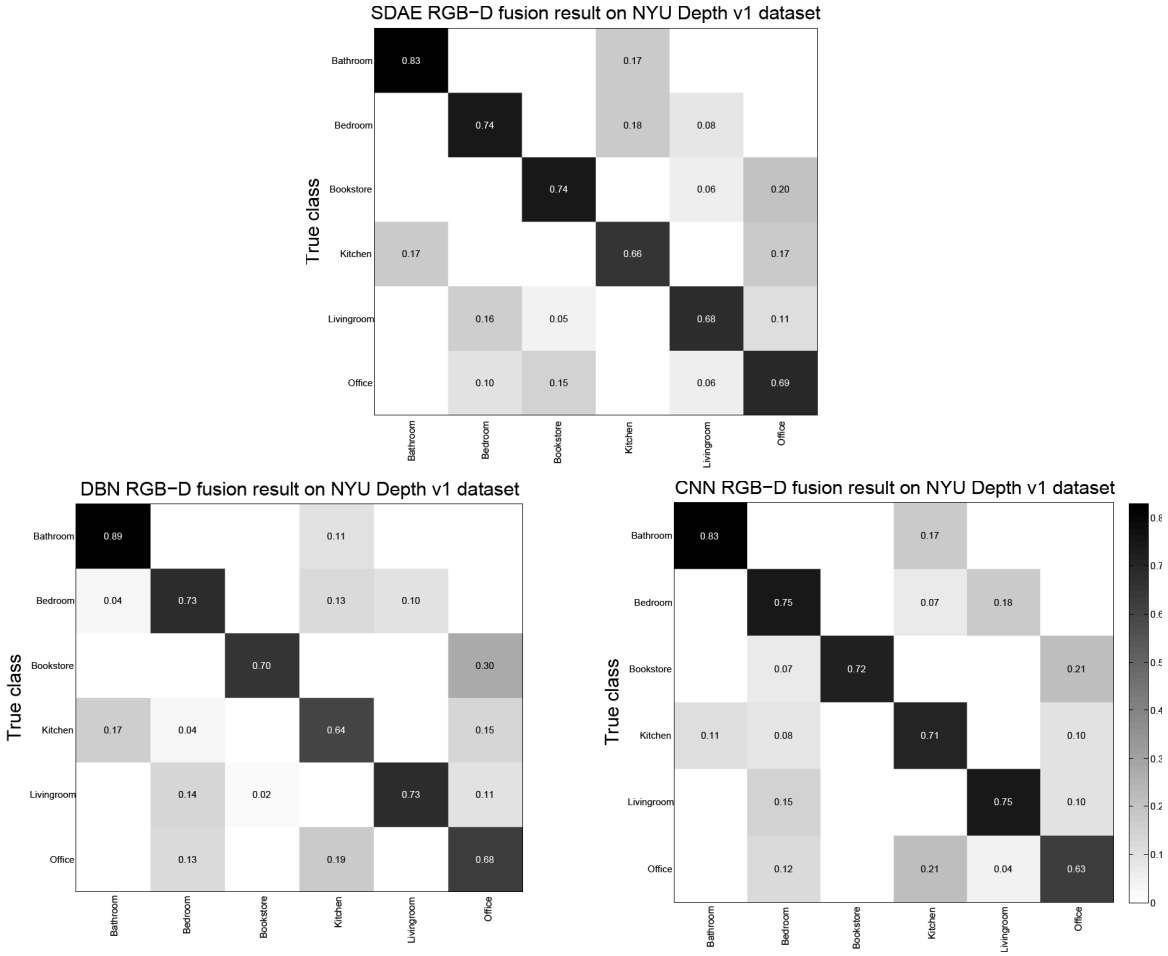


Fig. 2.4 Confusion matrixes about three deep learning models on NYU Depth v1 dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

Table 2.10 Hyper-parameters about DBNs experiments on NYU Depth v1 dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	3
Units for each layer	120/100/80	120/100/80	110/100/100
Unsupervised learning rate	0.06	0.04	0.1
Supervised learning rate	0.006	0.008	0.008
Number of unsupervised epochs	3	3	3
Number of supervised epochs	35	45	22

Table 2.11 Hyper-parameters about SDAE experiments on NYU Depth v1 dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	3
Units for each layer	120/100/80	120/100/60	130/200/120
Unsupervised learning rate	0.01	0.01	0.01
Supervised learning rate	0.1	0.1	0.1
Number of unsupervised epochs	15	15	15
Number of supervised epochs	30	35	50

and 71.1% SDAE) compared to RGB-D concatenation (SVM) and RGB-D fusion. It may be because the scene images from NYU depth v1 dataset contain many irregular objects which seem much more complicated than the object images from the previous two datasets. From the confusion matrixes about these three deep learning methods, to a great extent, it can be seen that the distribution of error rates is similar.

2.4.4 Sheffield Kinect Gesture (SKIG) Dataset

We also evaluate these four deep feature learning algorithms on video classification datasets. SKIG is a hand gesture dataset which contains 10 categories of hand gestures with 2160 hand gesture video sequences from six people, including 1080 RGB sequences and 1080 depth sequences respectively. In our experiments, since it has been proved that 5$\sim$7 frames

Table 2.12 Hyper-parameters about CNNs experiments on NYU Depth v1 dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of convolution layers	2	2	2
Number of sub-sampling layers	2	2	2
Kernel size	8	8	8
Learning rate	0.008	0.008	0.004
Number of epochs	50	45	80

(0.3∼0.5 seconds of video) are enough to have the similar performance with the one obtainable with the entire video sequence [208]. Therefore, each video sequence is resized into $64 \times 48 \times 13$. Following the experimental setting in [163], we choose four objects as the training set and test on the remaining data. Table 2.13 shows the performance comparison between neural-network classifier and SVM on SKIG dataset. Additionally, since 3D-CNNs gain much success in video data classification, we use 3D-CNNs instead of 2D-CNNs in our experiments.

Table 2.13 The performance comparison results between neural-network classifier and SVM on SKIG dataset. The second, fourth and seventh columns are the results of RGB test videos, depth test videos and RGB-D fusion test videos on the neural-network classifier separately. The third, fifth, sixth and eighth columns are the results of RGB test videos, depth test videos, concatenated RGB-D vedio features and RGB-D fusion test videos on SVM separately.

Method	RGB	RGB (SVM)	Depth	Depth (SVM)	RGB-D Concatenation (SVM)	RGB-D fusion	RGB-D fusion (SVM)
DBNs	78.3	83.1	68.9	73.8	84.7	81.5	**85.9**
3D-CNNs	87.2	91.3	77.5	82.2	92.6	88.1	**93.3**
SDAE	78.9	79.1	74.4	78.9	81.1	78.3	**83.3**
LSTM	82.6	83.1	75.7	77.5	87.2	86.7	**91.3**

The hyper-parameters of DBNs, SDAE, 3D-CNNs and LSTM can be found in Table 2.14, Table 2.15, Table 2.16 and Table 2.17.

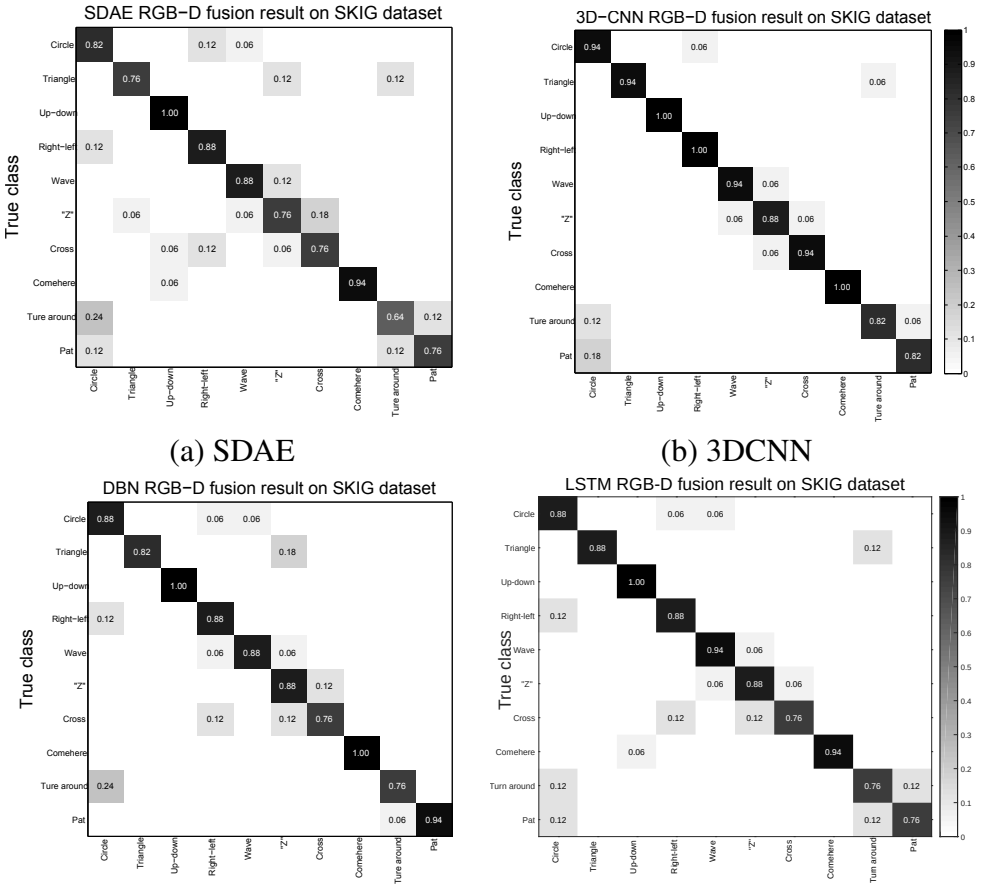
Table 2.14 Hyper-parameters about DBNs experiments on SKIG dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	3
Units for each layer	120/100/100	120/100/100	110/100/100
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.01	0.009	0.006
Number of unsupervised epochs	3	3	3
Number of supervised epochs	30	40	55

To get better results in the 3D-CNNs model, we decay the learning rate a half in each epoch.

Fig. 2.5 shows confusion matrixes about our four deep learning models across 10 classes on the SKIG dataset.

From the comparison of these four deep learning models in Table 2.13, we can see that 3D-CNNs achieve the best performance among four - 93.3%. It may be because that



(a) SDAE (b) 3DCNN

(d) DBN (e) LSTM

Fig. 2.5 Confusion matrixes about four deep learning models on SKIG dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes. From left to right in order, (a) SDAE, (b) 3DCNNs, (c) DBN, (d) LSTM.

Table 2.15 Hyper-parameters about SDAE experiments on SKIG dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	2	2	2
Units for each layer	100/80	100/85	100/100
Unsupervised learning rate	0.01	0.01	0.01
Supervised learning rate	0.01	0.015	0.01
Number of unsupervised epochs	12	15	30
Number of supervised epochs	1200	500	500

Table 2.16 Hyper-parameters about 3D-CNNs experiments on SKIG dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of convolution layers	2	2	2
Number of sub-sampling layers	2	2	2
First Kernel size	$7 \times 7 \times 7$	$7 \times 7 \times 7$	$7 \times 7 \times 7$
Second Kernel size	$7 \times 7 \times 5$	$7 \times 7 \times 5$	$7 \times 7 \times 5$
Initial Learning rate	0.0005	0.0005	0.0004
Number of epochs	40	45	60

3D-CNNs consider the more temporal correlation between video frames [122]. Sequence learning method LSTM with raw pixel features achieves 91.3% on the SKIG dataset, which is better than the performances of DBN and SDAE. It is reasonable because LSTM can learn from experience to classify, process and predict time series. Overall, we obtain high accuracies in this dataset. The main reason is that the ten categories in SKIG dataset can be classified easily. Each category is much different from other categories, and every test video in one category is similar to other test videos in the same category. Therefore, in terms of SKIG dataset, inter-class distance is big and intra-class distance is small. The analysis above suggests that deep learning will produce a good performance with less training samples if the experimental dataset is not challenging.

Table 2.17 Hyper-parameters about LSTM experiments on SKIG dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Memory blocks	50	50	60
Output neurons	10	10	10
Learning rate	0.0001	0.0001	0.0001
Number of epochs	2000	2000	2500

2.4.5 MSRDailyActivity3D Dataset

The last dataset which we test on is MSRDailyActivity3D dataset [265]. We do the same preprocessing procedure like SKIG and resize each sequence to $64 \times 48 \times 13$. Then subject 1 to subject 5 of "sitting on sofa" and subject 1 to subject 5 of "standing" in this dataset are used as training set and the rest are used for evaluation. Table 2.18 shows the accuracies of three deep learning methods.

Table 2.18 The performance comparison results between neural-network classifier and SVM on MSRDailyActivity3D Dataset. The second, fourth and seventh columns are the results of RGB test videos, depth test videos and RGB-D fusion test videos on the neural-network classifier separately. The third, fifth, sixth and eighth columns are the results of RGB test videos, depth test videos, concatenated RGB-D video features and RGB-D fusion test videos on SVM separately.

Method	RGB	RGB (SVM)	Depth	Depth (SVM)	RGB-D Concatenation (SVM)	RGB-D fusion	RGB-D fusion (SVM)
DBNs	51.9	62.5	50.6	53.1	66.3	65.0	**68.1**
3D-CNNs	50.5	65.6	47.3	58.2	61.3	61.3	**68.9**
SDAE	57.5	59.4	46.3	48.1	64.4	62.5	**66.3**
LSTM	49.4	64.4	46.3	57.5	63.1	60.0	**68.1**

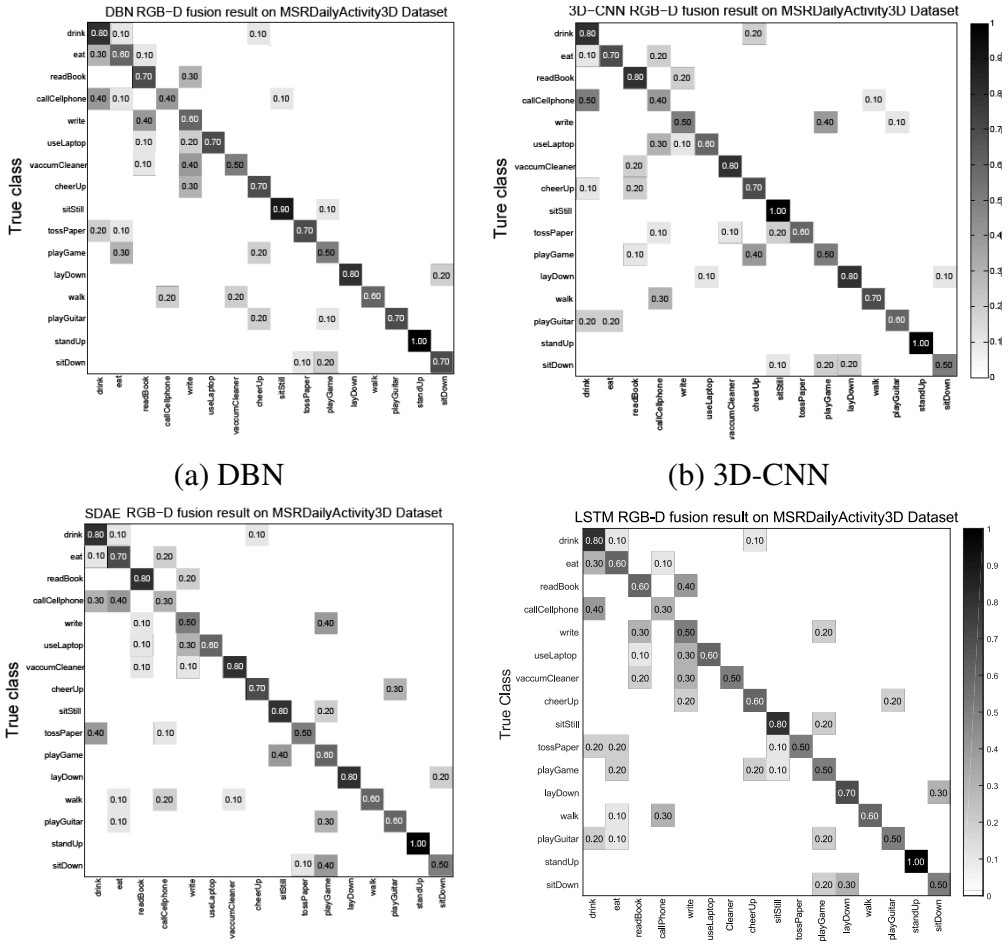
The hyper-parameters of DBNs, SDAE, 3D-CNNs and LSTM are shown in Table 2.19, Table 2.20, Table 2.21 and Table 2.22.

Table 2.19 Hyper-parameters about DBNs experiments on MSRDailyActivity3D Dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	3	3	3
Units for each layer	120/100/100	120/100/100	110/100/100
Unsupervised learning rate	0.1	0.1	0.1
Supervised learning rate	0.004	0.008	0.005
Number of unsupervised epochs	4	4	4
Number of supervised epochs	55	46	60

To get better results in the 3D-CNNs model, we use the same trick as in the experiments of SKIG Dataset by decaying the learning rate a half in every epoch.

In our deep learning experiments on MSRDailyActivity3D dataset, 3D-CNNs achieve a higher accuracy (68.9%) than DBNs (68.1%) and SDAE (66.3%). But compared to the performance of SKIG dataset, we only obtain low accuracies. There are two main reasons. First, it is a very challenging video dataset. According to this dataset, inter-class distance is



(a) DBN (b) 3D-CNN

(d) SDAE (e) LSTM

Fig. 2.6 Confusion matrixes about four deep learning models on MSRDailyActivity3D dataset. The labels on the vertical axis express the true classes and the labels on the horizontal axis denote the predicted classes. From left to right in order, (a) DBN, (b) 3D-CNNs, (c) SDAE, (d) LSTM.

Table 2.20 Hyper-parameters about SDAE experiments on MSRDailyActivity3D Dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of hidden layers	2	2	2
Units for each layer	110/80	110/85	100/100
Unsupervised learning rate	0.01	0.01	0.01
Supervised learning rate	0.01	0.015	0.01
Number of unsupervised epochs	15	20	33
Number of supervised epochs	1000	800	800

Table 2.21 Hyper-parameters about 3D-CNNs experiments on MSRDailyActivity3D Dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Number of convolution layers	2	2	2
Number of sub-sampling layers	2	2	2
First Kernel size	$7 \times 7 \times 7$	$7 \times 7 \times 7$	$7 \times 7 \times 7$
Second Kernel size	$7 \times 7 \times 5$	$7 \times 7 \times 5$	$7 \times 7 \times 5$
Initial Learning rate	0.0003	0.0005	0.0004
Number of epochs	50	45	60

small and intra-class distance is big. Second, there are no enough training samples for deep learning models. Therefore, it can be seen that it will show a bad performance with less training samples if the experimental dataset is very challenging. Fig. 2.6 shows confusion matrixes about our four deep learning models across 16 classes over MSRDailyActivity3D dataset.

2.4.6 Tricks For Adjusting Hyper-parameters

Deep neural network learning involves many hyper-parameters to be tuned such as the learning rate, the momentum, the kernel size, the number of layers and the number of epochs. In the process of adjusting hyper-parameters, inappropriate parameters may result in overfitting or convergence to a locally optimal solution, so it requires a strong practical experience.

Table 2.22 Hyper-parameters about LSTM experiments on MSRDailyActivity3D dataset.

Selected hyper-parameters	RGB	Depth	RGB-D fusion
Memory blocks	60	60	70
Output neurons	16	16	16
Learning rate	0.0001	0.0001	0.0001
Number of epochs	2000	2000	2500

Therefore, many researchers who did not utilize neural networks in the past have the impression of this tuning as a "black art". It is true that experiences can help a lot, but the research on hyper-parameter optimization moves towards a more fully automated fashion. The widely used strategies on hyper-parameter optimization are grid search and manual search. Bergstra and Bengio [15] first proposed the very simple alternative called "random sampling" to standard methods which works very well. Meanwhile, it is easy to implement. Bergstra et al. then presented automatic sequential optimization which outperforms both manual and random search in [16]. This work is successfully extended in [228] which considers the hyper-parameters optimization problem through the framework of Bayesian optimization. In this chapter, we give some tricks about how to choose hyper-parameters in our experiments. It can help other researchers use deep neural networks.

During our experiments, we find that DBNs are more difficult than CNNs and SDAE in hyper-parameter optimization. With inappropriate parameters, DBNs easily converge to locally optimal solutions. According to DBNs, CNNs, SDAE and LSTM, the reconstruction error always increases remarkably if the learning rate is too large. Therefore, we follow the simplest solution and try several small log-spaced values $(10^{-1}, 10^{-2}, \ldots)$ [99]. Then we narrow the region and choose the value where we obtain the lowest error. During the training, the learning rate is reduced half in each epoch prior to termination. The choice of the number of hidden layers and units for each layer is very much dataset-dependent. From most tasks that we worked on, it can be found that when the image size is small and training samples are not a lot, it does not need a large number of hidden units and very deep hidden layers in DBNs and SDAE. Therefore, we define the initial number of hidden layers as 2 and the initial units for each layer as 100. Then we keep fine-tuning the number of hidden layers and the units manually till finding the ideal results. For CNNs, the kernel size of small image datasets is usually in the 5×5 range. On the other hand, natural image datasets which have more pixels in each dimension are better to use large kernel sizes such as 10×10 or 15×15. In all of our experiments, we set momentum which is used for increasing the speed of learning as 0.9. The number of unsupervised epochs and number of supervised epochs is usually initialized as 10 and increased with the step 5 $(10, 15, 20, \ldots)$.

2.4.7 Overall Performance Analysis

Based on the experimental results reported and analyzed above, we also conduct a detailed analysis of all the benchmarking deep learning models and RGB-D datasets. From the comparison of selected deep learning models (DBNs, SDAE, LSTM and 2D, 3D-CNNs), 2D-CNNs for RGB-D images and 3D-CNNs for RGB-D videos always outperform DBNs, SDAE and LSTM in classification tasks. LSTM shows advantages compared to DBNs and

SDAE in RGB-D video classification tasks. The results of RGB-D concatenation (SVM) and RGB-D fusion (SVM) are better than other methods. For a fair comparison, we take almost the same time to adjust hyper-parameters. From the final performances of Table 2.1, Table 2.5 and Table 2.9, we can find that the more challengeable the dataset is, the lower the accuracy. In our RGB-D video experiments, the results in Table 2.13 reveal that it will also show a great performance without lots of training samples when the experimental datasets are simple. According to the results of our experiments, we can find that sometimes the performance of the depth images is a little better than the performance of the RGB images (Table 2.1). It may be due to the choice of the particular dataset. Compared to other selected RGB-D datasets, 2D&3D object dataset is an object-centric dataset which has less categories and samples. Particularly, since the scene-centric datasets such as NYU Depth v1 have more objects in one image, which makes the RGB images variable, the appearance and texture information from the RGB images play a more important role than the shape information from the depth images. Therefore, in the experiments of NYU Depth v1 dataset, RGB images are more informative than depth images. On the other hand, in the comparison between 2D&3D object dataset and Object RGB-D dataset, though both of these two datasets are object-centric datasets, 2D&3D object dataset has much less categories and samples (18 categories and 5832 RGB-D image pairs) than Object RGB-D dataset (51 categories and 45000 RGB-D image pairs). It makes the shapes of different categories show larger discrimination among inter-class object images. For example, the categories (onion, ball, apple and tomato) in Object RGB-D dataset will result in similar shapes. Less categories which have similar shapes in a classification task will result in a higher performance of object depth images. Therefore, since 2D&3D object dataset has a small amount of images which results in the RGB image pairs are not variable and the depth image pairs have larger discrimination in the shape, it is reasonable that the depth images have a better performance than RGB images in this dataset. In addition, in our experiments, sometimes the combination of RGB data and depth information does not show a significant improvement than the utilization of only one of them. This is what we expect, the conclusion is that combining both RGB and depth information gives higher overall performance regardless of the choice of deep learning methods. The information complement each other. According to some experimental results, using RGB data alone already gives a high accuracy. Therefore, including depth information does not increase performance significantly. The scene image classification task is much more challenging which results in the combination of RGB and depth information improves the accuracy dramatically.

2.5 Summary

In this chapter, we perform large-scale experiments to comprehensively evaluate the performance of deep feature learning models for RGB-D image/video classification. Based on the benchmark experiments, we give the overall performance analysis about our results and introduce some tricks about adjusting hyper-parameters. We note that RGB-D fusion methods using CNNs with numerous training samples always outperform our other selected methods (DBNs, SDAE and LSTM). Since LSTM can learn from experience to classify, process and predict time series, it achieved better performances than DBN and SDAE in video classification tasks. Moreover, this large-scale performance evaluation work facilitates a better understanding of the deep learning models on RGB-D datasets.

Chapter 3

Hyper-parameter Optimization via Classification Complexity Assessment

3.1 Overview

Following the work in Chapter 2, we observe that the performances of many machine learning methods vary significantly with different sets of hyper-parameters especially for complex models, such as DBN, SDAE, CNNs and other deep learning models, which always have tens to hundreds of hyper-parameters. Hyper-parameters such as learning rate, kernel size, number of layers and number of epochs, which are generally set via trials, are different from the parameters which are learned on data in a training procedure. The difference between hyper-parameters and parameters can be found in Fig. 3.1. Adequate control of hyper-parameters can prevent over-fitting which happens when the model is too flexible. Since deep learning models involve many hyper-parameters to be tuned, which often results in a huge difference between unoptimized and state-of-the-art performance, it is difficult to reproduce and extend published results [47, 49, 196]. Therefore, choosing a proper set of hyper-parameters is both crucial and difficult.

The rest of this chapter is organized in the following way. In Sect. 3.2, we show the literature review. The motivation and contributions of this chapter are discussed in Sect. 3.3. In Sect. 3.4, we introduce the classification complexity measures which are used in our methodology. In Sect. 3.5, we give a detailed introduction of our methodology including a flow chart. Experimental setup and results on the verification details of our framework and the relevant experimental result analysis are comprehensively presented in Sect. 3.6. Finally, the scenario and summary of this work are given in Sect. 3.7 and 3.8.

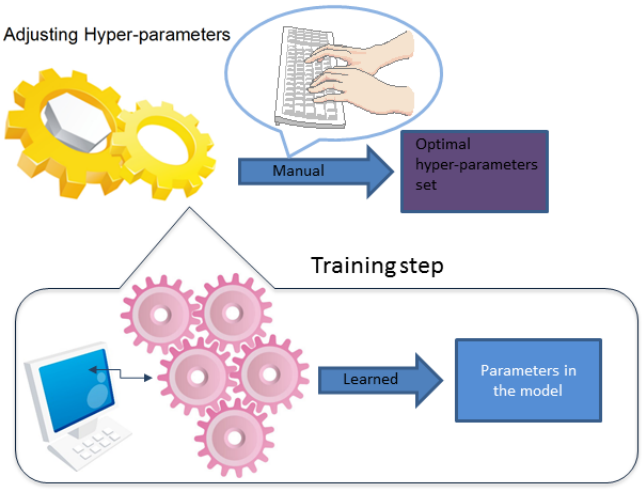


Fig. 3.1 The difference between hyper-parameters and parameters. The set of hyper-parameters is manually adjusted. On the other hand, the parameters in the models are learned in a training procedure.

3.2 Literature Review

Hyper-parameter optimization has not made great progress up till now. Previously, the only available methods were Bayesian optimization [181], grid search [145] and random search [15]. Bayesian optimization based on Gaussian process models is a methodology for the global optimization of the unknown functions, which proves to outperform other global hyper-parameter optimization methods in low-dimensional problems with numerical hyper-parameters on a few challenging optimization benchmarks [25] [125]. Grid search is a commonly used way in practice for hyper-parameter optimization, which is quite simple but very expensive. It applies a grid on all hyper-parameter values and exhaustively searches each block of the grid in a learning algorithm. Therefore, a strategy called manual grid search, which first runs a small grid to observe the optimum point and then expands the grid in that direction, is proposed in [145] [98]. Random search is a very simple alternative of grid search, which is much more efficient than grid search for optimizing the parameters of neural networks [15] and very easy to implement. It takes a fewer random sample points on the grid instead of the points over the entire grid. Meanwhile, random search can be improved into automatic sequential optimization [16]. Bergstra et al. show that random search performs as well as grid search in high-dimensional spaces in [15].

Following the above works, an increasing number of more sophisticated tuning methods have been proposed in recent years. Frank Hutter et al. proposed a sequential model-based optimization method, which approximates the response surface through training a random

forest of regression trees and offers principled approaches to weight the importance of each dimension [113] [115] [237]. This method is proved to outperform Gaussian processes for categorical hyper-parameters. Zheng and Bilenko present Nelder-Mead for improving the efficiency of hyper-parameter search in supervised machine learning by minimizing the number of re-sampled evaluations at each configuration [291]. This algorithm is easy to be implemented and no less efficient than Bayesian optimization. Bardenet et al. proposed a generic method called surrogate-based collaborative hyper-parameter tuning which incorporates the knowledge from past experiments when simultaneously tuning a learning algorithm on new problems [60]. Yang et al. introduced structured search which generalizes the notion of independence among sparseness of the Hessian in nonlinear optimization, random variables in statistics and the generalized distributive law [279]. In addition, there are some released software packages for hyper-parameter optimization which are widely used by many researchers, such as scikit-learn (Grid search and random search) [195], Spearmint (Bayesian optimization using Gaussian processes) [228] and SMAC (Random forest tuning) [114].

3.3 Motivation and Contributions

Motivation. Above mentioned hyper-parameters optimization methods have their limitations. For example, the computational expense of grid search grows dramatically with the number of hyper-parameters in the model, thus in general grid search is limited to few hyper-parameters with no more than 10 which makes it difficult to evaluate some challenging benchmarks such as DBN and CNNs. Even though following method random search which randomly chooses trials has been proved that it is more efficient and practical than grid search, it has to face the cumbersome step: how to choose the initial set of trials from $\{P_1, P_2, \cdots, P_n\}$? Currently, the initial sets of all methods are randomly chosen by researchers. If the initial set of hyper-parameters is far away from the optimal set, the procedure of hyper-parameter optimization will become extremely inefficient. In [15], Yoshua Bengio et al. propose that different datasets, tasks and learning algorithm families result in different sets of hyper-parameters. Motivated by this, we assume that there should be a relationship between the data distribution of datasets and hyper-parameters, then we give a novel framework to assess our assumption. Results from three representative deep learning models on six real-world datasets demonstrate that our assumption provides insight into the relationship between dataset classification tasks and hyper-parameters optimization. Through our framework, we can obtain an initial set of relative hyper-parameters, thus reducing the number of trials in terms of hyper-parameter space $\{P_1, P_2, \cdots, P_n\}$.

Contributions. Firstly, we propose a novel framework to assess the relationship between dataset classification complexity measures and hyper-parameters optimization. Through this framework, it can be available to choose the initial set of hyper-parameters for a random classification task. To our best knowledge, it is the first work to optimize hyper-parameters through the classification complexity concept. Secondly, a new method to search similar classification complexity data distribution is presented in this chapter. This new method not only bridges dataset classification complexity measures and hyper-parameters optimization in deep learning models, but also can be widely used in optimization of general machine learning models, effective prototype selection and classifier selection.

3.4 Classification Complexity Measures

The concept of dataset classification complexity measure is firstly proposed in [103], which is immediately widely used in classifier selection [171], instance selection [156] and prototype selection [183]. In our experiments, to verify the relationship between classification complexity and hyper-parameters, we select 10 n-class complexity measures from [103] and [183]. Some other complexity measures which are only defined for two-class discrimination in [103] are extended to n-class discrimination in [183]. Then we put these measures into a 10-dimensional measurement space and represent each classification complexity by points in this space. These measures are normalized as far as possible in the experiments for comparability. Based on the literature of supervised and unsupervised learning, Ho and Basu group classification complexity measures into three categories: measures of overlap, measures of class separability and measures of geometry, and topology and density. These selected measures are briefly described in the following subsections.

3.4.1 Measures of overlap

Generalized Fisher's Discriminant Ratio (F1). The plain version of Fisher's discriminant ratio which is first proposed in [103] computes the two classes separability according to feature distribution. It is then extended to n-class which also considers the whole space. It measures the inter-class distances over the inner-class distances. It can be denoted as:

$$F1 = \frac{\sum_{i=1}^{C} n_i \cdot \delta(m, m_i)}{\sum_{i=1}^{C} \sum_{j=1}^{n_i} \delta(x_j^i, m_i)}, \tag{3.1}$$

where n_i is the number of samples in class i, δ is the metric, m is the overall mean, x_j^i expresses the sample j of class i and m_i is the mean of class i. The range of this measure is

from 0 to $+\infty$. If $F1$ is small, it means strong overlapping.

Volume of overlap region (F2). This measure is to find the maximum and minimum values of each class for each feature and then compute the length of the overlap region. $F2$ can be denoted as:

$$F2 = \sum_{(c_i,c_j)} \prod_{k} \frac{MINmax_k - MAXmin_k}{MAXmax_k - MINmin_k}, \tag{3.2}$$

where (c_i, c_j) are all pairs of classes and $i = 1, \ldots, d$ for a d-dimensional problem. Then the minimum and maximum values of each feature f_k can be defined in class c_i and class c_j as $min(f_k, c_i)$, $min(f_k, c_j)$, $max(f_k, c_i)$ and $max(f_k, c_j)$ respectively. Therefore, in Eq. (3.2), we can let

$$MINmax_k = min\{max(f_k, c_i), max(f_k, c_j)\},$$

$$MAXmin_k = max\{min(f_k, c_i), max(f_k, c_j)\},$$

$$MAXmax_k = max\{max(f_k, c_i), max(f_k, c_j)\},$$

$$MINmin_k = min\{min(f_k, c_i), min(f_k, c_j)\}.$$

The range of $F2$ is 0 to 1. If values of $F2$ are small, it indicates small overlapping.

Feature efficiency (F3). This measure evaluates how much each feature contributes to the classification of n-class. J. M. Sotoca *et al.* define it as the whole parts of points in the overlap range of any features in all pairs of classes. The points which are in more than one range are only counted one time. The joint contribution features are not included in this measure. This value can be calculated as follows:

$$F3 = \max_{h=1,2,\cdots,d} \frac{|P_h|}{N_e}, \tag{3.3}$$

$$P_h = \{x_i | x_{ih} \leq MAXmin_h \text{ or } x_{ih} \geq MINmax_h\}, \tag{3.4}$$

where N_e is the number of examples in the dataset. x_{ih} is the *h-th* attribute value of the *i-th* pattern x_i. The value of $F3$ is from 0 to 1. Small value of $F3$ expresses a high overlap.

3.4.2 Measures of class separability

Minimized sum of error distance by linear programming (L1). The formulation of this measure is proposed in [227] for solving both separable and nonseparable conditions. It can be obtained through minimizing the sum of distances of the error points to the separating

hyperplane,

$$\begin{aligned} &minmize && \mathbf{a}^t\mathbf{t} \\ &subject\ to && Z^t\mathbf{w}+\mathbf{t}\geq\mathbf{b} \\ & && \mathbf{t}\geq 0, \end{aligned} \tag{3.5}$$

where **a**, **b** are constant vectors which are both chosen as **1**, **t** is the error vector. **w** is the weight vector. Z is the matrix. Each column **z** on Z can be defined on an input vector **x** and its class c (c_1 or c_2):

$$\begin{cases} \mathbf{z}=+\mathbf{x} & \text{if } c=c_1 \\ \mathbf{z}=-\mathbf{x} & \text{if } c=c_2 \end{cases} \tag{3.6}$$

If $L1$ is small, it means dataset is linearly separable. The value of $L1$ is in the range [0,1].

Error rate of linear classifier by linear programming (L2). Considering the definition of $L1$, measure $L2$ is expressed as the error rate of the linear classifier on the original training set. The value of $L2$ is from 0 to 1. Small values in $L2$ mean that the dataset is linearly separable.

Fraction of points on class boundary (N1). Through a class-blind Minimum Spanning Tree (MST) connecting all the points to their nearest neighbors over the entire dataset, the part of the points connected to other classes by an edge over all points in the dataset are calculated as measure $N1$ (see Fig. 3.2). $N1$ is in the domain [0,1]. Large values calculated by this measure indicate that the dataset is difficult for classification.

Ratio of average intra/inter class nearest neighbor distance (N2). After the computation of the Euclidean distance from each point to its nearest neighbor within or outside the class, the measure $N2$ can be calculated as the ration of the average of all the distances to intra-class nearest neighbors and the average of all the distance to inter-class nearest neighbors. Therefore, $N2$ can be expressed as:

$$N2=\frac{\sum_{i=0}^{N} intraDist(ex_i)}{\sum_{i=0}^{N} interDist(ex_i)}, \tag{3.7}$$

where ex_i is each input class, $intraDist(ex_i)$ is the distance of ex_i to its nearest neighbor within the class and $interDist(ex_i)$ is the distance of ex_i to nearest neighbor of any other classes. The domain of $N2$ is from 0 to 1. Larger values mean that the samples in the same class are disperse.

Error rate of 1 nearest neighbor classifier (N3). Measure N3 can be calculated by means of the leave-one-out error of the nearest neighbor classifier. The range of N3 is from

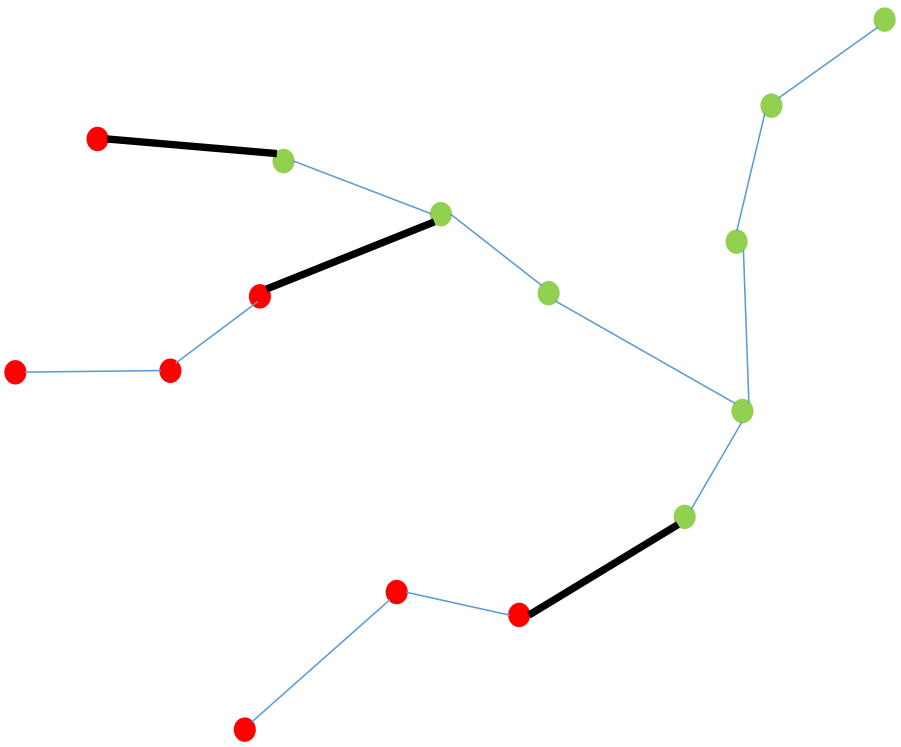


Fig. 3.2 Example of an MST. Bold lines connect instances which belong to different classes. The sum of these connections is divided by the total number of instances and taken as N1.

0 to 1. Low values indicate that the complexity of this classification problem is low.

3.4.3 Measures of geometry, topology and density

Nonlinearity of linear classifier by linear programming (L3). The measure on nonlinearity is first proposed in [105]. According to a training set, $L3$ is defined through the creation of a test set by linear interpolation between random pairs of points belonging to the same class (see Fig. 3.3). The domain of $L3$ is from 0 to 1. If $L3$ is small, it indicates that the dataset is not linearly separable.

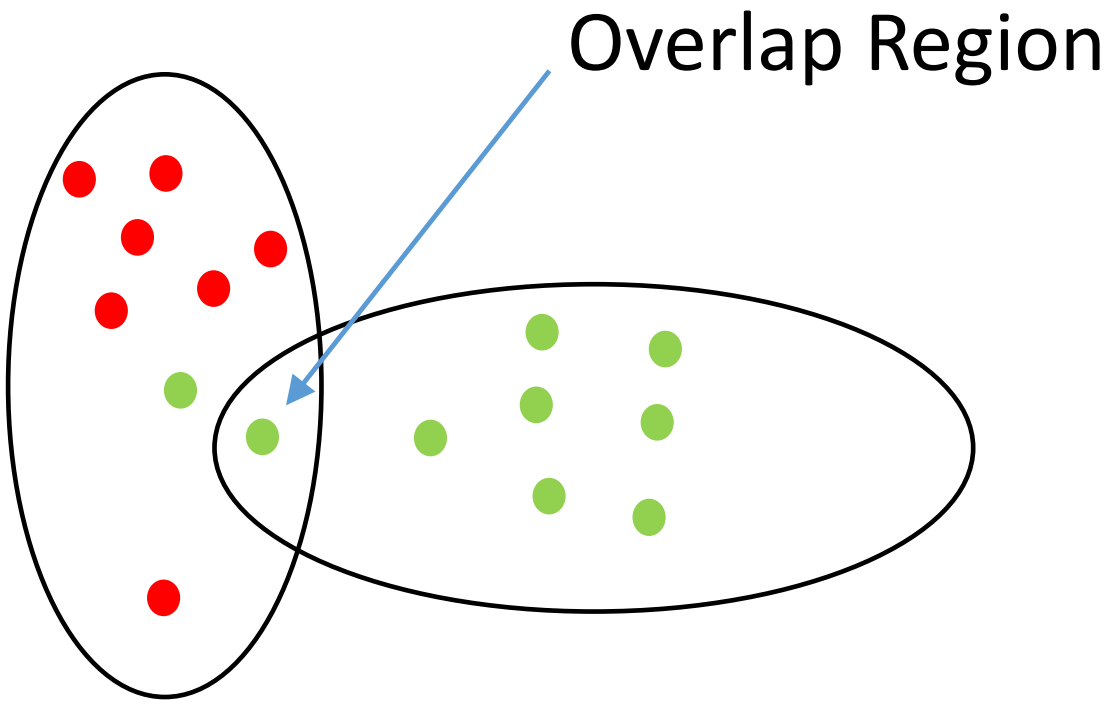


Fig. 3.3 Example of an overlap region obtained by L3.

Nonlinearity of 1 nearest neighbor classifier (N4). $N4$ is defined in the same way with $L3$, but considers classifier as the 1 nearest neighbor. $N4$ is in the range [0,1]. Contrary to $L3$, low values of $N4$ indicate that the dataset is linearly separable.

ε-Neighborhoods (T1). Lebourgeois and Emptoz first propose the measure of $T1$ in [72]. In [103], a reflexive and symmetric binary relation $\mathscr{R}$ of two points x and y in a set F is considered. $\mathscr{R}$ can be defined by $x\mathscr{R}y \leftrightarrow d(x,y) < \varepsilon$, where $d(x,y)$ is a matric and ε is nonzero constant. Define $\Gamma(x) = \{y \in F \mid y\mathscr{R}x\}$, the adherence mapping ad from the power

set $\mathscr{P}(F)$ to $\mathscr{P}(F)$ can be expressed as:

$$\begin{cases} ad(\phi) & = \phi \\ ad(\{x\}) & = \{x\} \cup \Gamma(x) \\ ad(A) & = \cup_{x \in A} ad(\{x\}) \; \forall A \subset F \end{cases} \tag{3.8}$$

Adherence subsets can be grown from a singleton:

$$\begin{aligned} \{x\} : \{x\} &= ad^0(x), \\ ad(\{x\}) &= ad^1(\{x\}), \cdots, \\ ad(ad^n(\{x\})) &= ad^{n+1}(\{x\}), \end{aligned} \tag{3.9}$$

where j can be called as the adherence order in $ad^j(\{x\})$. From a point of each class, one can grow successive adherence subsets to the higher order n such that $ad^n(\{x\})$ includes only points of the same class but $ad^{n+1}(\{x\})$ includes points of the opposite class. $T1$ can be obtained from the number of biggest adherence subsets needed to cover each class. Then $T1$ can be calculated through the normalization of the counted number with the total number of points (see Fig. 3.4). The values of $T1$ are in the range [0,1]. Small values of $T1$ mean that the samples in this dataset are easily separable. More details about $T1$ can be found in [72].

Density measure (T2). Measure $T2$ relates the density of spatial distributions of samples to the space through calculating the average number of instances of the dataset over the feature dimension number, and it is expressed as follows:

$$T2 = \frac{N}{D}, \tag{3.10}$$

where N is the average number of instances in the dataset and D is the number of feature dimensions.

From the above 12 complexity measures for classification problem, since $L3$ cannot be extended to a multi-classification problem and $T2$ is not suitable to our framework, we select 10 ($F1$, $F2$, $F3$, $L1$, $L2$, $N1$, $N2$, $N3$, $N4$, $T1$) of them to define our vector for assessing complexity of a dataset classification problem. In this chapter, we consider each measure as one feature in the complexity feature vector. In addition, the complexity feature vector is always corresponding to one dataset. In total, we extract 10 dimensional features for each classification problem. The depict of our complexity feature vector is shown in Fig. 3.5.

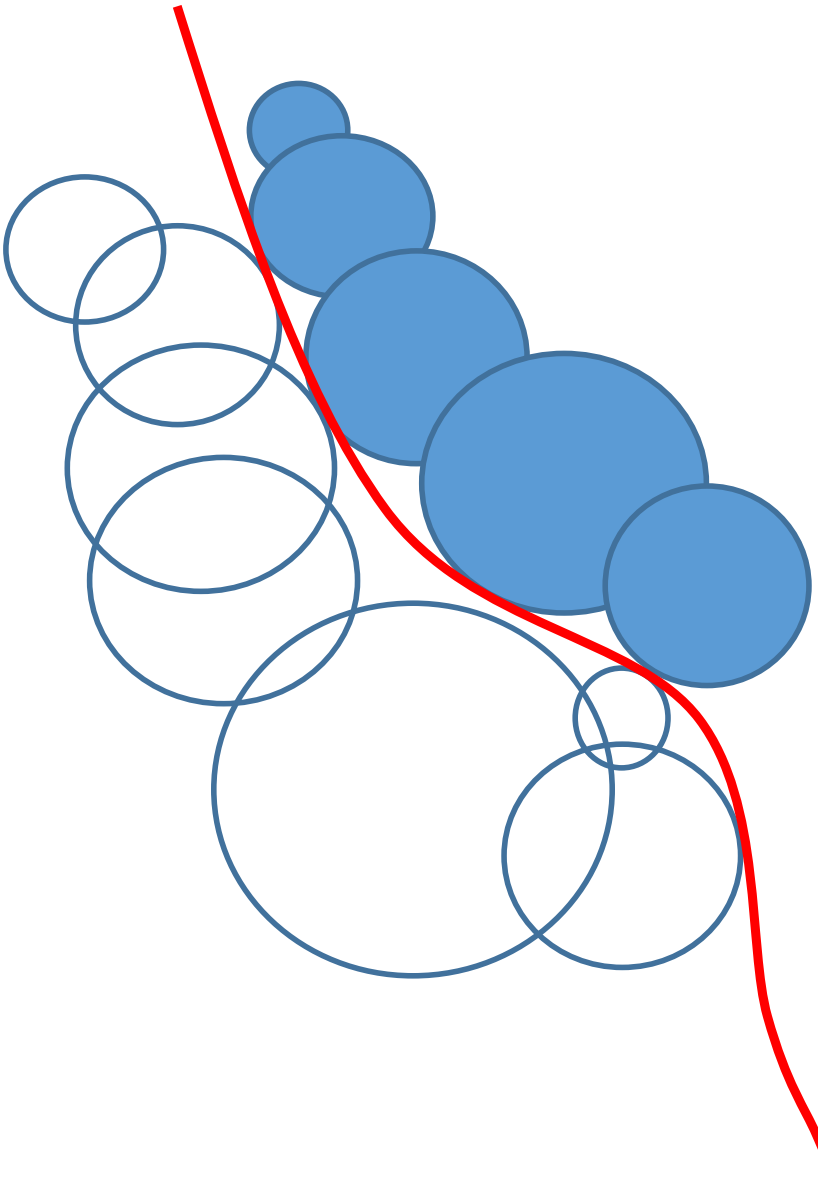


Fig. 3.4 Example of adherence subsets required to describe the class boundary between two classes.

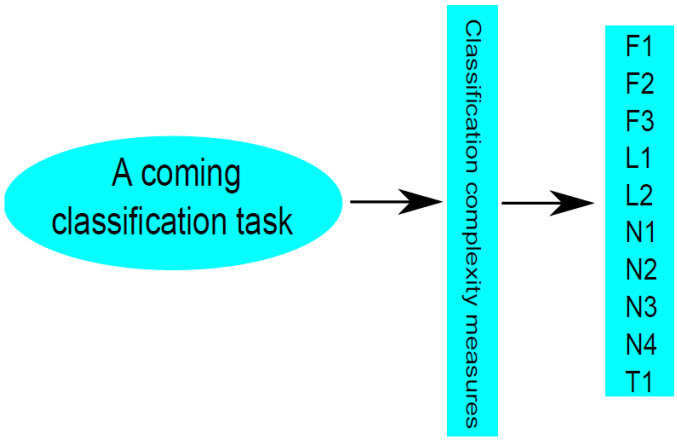


Fig. 3.5 The flow chart of the extraction procedure of our complexity feature vector. We use the defined complexity measures to extract the complexity feature vectors.

3.5 Methodology

In this chapter, we intend to combine classification complexity measures and hyper-parameters optimization to build an effective hyper-parameters selection framework. Our basic assumption is that the selection of hyper-parameters may relate to classification complexity. According to much experimental experience, we find that sometimes when a set of hyper-parameters is manually adjusted and available for one classification task, it is also suitable for another classification task. But sometimes the available set of hyper-parameters is inappropriate to other classification tasks. To verify our assumption and find the relationship between classification tasks and hyper-parameters, we create the illustrated framework in Fig. 3.6. As we can see from Fig. 3.6, the proposed hyper-parameters selection framework mainly consists of two components: classification tasks manual adjustment procedure and new classification task selection procedure. The classification tasks manual adjustment procedure is marked by blue arrows and new classification task selection procedure is marked by red arrows. In the first component, it shows a lot of classification tasks $\{T_1, T_2, \cdots, T_n\}$ including object RGB/Depth image classification task, indoor scene RGB/Depth classification task and so on. Then these tasks are put into complexity measure system and deep learning models respectively. Through the complexity measure system, a complexity feature vector set $\{V_1, V_2, \cdots, V_n\}$ can be obtained. Each vector V_n is corresponding to a classification task T_n. The hyper-parameters of deep learning models are manually adjusted. Each set of hyper-parameters is collected into the hyper-parameter sets $\{P_1, P_2, \cdots, P_n\}$. In the second component, the complexity feature vector V_{new} of new classification task T_{new} is firstly obtained through complexity measure system. Via k-Nearest Neighbors algorithm,

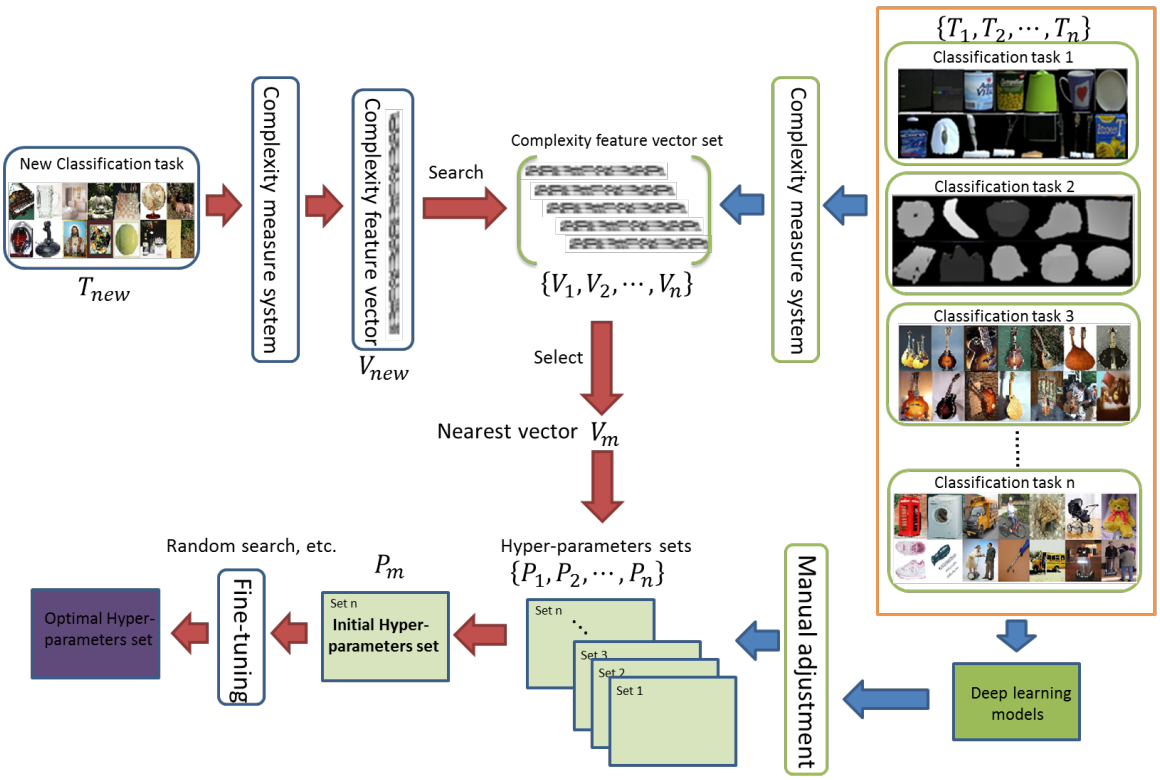


Fig. 3.6 The flow chart of the proposed framework. The framework intends to select the optimal set of hyper-parameters for a new classification task (see text for more details).

the nearest vector V_m of V_{new} can be selected from the complexity feature vector set, where $m \in \{1, \cdots, n\}$. Then the corresponding hyper-parameter set P_m of V_m is chosen as initial hyper-parameter set for T_{new}. After fine-tuning through random search or other methods on P_m, the optimal hyper-parameter set of T_{new} can be acquired.

3.6 Experimental Setup

In this section, we describe the selected datasets, the verification details of our framework and the relevant experimental results analysis.

3.6.1 Datasets

In this chapter, we systematically verify the relationship between classification complexity and hyper-parameters in deep learning models on three popular RGB-D datasets including 2D&3D object dataset, RGB-D object dataset and NYU Depth v1 indoor scene dataset.

Since each RGB-D dataset has one RGB dataset and one depth dataset, we actually carry out our verification on six datasets.

2D&3D object dataset: For the consistency with the setup in [26], since the low number of examples of classes perforator and phone, our experiments do not include them. Meanwhile, knives, folks and spoons are combined into one category 'silverware'. We choose 6 objects per category for training, and the left are used for test. If the number of objects in category is less than 6 (scissors), 2 objects are added into test. Since cropped images in different sizes, we resize each image into 56×56 pixels.

RGB-D object dataset: Following the setup in [144], we choose to have category recognition experiments with randomly sample one object from the categories for testing. Each image in object RGB-D dataset is resized into 56×56 pixels.

NYU Depth v1 indoor scene dataset: Since the standard classification protocol removes scene 'cafe' from dataset1, we use the remaining 6 different scenes.

3.6.2 Experimental environment and data preprocessing

In these six datasets, we follow the standard setting procedure with the corresponding authors on their respective data. Over our experiments, we process raw RGB images into grey-scale images and choose the first channel of the depth images as training and test data. Generally, a disordered data distribution will bring much trouble to the researchers who are without much working experience about the deep learning methods in the adjustment of hyper-parameters. Tuning parameters will become easier if the data values are processed into a small regular range. Therefore, in order to gain a better performance with the related task, we choose reasonable data preprocessing steps before putting data into complexity assessment system and deep learning models. Common preprocessing method includes normalization (simple rescaling, per-example mean subtraction and feature standardization) and PCA/ZCA whitening. The choice of these methods clearly depends on the data. Therefore, we will first have tests on the datasets and then choose the suitable preprocessing steps according to the properties of data. In our experiments, same with the data preprocessing work in Chapter 2, since feature standardization can set every dimension of raw data to have unit-variance and zero-mean, we choose feature standardization as our normalization method. Our data is normalized through first subtracting the mean of each dimension from each dimension and then dividing it by its standard deviation. On the other hand, PCA/ZCA whitening improves the performance of CNNs. But for SDAE and DBN, the results after whitening cannot have an obvious improvement. To make the experiments under a fair environment, as long as whitening method does not lead to a worse result, we choose to do ZCA whitening to the normalized data. Since we transfer RGB images to grey-scale images for

making the data have the stationary property and the data has been scaled into a reasonable range, the value of *epsilon* in ZCA whitening is set large (0.1) for low-pass filtering. Aim to obtain accurate experimental results, after weights are learned in the deep neural networks, we enable to extract the image features from the preprocessed images. The classification measure to each dataset is also calculated from preprocessed data. All our experiments are performed using Matlab 2013b and C++ on a server configured with a 16-core processor and 500G of RAM running the Linux OS.

3.6.3 Classification complexity measures in experiments

Table. 3.1 summarizes the values for the selected complexity measures ($F1$, $F2$, $F3$, $L1$, $L2$, $N1$, $N2$, $N3$, $N4$, $T1$) on six experimental datasets (2D&3D object RGB, 2D&3D object depth, object RGB, object depth, NYU v1 RGB and NYU v1 depth). The complexity feature vectors are defined as 10-dimensional. In our experiments, we consider the complexity feature vectors on 2D&3D object RGB, 2D&3D object depth, object RGB, object depth, NYU v1 RGB and NYU v1 depth as vector 1, vector 2, vector 3, vector 4, vector 5 and vector 6 respectively. Aim to better express the relationship among these classification tasks, we use k-Nearest Neighbors algorithm (k-NN) to calculate the distance in these features. Fig. 3.7 shows the relationship among these vectors. Low values mean that these two vectors are near, and high values mean that these two vectors are far. It can be seen from this figure that the distance among vector 1, vector 2, vector 3 and vector 4 is short and the distance between vector 5 and vector 6 is also short, but vector 1 to vector 4 are far from vector 5 and vector 6. It maybe because of two reasons. One reason is that vector 1 to vector 4 are from object classification tasks. The images from object classification tasks are cropped objects. But vector 5 and vector 6 are from scene classification tasks. Each image from scene classification tasks are indoor scenes. The other reason is that NYU v1 RGB and NYU v1 depth datasets are more challenging than 2D&3D object RGB, 2D&3D object depth, object RGB and object depth.

3.6.4 Hyper-parameters and performance in experiments

In this part, we will provide the values of the hyper-parameters about the performance in our classification task experiments. In total, we can obtain six sets of hyper-parameters on six datasets. Since there are a great many permutations of hyper-parameter values and all of the adjustment work is accomplished manually, our results are probably not the best performance of the selected deep learning models. During our manual adjustment procedure, we follow the simplest solution and try several small log-spaced values $(10^{-1}, 10^{-2}, \ldots)$ [99].

Table 3.1 Complexity measures on experimental datasets. We consider the complexity feature vectors on 2D&3D object RGB, 2D&3D object depth, object RGB, object depth, NYU v1 RGB and NYU v1 depth as vector 1, vector 2, vector 3, vector 4, vector 5 and vector 6 respectively.

Measure	2D&3D RGB (vector 1)	2D&3D depth (vector 2)	Object RGB (vector 3)	Object depth (vector 4)	NYU v1 RGB (vector 5)	NYU v1 depth (vector 6)
F1	0.5739	0.8210	0.7505	0.5940	0.2231	0.2518
F2	0	0	0	0	$3.6293e^{-47}$	$3.8727e^{-314}$
F3	0.6471	0.6500	0.5521	0.5832	0.1200	0.0619
L1	0.0048	0.0048	0	0	0	0.0010
L2	0.0012	0.0012	0	0	0	0.0010
N1	0.1248	0.1252	0.1172	0.2179	0.6561	0.4837
N2	0.3921	0.3823	0.4258	0.4746	0.7387	0.5749
N3	0.0440	0.0507	0.0229	0.0852	0.4724	0.3020
N4	0.0812	0.0160	0.0027	$9.1575e^{-04}$	0.0378	0.0714
T1	0.3293	0.3610	0.3595	0.3559	0.5102	0.6337

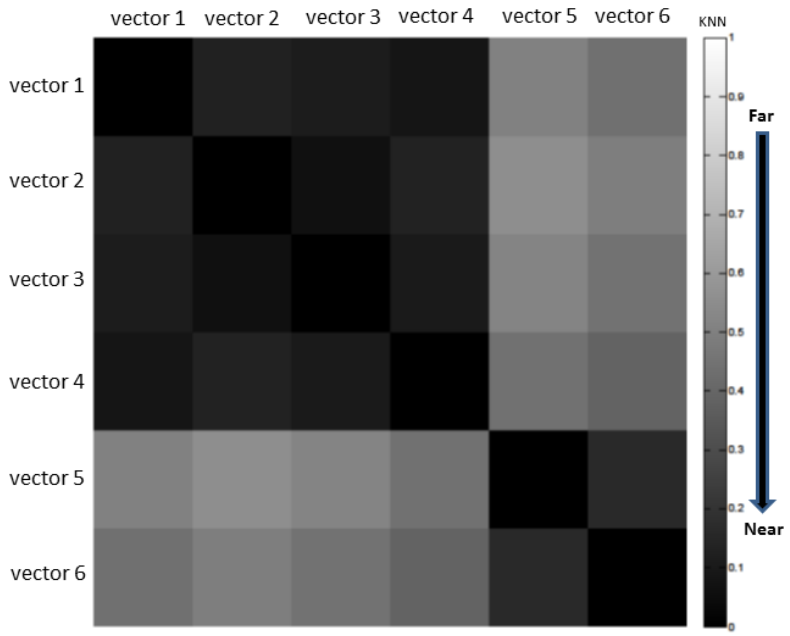


Fig. 3.7 This figure shows the distance among the complexity feature vectors of experimental datasets. Each block is the relationship between two vectors. Low values mean that these two vectors are near, and high values mean that these two vectors are far.

Then we narrow the region and choose the value where we obtain the lowest error. This process takes much time in the experiment. During training, the learning rate is reduced half in each epoch prior to termination. The choice of the number of hidden layers and units for each layer is very much dataset-dependent. In all of our experiments, we set the momentum which is used for increasing the speed of learning as 0.9. The number of unsupervised epochs and number of supervised epochs are usually initialized as 10 and increased with the step 5 $(10, 15, 20, \ldots)$. The chosen hyper-parameters of DBN, SDAE and CNNs on six datasets are shown in Table. 3.2.

After obtaining the hyper-parameters of DBN, SDAE and CNNs on six datasets, to observe the accuracy performance, we test each set of hyper-parameters of one dataset over other five datasets. It leads to 3 (deep models)$\times$6 (hyper-parameter sets)$\times$6 (datasets)= 108 accuracy results in total. These data are collected in Table. 3.3 (DBN), Table. 3.4 (SDAE) and Table. 3.5 (CNNs). In these tables, set 1 to set 6 are the hyper-parameters of DBN, SDAE or CNNs in Table. 3.2. Each set of hyper-parameters corresponds to one cell from the left to the right of one model in Table. 3.2. The values express the classification results of each set of hyper-parameters on different datasets. We highlight the highest accuracies in each dataset.

Performance analysis: In Table. 3.3, Table. 3.4 and Table. 3.5, the highlighted diagonal lines are the highest accuracies which we obtained from our classification performance experiments in Chapter 2 (Table 2.1, Table 2.5 and Table 2.9). Other values are all lower than the values on the diagonal lines. The experiment setting follows three aspects: 1) No other pre-training data are included. 2) No other RGB-D coding methods are included. 3) The fixed time for hyperparameter adjustment. The highest accuracies which we obtained use the same set of hyperparameters with the accuracies in Table. 2.1, Table. 2.5 and Table. 2.9. From these tables (Table. 3.3, Table. 3.4 and Table. 3.5), we can find that when we use hyper-parameter set 1 to set 4 to test 2D&3D object RGB dataset, 2D&3D object depth dataset, object RGB dataset and object depth dataset, the accuracies are always dramatically higher than hyper-parameter set 5 and set 6 on these datasets, and the difference among these accuracies is small. Meanwhile, when we use hyper-parameter set 5 and set 6 to test NYU v1 RGB dataset and NYU v1 depth dataset, the accuracies are dramatically higher than hyper-parameter set 1 to set 4 on these two datasets, and the difference among these accuracies is also small. Therefore, it shows that hyper-parameter set 1 to set 4 are more suitable for 2D&3D object RGB dataset, 2D&3D object depth dataset, object RGB dataset and object depth dataset, and hyper-parameter set 5 and set 6 are more suitable for NYU v1 RGB dataset and NYU v1 depth dataset. The experimental results and the relationship among the complexity feature vectors of experimental datasets (in Fig. 3.7) show how our

Table 3.2 The chosen hyper-parameters of DBN, SDAE and CNNs on six datasets which include 2D&3D object RGB dataset, 2D&3D object depth dataset, object RGB dataset, object depth dataset, NYU Depth v1 RGB dataset and NYU Depth v1 depth dataset. In this table, N_h is the number of hidden layers, Units is the number of units for each layer, Un_l_r is unsupervised learning rate, l_r is Supervised learning rate, un_ep= 13 is the number of unsupervised epochs and ep is the number of supervised epochs in DBN and SDAE. Con_layers, Sub_layers, Ker_size, l_r and ep are number of convolution layers, number of sub-sampling layers, kernel size, learning rate, number of epochs and outsamples respectively in CNNs.

Models	2D&3D RGB	2D&3D depth	Object RGB	Object depth	NYU v1 RGB	NYU v1 depth
DBN	N_h= 3	N_h= 3	N_h= 3	N_h= 3	N_h= 3	N_h= 3
	Units= 100/100/100	Units= 100/100/100	Units= 110/100/20	Units= 110/100/20	Units= 120/100/80	Units= 120/100/80
	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.06	Un_l_r= 0.04
	l_r= 0.009	l_r= 0.009	l_r= 0.009	l_r= 0.009	l_r= 0.006	l_r= 0.008
	un_ep= 13	un_ep= 13	un_ep= 13	un_ep= 13	un_ep= 3	un_ep= 3
	ep= 17	ep= 30	ep= 8	ep= 10	ep= 35	ep= 45
SDAE	N_h= 2	N_h= 2	N_h= 2	N_h= 2	N_h= 3	N_h= 3
	Units= 100/100	Units= 100/100	Units= 100/100	Units= 130/100	Units= 120/100/80	Units= 120/100/60
	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.1	Un_l_r= 0.01	Un_l_r= 0.01
	l_r= 0.1	l_r= 0.1	l_r= 0.1	l_r= 0.08	l_r= 0.1	l_r= 0.1
	un_ep= 10	un_ep= 10	un_ep= 10	un_ep= 15	un_ep= 15	un_ep= 15
	ep= 10	ep= 10	ep= 15	ep= 30	ep= 30	ep= 35
CNNs	Con_layers= 2	Con_layers= 2	Con_layers= 2	Con_layers= 2	Con_layers= 2	Con_layers= 2
	Sub_layers= 2	Sub_layers= 2	Sub_layers= 2	Sub_layers= 2	Sub_layers= 2	Sub_layers= 2
	Ker_size= 5	Ker_size= 5	Ker_size= 5	Ker_size= 5	Ker_size= 8	Ker_size= 8
	l_r= 0.1	l_r= 0.06	l_r= 0.1	l_r= 0.06	l_r= 0.008	l_r= 0.008
	ep= 30	ep= 60	ep= 30	ep= 60	ep= 50	ep= 45
	outsamples= 6, 12	outsamples= 6, 12	outsamples= 8, 16	outsamples= 8, 16	outsamples= 10, 20	outsamples= 9, 18

Table 3.3 The performance of each hyper-parameter set of DBN on six datasets. Set 1 to set 6 are the hyper-parameters of DBN in Table. 3.2. The values express the classification results of each set of hyper-parameters on different datasets. The highest accuracies are highlighted.

DBN	2D&3D RGB	2D&3D depth	Object RGB	Object depth	NYU v1 RGB	NYU v1 depth
Set 1 (%)	**72.1**	67.3	71.6	72.2	39.7	36.5
Set 2 (%)	70.3	**75.7**	70.5	68.4	37.7	39.7
Set 3 (%)	69.2	72.0	**80.9**	74.6	38.3	37.2
Set 4 (%)	69.5	72.2	78.1	**75.1**	38.0	40.3
Set 5 (%)	52.1	44.5	60.2	55.9	**62.4**	49.3
Set 6 (%)	42.4	41.3	47.2	50.8	58.3	**57.3**

Table 3.4 The performance of each hyper-parameter set of SDAE on six datasets. Set 1 to set 6 are the hyper-parameters of SDAE in Table. 3.2. The values express the classification results of each set of hyper-parameters on different datasets. The highest accuracies are highlighted.

SDAE	2D&3D RGB	2D&3D depth	Object RGB	Object depth	NYU v1 RGB	NYU v1 depth
Set 1 (%)	**73.0**	74.2	76.3	68.4	43.1	29.6
Set 2 (%)	73.0	**74.2**	76.3	68.4	43.1	29.6
Set 3 (%)	71.2	72.5	**81.4**	66.3	45.0	32.2
Set 4 (%)	67.4	71.7	76.6	**71.9**	46.7	34.5
Set 5 (%)	36.4	40.1	51.4	39.2	**65.2**	47.8
Set 6 (%)	38.5	33.9	37.5	37.8	63.2	**51.5**

Table 3.5 The performance of each hyper-parameter set of CNNs on six datasets. Set 1 to set 6 are the hyper-parameters of CNNs in Table. 3.2. The values express the classification results of each set of hyper-parameters on different datasets. The highest accuracies are highlighted.

CNNs	2D&3D RGB	2D&3D depth	Object RGB	Object depth	NYU v1 RGB	NYU v1 depth
Set 1 (%)	**77.3**	76.1	74.5	68.3	37.2	31.3
Set 2 (%)	70.8	**81.0**	72.6	71.6	41.1	35.6
Set 3 (%)	72.4	68.7	**82.4**	72.6	39.8	36.9
Set 4 (%)	69.9	73.3	80.2	**75.5**	42.2	31.8
Set 5 (%)	50.0	44.5	52.7	60.1	**68.4**	44.2
Set 6 (%)	53.4	47.3	44.8	57.8	61.7	**56.5**

system is validated. Specifically, the nearer the complexity feature vectors are, the smaller the differences among these accuracies are. On the contrary, the farther the complexity feature vectors are, the bigger the differences among these accuracies are. The experimental results validate our system and can be considered as the successful tests. It can also be found in Table. 3.2 that hyper-parameter set 1 to set 4 or hyper-parameter set 5 and set 6 can easily approximate to the best hyper-parameter set with a fine tuning.

In this Chapter, we use the strategy that the same fixed time for hyper-parameter adjustment to select hyper-parameters. Currently, deep learning model is still a "black box" which has to depend on trial-and-error to adjust hyper-parameters. However, this optimization requires strong expertise. Indeed, the use of some global optimization techniques as genetic algorithms, harmony search [76], and particle swarm optimization [128] can result in higher classification accuracies. If the deep learning model has to face abundant data and has a long training time, it becomes impractical to use these global optimization techniques. For example, in our experiments, the test for one set of hyperparameters will cost over 12 hours. If the genetic algorithm initial population has 100 individuals, we will have to spend over 50 days to calculate the fitness, which makes the use of genetic algorithm unpractical. Therefore, we follow the tricks in Chapter 2.4.6 to adjust the hyperparameters manually.

3.6.5 Overall performance analysis

In this subsection, we combine subsection 3.6.3 and subsection 3.6.4 together to show a comprehensive analysis about the experiment. From the six experimental datasets, we select 2D&3D object RGB dataset as a new coming task. Each of other datasets is corresponding

to one vector in complexity feature vector set and one hyper-parameter set. Through the complexity measure system (10-dimensional) which is defined by us, we can use vector 1 to express the classification complexity of 2D&3D object RGB dataset. From the given relationship among vector 2 to vector 6 in Fig. 3.7, we then choose vector 4 which is the nearest vector towards vector 1 as the similar vector. The hyper-parameter set which is corresponding to vector 4 is considered as the initial set for 2D&3D object RGB dataset. From the performance of DBN in Table. 3.3, we can find that the accuracy of hyper-parameter set 4 on 2D&3D object RGB dataset is 69.5% which is a bit lower than the highest accuracy 72.1%. From Table. 3.2, we can find that, with a fine-tuning, it is easy to obtain the optimal set for 2D&3D object RGB dataset. According to the performance of SDAE in Table. 3.4, it can be found that the result of hyper-parameter set 4 on 2D&3D object RGB dataset is 67.4% which is a bit lower than the highest performance 72.1%. With a fine-tuning, we can obtain the optimal set for 2D&3D object RGB dataset. In CNNs method, the accuracy of hyper-parameter set 4 on 2D&3D object RGB dataset is 69.9% which is not far away from the highest accuracy 77.3%. Generally, in our experiment, when we select one dataset as a new coming task, it can be found that the accuracy of initial hyper-parameter set on this dataset is always only a bit lower than the highest accuracy. Meanwhile, it is easy to obtain the optimal set for the new coming task with a fine-tuning.

In summary, our experiments include six datasets and three deep learning models. According to the above analysis, the experiments prove to support our framework. In addition, if we can have more complexity feature vectors extracted from other datasets, the distance between complexity feature vector from the new coming task and the selected similar complexity feature vector will become shorter. In other words, the accuracy of initial hyper-parameter set on the new coming task will be higher.

3.7 Real-world scenario

Since it has been verified that our framework plays a significant role on hyper-parameter optimization among a small scale of datasets, to extent this work, we illustrate a scenario for further research around our framework. It needs other researchers to collect more complexity feature vectors from other datasets, which can make the selection of initial set of hyper-parameters closer to the optimal set. Meanwhile, more complexity feature vectors can speed up the optimization process. Furthermore, the collection of more datasets can make the hyper-parameter sets heterogeneous on different kinds of machine learning methods. Through the contribution from other researchers, deep learning will become easier to use under this framework in the future.

3.8 Summary

In this chapter, aim to tackle the challenges of the computationally expensive cost and initial hyper-parameter set choice in hyper-parameter optimization, we extend the work of classification complexity which previously is only applied to classifier selection, instance selection and prototype selection. This new framework is easy and practical which improves the efficiency and accuracy of hyper-parameter optimization by combining classification complexity and hyper-parameter optimization. We first select 10 complexity measures to define our complexity feature vector to assess complexity of a classification problem. Then, to verify our assumption about the relationship between selection of hyper-parameters and classification complexity, we choose three representative deep learning models on six real-world RGB-D datasets. After the analysis of experiments, we confirm that our framework can provide deeper insights into the relationship between dataset classification tasks and hyper-parameters optimization, thus quickly choosing an accurate initial set of hyper-parameters for a new coming classification task.

Chapter 4

Feature Learning for RGB-D Scene Classification

4.1 Overview

Indoor scene classification has received increasing attention in both academia and industry over the past few years. It plays an important role for a wide range of practical applications, *e.g.,* semantic recognition [23, 50, 141], content-based image indexing and retrieval [17, 73, 259] and mobile robots [267, 297]. In the real world situation, the intra-class variation of scenes is massive and the spatial layouts are vastly different. In addition, occlusion, low illumination, sophisticated background, and even different view angles can result in more challenges. Therefore, although much progress has been made, indoor scene recognition is still a challenging task.

CNNs obtain great success for high-level tasks, such as action recognition [43, 160], image classification [169, 245, 280] and object detection [79]. It improves the state-of-the-art performance on many important datasets (*e.g.*, the ImageNet dataset [56]), and even surpasses human performance on some datasets [261]. However, on the scene classification task, CNN features are still used rudimentarily. For example, Zhou et al. [294] simply collected a large-scale scene-centric dataset called "Places" to train Alexnet [136]. Then they directly extracted holistic CNN features from the model. Additionally, they simply combined the training set of Places-CNN and the training set of ImageNet-CNN to train a Hybrid-CNN for Hybrid features. Although the published performance can be improved through scene-centric CNNs, it mainly depends on the abundant training data and very deep networks. Therefore, it is considered that scene classification with Deep Neural Networks is still in its infancy.

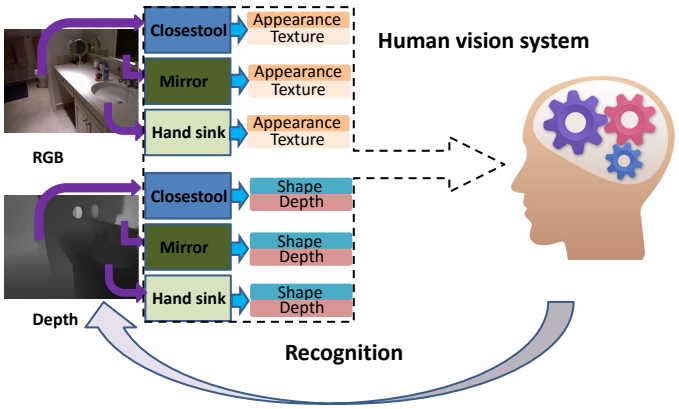


Fig. 4.1 Human vision system for scene classification in a natural environment. Humans firstly find some representative objects in these scenes. Then humans learn these scene categories through the mid-level object features. Consequently, humans can show better recognition performance on the similar scenes.

An indoor scene usually contains many different objects which can provide indirect clues for higher level tasks. Humans recognize the class of one unknown scene mainly relying on the object-level information. Given a new indoor image, for example, "Toilet", we can quickly recognize this scene category when we find some representative objects such as "closestool", "mirror" and "hand sink" in it. In addition, we will enhance the recognition accuracy on this kind of scenes when we see other similar scenes. How humans recognize a scene is illustrated in Fig. 4.1. Therefore, the scene categories represented by object information would reduce the variety among intra-class scene images and show larger discrimination among inter-class scene images. It makes constructing mid-level representations with discriminative object parts generally more useful than directly considering all pixels in the whole image for the scene classification task.

Recent developments in low-cost RGB-D sensors have opened a new dimension which can generate depth information from the surrounding environment. From the human vision perspective, when humans simultaneously obtain the appearance, texture and shape information of one object, it can help us improve the recognition accuracy. Though it has been proved that combining RGB and depth information in image/video classification can significantly improve the classification accuracy, it still needs a highly efficient method to fuse information from these two modalities to perform high-level reasoning.

There is no doubt that combining the advantages of Deep Convolutional Neural Networks, local scene features and RGB-D image information can help researchers design more sophisticated scene classification algorithms. Most of the methods which are proposed for indoor scene image classification using local information and depth data have something

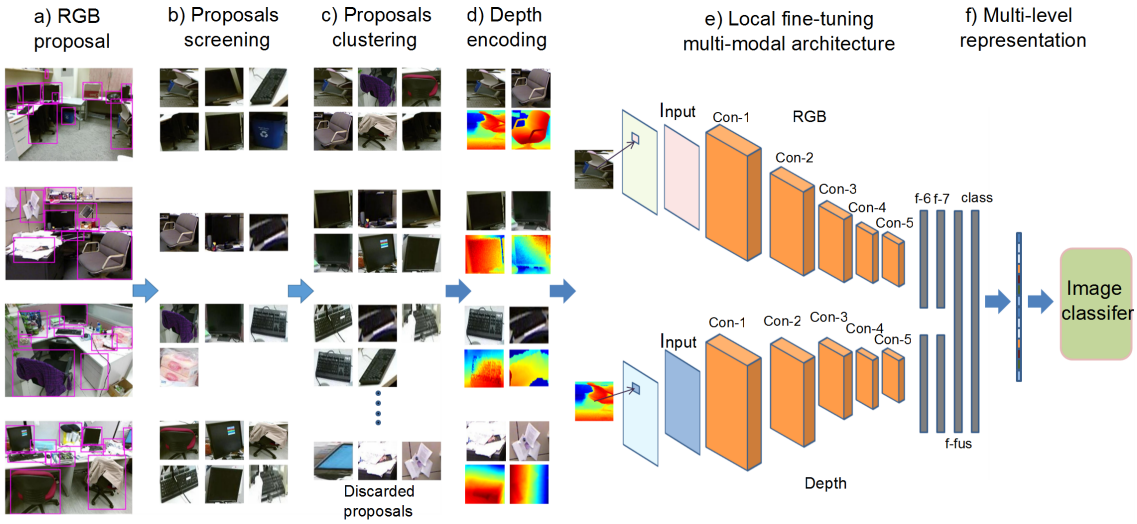


Fig. 4.2 The flow chart of the proposed pipeline. a) High-quality RGB region proposals are generated through Edge Boxes. We choose the top-ranked proposals. b) Unrepresentative region proposals are removed by one-class SVMs. c) These proposals are grouped into clusters through RIM. The discarded region proposals are considered in an extra cluster. d) Depth region proposals are then encoded into JET-style depth proposals. e) Multi-modal local fine-tuning architecture is performed on Caffe pre-trained network, which can decide the category of a test region. f) We train an image classifier for multi-level representation from region proposals. (See text for more details)

in common: RGB-D CNN features are firstly extracted at different locations and scales of an image separately. And then these learned features are simply concatenated as RGB-D features or encoded as a combined feature representation. At last, the feature representations are classified using a classifier such as SVM. Though results show that these kinds of feature representations are competitive and object-level information has the potential to improve scene classification, it still has one issue: according to scene images, since spatial aggregation performed by pooling layers in CNNs is too simple and does not hold much information about local feature distributions, the fundamental architecture of CNNs is not supposed to be most suitable for classifying scene images. Local feature distributions in the aggregated features are neglected when critical inferences happen in the fully connected layers near the top of CNNs. Meanwhile, the learning procedure cannot be adjusted mutually.

In order to address this issue, in this chapter, we propose an RGB-D local multi-modal feature learning method (LM-CNN) for scene classification. LM-CNN can effectively capture much of the local structure from the RGB-D scene images and automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities. The flow chart of our proposed method

is illustrated in Fig. 4.2. We firstly utilize one region proposal extraction method over experimental RGB-D datasets and do proposal screening on these generated region proposals to select the representative region proposals. Then we group these selected proposals into clusters and encode selected depth proposals. Following the human way of scene recognition, we perform the CNNs to understand objects in the early stage. Our local fine-tuning multi-modal network automatically learns to combine these two processing streams on an additional layer in a late fusion approach. At last, the multi-level scene image representation is built from top of the probability distribution of the region proposals. LM-CNN is described in detail in Section 4.3. The main contributions of this chapter are a novel pipeline for scene classification built on top of CNN features and a local fine-tuning multi-modal network using the representative proposals discovered from the target dataset.

The rest of this chapter is organized in the following way. In Section 4.2, we show the literature review. In Section 4.3, we give a detailed introduction of LM-CNN. Experimental setup and results on the verification details of our framework and the relevant experimental result analysis are comprehensively presented in Section 4.4. Finally, the summary is given in Section 4.5.

4.2 Literature Review

As for the global deep feature learning methods, Place-CNN [294] is the most successful deep feature learning model in scene classification. Place-CNN is trained on a large-scale scene-centric dataset with 205 scene categories and 2.5 million images with category labels using the well-known architecture Alexnet [136]. During the past few years, much work on classifying scene images using local deep learned information has been conducted. Gong et al. [83] presented a multi-scale orderless pooling scheme (MOP-CNN) which extracted CNN activations for local image patches at multiple scale levels. MOP-CNN performs orderless vector of locally aggregated descriptors (VLAD) pooling of these activations at each level separately, and then concatenates the features as the final feature representation. Yoo et al. [283] presented a straightforward framework for better image representation by combining low-level local descriptors and mid-level deep neural activations of CNNs. The proposed multi-scale pyramid pooling method can perform better utilization of neural activations from pre-trained CNNs. Liao et al. [159] developed a scene classification model with regularization of semantic segmentation based on the Alexnet, called SS-CNN, where the features learned for scene classification in SS-CNN automatically contain object-level information. Although these methods have made great progress in scene recognition tasks, they do not provide a natural solution to fuse with the depth information.

There are also several methods proposed for RGB-D data fusion. For example, Gupta et al. [92] proposed one algorithm which generalizes the $gPb-ucm$ approach [6] through making effective use of depth information. Bo et al. [21] proposed Hierarchical Matching Pursuit (HMP) to obtain abstract representations of RGB-D data, which builds feature hierarchies with an increasing receptive field layer by layer. Beyond this, Lai et al. [144] used $3D$ spin images and SIFT descriptors for depth features, and texton, color histogram and standard deviation of each color channel for RGB features. Socher et al. [229] presented a model which is based on the combination of CNNs and Recursive Neural Networks (RNNs) [230], but this model extracted features of RGB-D images separately. Song et al. [234] proposed an approach which made a $3D$ volumetric scene from RGB-D images as inputs and a $3D$ object bounding box as output through Region Proposal Network (RPN) to learn objectness and a joint $2D+3D$ object recognition network to extract geometric features in $3D$ and color features in $2D$. However, the above mentioned methods have not explored the correlation and complementarity between raw RGB and depth images. Most of the methods just learn features from RGB and depth separately and then simply concatenate them as RGB-D features or encode these two kinds of features. The major disadvantage is that the correlation and complementary property between RGB and depth are ignored, and the learning procedure cannot be adjusted mutually.

In contrast, the proposed CNNs-based local multi-modal feature learning framework (LM-CNN) in this chapter can effectively capture much of the local structure from the RGB-D scene images and automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities.

4.3 Methodology

In this section, we introduce our LM-CNN in detail. The pipeline can be implemented as follows. As we mentioned in previous sections, our pipeline is built on the top of pre-trained CNNs. We firstly choose the state-of-the-art region proposal generating method which is most suitable for our pipeline to do region proposal extraction on our RGB-D datasets (In Subsection 4.3.1). The reasons of this step are as following: 1) The recent scene classification work still uses the CNN features rudimentarily (such as the work in [294]). The first step of our proposed pipeline is to do region proposal extraction on the RGB-D datasets. An end-to-end CNNs method is considered that it may not be best suited for classifying images, especially scene images, where local features follow a complex distribution. According to scene images, since spatial aggregation performed by pooling layers in CNNs is too simple and does not hold much information about local feature distributions, the fundamental ar-

chitecture of CNNs is not supposed to be most suitable for classifying scene images. Local feature distributions in the aggregated features are neglected when critical inferences happen in the fully connected layers near the top of CNNs. Meanwhile, according to the whole image, it shows that it is consistently better to extract CNN features from local region proposals arranged in regular grids [83]. 2) In addition, the interaction among different objects in the whole scene images, such as occlusion, shows a challenge. Therefore, applying the end-to-end CNN structure directly for scene image classification is not feasible. Then we do proposal screening on these generated region proposals for the selection of representative region proposals (In Subsection 4.3.2). After obtaining the discriminative region proposals, we group these proposals into clusters through an approach called Regularized Information Maximization (RIM) [134] (In Subsection 4.3.3). Before we apply local fine-tuning of the multi-modal model on the above grouped region proposals, an RGB to depth encoding algorithm is performed over these proposals (In Subsection 4.3.4). Our local multi-modal fine-tuning model is introduced in Subsection 4.3.5. At last, the multi-level scene image representation can be built from the top of the probability distribution of region proposals (In Subsection 4.3.6).

4.3.1 RGB-D region proposal extraction

Generating region proposals aims to obtain a set of relative bounding boxes which try to contain all objects of the image. It has wide applications such as efficient object detection [253, 268] and weakly supervised learning [58, 226]. Currently, many approaches have been proposed for generating region proposals including BING [42], MCG [7] and Edge Boxes [299]. BING trains a simple linear classifier over edge features, and then this classifier is applied in a sliding window manner. After this, a very fast agnostic detector can be obtained. MCG combines the advantages of two leading methods for generating proposals (gPbUCM [91] and CPMC [33]). Moreover, MCG proposes an improved hierarchical segmentation, a new method to generate proposals and a new ranking procedure. Edge boxes is similar to BING, but it uses object boundaries as features for the scoring. In our pipeline, the quality of the extracted region proposals plays an important role. The effective region proposals should satisfy three criteria: *high recall rate*, *few number of proposals* and *tolerable evaluation speed*. One paper which evaluates ten publicly available detection proposal methods has proposed that only Edge Boxes maintains good performance on above three criteria [107]. Therefore, we choose Edge Boxes to generate high-quality RGB region proposals in our pipeline. Meanwhile, region proposals from hierarchical image segmentation [6] are also used. The corresponding depth region proposals can be acquired through cropping the depth scene images into depth region proposals along the location of the RGB region proposals

Fig. 4.3 Example images about RGB-D region proposals from the SUN RGB-D dataset and the NYU Depth v1 dataset. Images on the same row belong to the same cluster. First, third, fifth and seventh rows are the RGB region proposals. Second, fourth, sixth and eighth rows are the corresponding depth region proposals.

on the RGB scene images. Some example images about RGB-D region proposals can be found in Fig. 4.3. We choose ImageNet-CNN features [136] which are learned from the pre-trained Caffe model [123] on image classification dataset (*i.e.* ImageNet) for all the image region proposals. The feature dimension after CNNs in the FC7 layer is 4096.

4.3.2 Region proposal screening

In practice, we can consider that there exist some representative region proposals in each scene category. Some other unrepresentative region proposals may also appear in this scene category, but only few images contain these region proposals. We consider these unrepresentative region proposals as outliers. Motivated by [210], we use one-class SVM to remove these unrepresentative region proposals and then estimate the discriminative power among

scene categories for each region proposal. A one-class SVM can separate the data from the origin to remove unrepresentative region proposals. The region proposal screening is a separate part in our pipeline. After the discriminative region proposals are obtained, these region proposals are sent into the local fine-tuning multi-modal architecture. Let $x_1, x_2, \cdots, x_n$ be the region proposals from one class, and kernel mapping φ: $X \to H$ maps original region features into another feature space. We solve the following optimization:

$$\begin{aligned} &\min_{u,\lambda,\eta} \frac{1}{2}\|u\|^2 + \frac{1}{vn}\sum_{i=1}^{n}\lambda_i - \eta, \\ &s.t.\ (u \cdot \varphi(x_i)) \geq \eta - \lambda_i, \lambda_i \geq 0, i \in (1,2,\cdots,n), \end{aligned} \tag{4.1}$$

where $v \in (0,1]$ controls the ratio of outliers. The decision function can be obtained:

$$f(x) = \text{sign}(u \cdot \varphi(x_i) - \eta), \tag{4.2}$$

which returns the positive sign when given the representative region proposals and returns the negative sign when given the outliers. Aiming to achieve better performance, we use three cascaded classifiers. We define that each of the classifiers labels 15% of the input region proposals as unrepresentative proposals and prune them.

According to our hypothesis, since each image I_i can be expressed as several region proposals, we define each region proposal from I_i as r_j^i, where i is the number of input images and j is the number of representative region proposals. Meanwhile, we use y_i to express the labels of the input scene images. In a natural world, a discriminative region proposal should usually appear in one scene category but unusually appear in other scene categories. Following this, we give the discriminative power for each region proposal among scene categories. The discriminative power is from 0 to 1. We can also consider it as the weight W_j^i of each region proposal. W_j^i can be expressed as follows:

$$W_j^i = P(y_i \mid r_j^i) = \frac{P(r_j^i, y_i)}{P(r_j^i)} \approx \frac{K_y}{K}, \tag{4.3}$$

where K_y is the number of region proposals among the K nearest neighbors which share the same scene labels with r_j. We set K as 100 in all our experiments. Table 4.1 gives the distribution of region proposal weights after the screening.

Table 4.1 Region proposal weight distribution.

Weight	[0,0.2]	(0.2,0.4]	(0.4,0.6]	(0.6,0.8]	(0.8,1]
Percentage	34.5%	26.1%	16.2%	16.0%	7.2%

4.3.3 Discriminative region proposals clustering

After obtaining the discriminative region proposals, we group these proposals into clusters. It can help us discover the relationship between scene category labels and region proposal labels. [134] proposed a framework called Regularized Information Maximization (RIM) which can simultaneously cluster the data and train a discriminative classifier. This algorithm contains the optimization of an intuitive information theoretic object function which strikes a balance among class separation, class balance and cluster complexity. As we can see from Fig. 4.3, the region proposals after clustering have the similar appearance and semantic meaning. Till now, we can obtain the region proposal clusters of RGB images. To acquire the corresponding depth region proposals, we crop the depth scene images into depth region proposals according to the locations of the RGB region proposals on the RGB scene images. Fig. 4.3 shows some RGB-D scene images and RGB-D region proposal pairs from the SUN RGB-D dataset and the NYU Depth v1 dataset.

4.3.4 Depth region proposal encoding

After we have obtained the depth region proposals, unlike RGB images, these depth images cannot be directly used as inputs for the CNNs. To solve this problem, one encoding method called HHA [93] has been proposed. HHA encodes the depth image into 0 to 255 range with three channels at each pixel and emphasizes complementary discontinuities in the image (depth, surface normal and height). In this step, we employ the MATLAB jet colormap which is effective and computationally cheap. Jet firstly normalizes the depth values between 0 to 255, then the normalized image is transformed from a single channel to a three channel image by a jet color map. For the pixels on the depth image, the distance to color values is mapped from red (near) over green to blue (far). Some examples of the encoded images from the dataset can be found in Fig. 4.4. In our experiments (Section 4.4.4), it proves that Jet encoding outperforms HHA encoding for our method.

4.3.5 Local fine-tuning of multi-modal architecture

After we obtain the RGB region proposals and their corresponding depth region proposals, we choose to fine-tune the pre-trained CNNs on these RGB-D region proposals. During

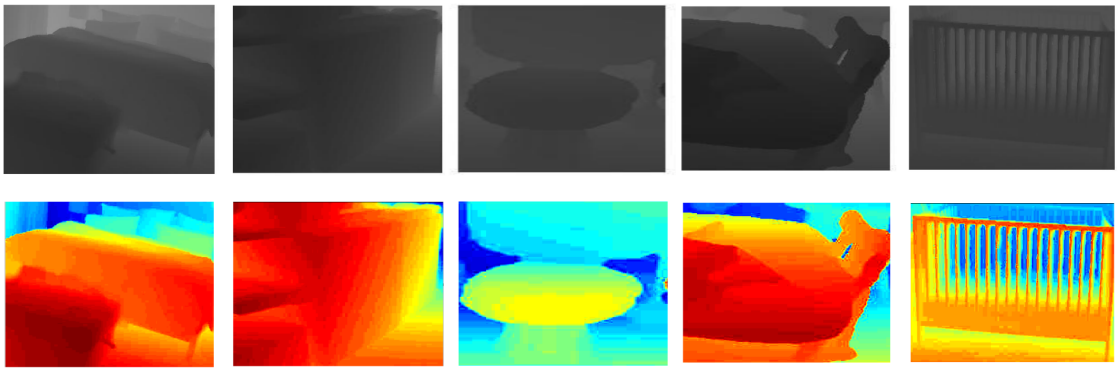


Fig. 4.4 Some examples of the Jet encoded images from the dataset. Images on the first row are the depth region proposals. Images on the second row are the Jet encoded depth region proposals.

the training phase, all the original RGB/Depth images are randomly cropped into 227×227 pixels. These cropped images are the inputs of the Caffe model. The used CNNs contain five fully convolutional layers, 60 million parameters and three fully-connected layers. The proposed architecture is shown in Fig. 4.2. It contains two streams - which process the RGB data and depth data independently, and then are combined in a late fusion approach. Each stream contains one Caffe model implementation of the CNNs which are pre-trained for object classification on the ImageNet dataset. Then we fine-tune the CNNs on our region proposal categories. The details about the CNNs architecture can be found in [136].

We have defined each RGB region proposal as Rr_j^i and each corresponding depth region proposal as Dr_j^i. The region proposal label can be expressed as y_i. In each individual stream, we choose to use the pre-trained Alexnet on the large-scale image classification dataset (ImageNet). All parameters including the weights and biases P^R from RGB region proposals and P^D from depth region proposals are initialized from the Caffe trained network on the ImageNet dataset. We then train the two streams separately through putting a randomly initialized softmax classification layer on the top of the RGB and depth layers. For the RGB or depth region proposal image stream network, we minimize the negative log likelihood $\mathscr{L}$ of the training data. They can be expressed as:

$$\min_{W^R,P^R} \sum_{i=1}^{N} \mathscr{L}(softmax(W^R g^R(r^i;P^R)),y_i), \tag{4.4}$$

$$\min_{W^D,P^D} \sum_{i=1}^{N} \mathscr{L}(softmax(W^D g^D(d^i;P^D)),y_i), \tag{4.5}$$

where W^R and W^D are the weights on the softmax layer, which map from $g(\cdot)$ to $\mathbb{R}^M$.

$g^R(r^i;P^R)$ and $g^D(d^i;P^D)$ are the representations of RGB's last fully connected layer and depth's last fully connected layer separately. The loss is expressed as:

$$\mathscr{L}(s,y) = \sum_k y_k \log s_k. \tag{4.6}$$

The softmax function is defined as:

$$softmax(z) = \frac{exp(z)}{\|z\|}. \tag{4.7}$$

After these two stream networks are trained, we firstly discard the softmax weights and then concatenate them. We fine-tune the responses $g^R(r^i;P^R)$ and $g^D(d^i;P^D)$ as we mentioned in Eq. (4.4) and Eq. (4.5). At last, a fusion stream $f([g^R(r^i;P^R),g^D(d^i;P^D)];P^F)$ is used for the last layer responses. Same with RGB and depth individual networks, RGB-D fusion network also ends at the softmax layer. All weights of the network are learned with a fixed momentum (set to 0.9). The dropout ratio of the fully-connected layer is set to 0.5. To avoid over-fitting, we initialize the learning rate to 0.01 and make it reduce to 0.001 after $20k$ iterations. It finally stops at $50k$ iterations. For the fusion network, we train this through jointly optimizing the parameters to minimize the negative log likelihood:

$$\min_{W^f,P^R,P^D,P^f} \sum_{i=1}^{N} \mathscr{L}(softmax(W^f f([\boldsymbol{g}^R,\boldsymbol{g}^D];P^f),y_i), \tag{4.8}$$

where $\boldsymbol{g}^R$ and $\boldsymbol{g}^D$ are the representations of RGB-D's last fully connected layers. W^f is the weight of the softmax layer, and P^f is the parameter from RGB-D fusion region proposals.

Note that we obtain **N** (around several hundreds) region proposal classes during the clustering step, but we change the output layer of the ImageNet 1000-way classification into (**N+1**)-way classification. Meanwhile, other layers remain unchanged. The extra way means the region proposals which are thrown away in the screening step. It can make the region proposal classifier robust to the noisy labels.

4.3.6 Multi-level representation from region proposals

After N RGB-D region proposal clusters have been learned in the discriminative clustering step, the recognition of a given RGB-D image pair can be summarized as follows. We firstly perform the EdgeBoxes method on the given image pair to generate the region proposals. Each RGB-D region proposal can be classified into one pair of the clusters through the region proposal classifier. Since it has been proved that Spatial Pyramid Matching (SPM) [278] and modified Vector of Locally Aggregated Descriptors (VLAD) [5, 120] are suc-

cessful, we decide to choose both of these two methods. According to SPM, we use three levels of SPM. We choose the center of all region proposal clusters which falls into one SPM region as splitting center in our experiments. Then we can obtain a hierarchical histogram of region proposal labels for this image. It is then used for the classification of the coming image. The difference between modified VLAD and VLAD is that modified VLAD does not use K-means clustering, and modified VLAD chooses discriminative region proposal clusters as the clusters for VLAD. Then we can obtain n clusters for RGB-D images separately. Each region proposal of the test image is assigned to its nearest cluster center resulting in a 4096-d vector per cluster. Then each 4096-d is reduced into $4096/2n$-d vector through PCA. At last, we can obtain a 4096-d VLAD descriptor through concatenating these $4096/2n$-d vectors.

Another kind of feature we consider is hybrid Places feature which is first mentioned in [294]. It is learned from over 2.5 million labelled pictures of scenes and combines local and global information of these scene images. In our experiments, RGB hybrid Places image features and depth hybrid Places image features are obtained separately.

At last, we concatenate normalized VLAD/SPM features and the RGB-D hybrid Places features to train a network with two hidden layers. The image representation can be expressed as a concatenation of all of the feature vectors. The activation function of the neurons during the fully-connected hidden layers is rectified linear function (ReLU).

4.4 Experimental results

We evaluate our LM-CNN along with current state of the art on the NYU Depth v1 dataset and the SUN RGB-D dataset. Both of these two datasets are derived from the publicly available RGB-D sensor-based scene database. In our experiments, we not only compare our method with hand-crafted methods such as GIST [191], but also deep feature learning models such as Alexnet, VGG [224] and some other representative models. The Places-CNN [294] scene features which are learned through Alexnet or VGG use the Places dataset for model pre-training. The Places2-CNN scene features use the Places2 dataset which has much more scene images than the Places dataset for model pre-training. Details of the datasets and experimental setup are provided below.

4.4.1 Datasets

NYU Depth v1

To evaluate our proposed pipeline, we firstly conduct experiments on the NYU Depth v1 dataset. Since the standard classification protocol removes scene 'cafe' from the dataset, we use the remaining 6 different scenes. Since there are so many objects in one scene and the correlation between images in one scene is low, it makes NYU Depth v1 a very challenging dataset.

SUN RGB-D

We also test our RGB-D local multi-modal scene classification pipeline on SUN RGB-D dataset. Following the setup in [231], we split the dataset to ensure approximate half for training and half for testing in each sensor. These images which are captured from the same building with similar furniture styles are not spread across both of the training and testing sets. In other words, the images in SUN RGB-D dataset are either all go into the training set or the test set. To the best of our knowledge, the SUN RGB-D dataset is the largest and most challenging RGB-D scene dataset currently. The baseline when using GIST and RBF kernel SVM only gives a classification accuracy of 23%.

4.4.2 Experiment Setup

The first experiment about the proposed LM-CNN is on the NYU Depth v1 dataset. During the experiment, we obtain 131 top ranked RGB region proposals through Edge Boxes from each image. Meanwhile, we also generate $32(96)$ region proposals in the top (bottom) level by hierarchical image segmentation from each image. After region proposal screening, 15% region proposals are removed as unrepresentative region proposals. Then we do discriminative clustering on these screened region proposals resulting in $70(30)$ proposal classes. To make the region proposal classifier robust to the noisy labels, the discarded region proposals in the screening step are considered as the 71-th proposal class. The corresponding depth region proposals can be acquired through cropping the depth scene images into depth region proposals along the location of these RGB region proposals on the RGB scene images. The whole local fine-tuning fusion procedure is performed on the famous public Caffe toolbox. We use the pre-trained model of the large-scale image classification dataset (ImageNet). The layers from both of the stream networks are initialized from the pre-trained eight layers. The softmax layer is discarded. Then we concatenate the softmax layer of the two individual networks. The output layer on the ImageNet 1000-way classification is changed

into a new output layer of 71-way (31-way) classification. In our multi-level representation step, through modified VLAD and Hybrid CNN, each image can be expressed as concatenated normalized VLAD features and RGB-D Hybrid Places features of the whole image. At last, the image classification step is completed over one NN network with two layers (200 nodes for each).

We compare LM-CNN with state-of-the-art methods including: 1) R. Socher et al. [229]: using the combination of CNNs and RNNs; 2) Le et al. [147]: using robust soft reconstruction cost for ICA; 3) Wang et al. [266]: using locality constraint to select similar basis of local image descriptors; 4) Bo et al. [21]: using transfer learning based method; 5) Jin et al. [124]: using self-trained CNNs and LLC; 6) Zhou et al. [294]: using Places2 dataset for pre-training, and fine-tuning on the training set of NYU depth v1 dataset; 7) Zhou et al. [294]: using both Places2 dataset and ImageNet dataset for pre-training, and fine-tuning on the training set of NYU depth v1 dataset. The comparison results are shown in Table 4.2. It can be seen that LM-CNN achieves the best performance and outperforms state-of-the-art methods in the RGB-D scene classification task. Fig. 4.5 shows the confusion matrix about our method across 6 classes on the NYU Depth v1 dataset, the diagonal elements of which represent the recognition accuracy for each category.

Table 4.2 The comparison results of our method and other published methods on the NYU Depth v1 dataset.

Method	RGB (%)	Depth (%)	RGB-D (%)
CNN-RNN [229]	73.5	65.2	75.7
RICA [147]	74.5	64.7	74.5
LLC [266]	66.1	61.5	66.3
SPM [278]	52.8	53.2	63.4
HMP-S [21]	72.6	63.9	72.8
CNNs+LLC [124]	73.1	61.8	75.4
Places2-CNNs [294]	74.5	66.9	76.8
Hybrid-CNNs [294]	74.1	**68.2**	77.4
LM-CNN	**74.7**	67.8	**79.3**

In Table 4.2, the performance of our method achieves 79.3% on RGB-D image pairs, which outperforms published methods, *i.e.*, RICA, LLC, SPM, CNNs+LLC, Places2-CNNs and Hybrid-CNNs by 4.8%, 13.0%, 15.9%, 3.9%, 2.5% and 1.9% respectively. The main reason is that deep learned features are indeed able to achieve higher performance than hand-crafted features when the number of training samples is large enough. However, the individual RGB or depth performance through our method hardly shows any advantages compared to individual performance by Places2-CNNs and Hybrid-CNNs. It proves that

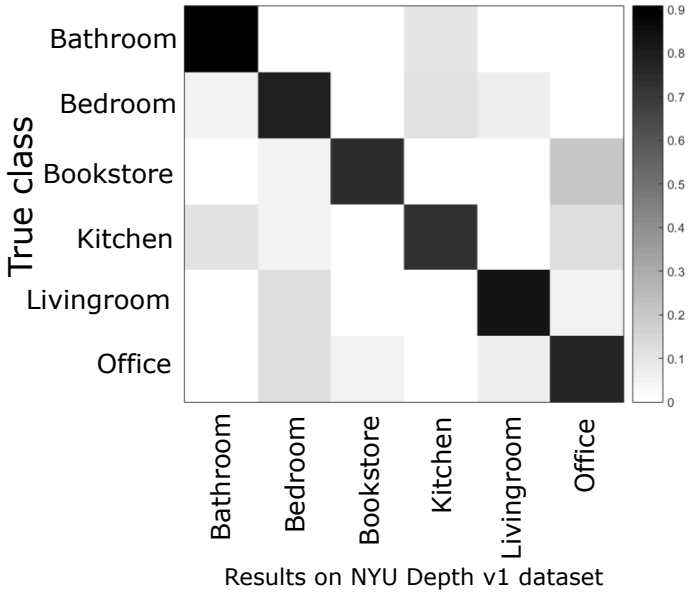


Fig. 4.5 Confusion matrix about our method on the NYU Depth v1 dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

our local fine-tuning multi-modal fusion architecture plays an important role in LM-CNN.

The second experiment is on the SUN RGB-D dataset. During the experiment, similar with the NYU Depth v1 dataset, we firstly obtain 103 top ranked RGB region proposals through Edge Boxes from each image. 32 (96) region proposals are generated in the top (bottom) level by hierarchical image segmentation from each image. Following region proposals screening, 15% region proposals are removed as unrepresentative region proposals. Then we do discriminative clustering on these screened region proposals resulting in 190 (60) proposal classes. The discarded region proposals in the screening step are considered as the 191-th proposal class. The corresponding depth region proposals are acquired through cropping the depth scene images into depth region proposals along the location of these RGB region proposals on the RGB scene images. We use the pre-trained Caffe model of the large-scale image classification dataset (ImageNet). The layers from both of the stream networks are initialized from the pre-trained eight layers. The softmax layer is discarded. Then we concatenate the softmax layer of the two individual networks. The output layer on the ImageNet 1000-way classification is changed into a new output layer of 191-way (61-way) classification. In our multi-level representation step, each image can be expressed as concatenated normalized VLAD features and RGB-D Hybrid Places features of the whole image. At last, a neural network with two fully-connected layers (200 nodes for each) is trained for image-level classification.

LM-CNN is compared with state-of-the-art methods including: 1) R. Socher et al. [229]: using the combination of CNNs and RNNs; 2) Oliva et al. [191]: using GIST features and RBF kernel SVM; 3) Zhou et al. [294]: using Places-CNN features and linear SVM; 4) Zhou et al. [294]: using Places-CNN features and RBF kernel SVM; 5) Liao et al. [159]: using the original Alexnet trained with only the SUN RGB-D dataset; 6) Liao et al. [159]: using Places-CNNs and object-level information. Moreover, we compare our method with two classical models: Places2-CNNs+softmax+Alexnet and Places2-CNNs+softmax+VGG [224]. However, according to the RGB-D image pairs, we simply concatenate RGB features and depth features. The comparison results are shown in Table. 4.3. It can be seen that our method achieves the best performance and outperforms state-of-the-art methods in the RGB-D scene classification task. Fig. 4.6 shows the confusion matrix about our method across 19 classes on the SUN RGB-D dataset, the diagonal elements of which represent the recognition accuracy for each category.

Table 4.3 The comparison results of our method and other published methods on the Sun RGB-D dataset.

Method	RGB (%)	Depth (%)	RGB-D (%)
CNN-RNN [229]	35.6	26.1	39.2
GIST+RBF kernel SVM [191]	19.7	20.1	23.0
Places-CNN+Linear SVM [294]	35.6	25.5	37.2
Places-CNN+RBF kernel SVM [294]	38.1	27.7	39.0
SUN RGB-D+Alexnet [159]	24.3	-	30.7
SS-CNN-R6 [159]	36.1	-	41.3
Places2-CNNs+softmax+Alexnet	41.7	32.1	42.3
Places2-CNNs+softmax+VGG [224]	43.5	**34.7**	45.1
LM-CNN	**44.3**	34.6	**47.2**

Table 4.3 shows that the performance of LM-CNN achieves 47.2% on RGB-D image pairs. It outperforms published methods, *i.e.*, GIST+RBF kernel SVM, Places-CNN+Linear SVM, Places-CNN+RBF kernel SVM, Alexnet and SS-CNN-R6 by 24.2%, 10.0%, 8.2%, 16.5%, 5.9%, 4.9% and 2.1%, respectively. Besides, our method also outperforms two popular global fine-tuned models: Alexnet and VGG, which are both pre-trained on the Place2 scene dataset. Similar with the NYU v1 depth dataset, the individual RGB or depth performance through our method hardly shows any advantages compared to individual performance by Places2-CNNs+softmax+Alexnet and Places2-CNNs+softmax+VGG. It proves that our architecture is efficient as a whole.

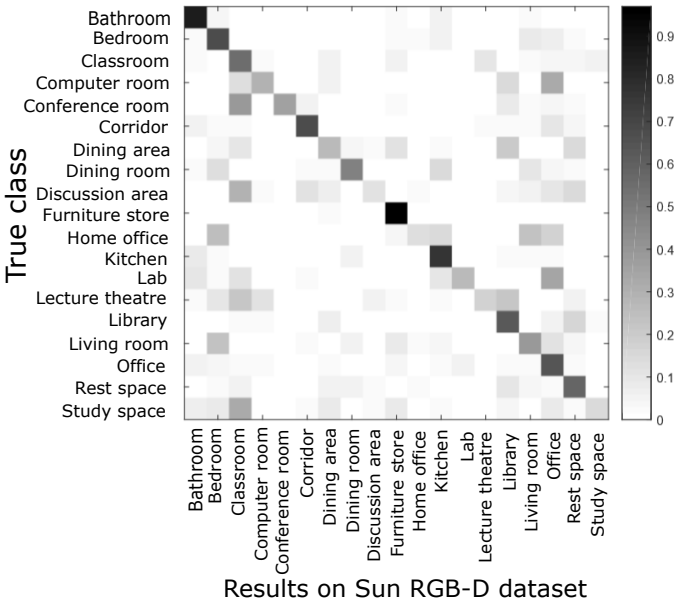


Fig. 4.6 Confusion matrix about our method on the SUN RGB-D dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

4.4.3 Global and Local Fine-tuning Discussions

To show the advantages of our local fine-tuning method, we perform some additional global fine-tuning experiments on the NYU v1 depth dataset and the SUN RGB-D dataset for image classification. These experiments focus on global fine-tuning and use a pre-trained deep network, which consider the entire images as the inputs and rely on the neural-network itself to learn the information when a new dataset comes. We set up these experiments with Alexnet, different pre-trained models and different classifiers. Studies on CNNs, for example GoogLeNet [244], indicate that using deeper models can more substantially improve classification performance than shallow models. Since our method is only based on Alexnet, we ignore other deeper networks, *i.e.,* VGG net [224] and Googlenet [244], and choose Alexnet as the network. Three different large-scale datasets are ImageNet, Places and Places2 which are pre-trained on Alexnet respectively. At the last step, we choose CNN features+SVM or softmax for image classification. In total, there are six different combinations in our experiment: ImageNet-CNNs features + Alexnet + SVM, ImageNet-CNNs + Alexnet + softmax, Places-CNNs features + Alexnet + SVM, Places-CNNs + Alexnet + softmax, Places2-CNNs features + Alexnet + SVM and Places2-CNNs + Alexnet + softmax. The comparison results are shown in Table 4.4.

As we can see from Table 4.4, our method outperforms other global fine-tuning methods

Table 4.4 The comparison results of global methods and our method on the NYU Depth v1 dataset and the SUN RGB-D dataset.

Methods		NYU Depth v1 dataset			SUN RGB-D dataset		
		RGB (%)	Depth (%)	RGB-D (%)	RGB (%)	Depth (%)	RGB-D (%)
Global	ImageNet-CNNs features + Alexnet + SVM	71.2	59.1	72.0	27.4	23.1	30.3
	ImageNet-CNNs + Alexnet + softmax	69.8	58.7	71.9	28.9	22.3	31.4
	Places-CNNs features + Alexnet + SVM	72.6	62.2	74.7	35.6	25.5	37.2
	Places-CNNs + Alexnet + softmax	72.4	64.3	73.8	36.2	24.6	38.7
	Places2-CNNs features + Alexnet + SVM	73.8	67.1	76.5	41.9	31.8	42.1
	Places2-CNNs + Alexnet + softmax	74.5	66.9	76.8	41.7	32.1	42.3
Local	**Our method**	**74.7**	**67.8**	**79.3**	**44.3**	**34.6**	**47.2**

by 4.9% at least. It indicates the advantages of our local fine-tuning pipeline of RGB-D region proposals and performing classification on top of them. According to other six global experiments, under the same dataset pre-trained Alexnet, different classifier choices result in little difference on classification performance. Therefore, the choice of the dataset for pre-training is mostly responsible for the classification accuracy. Following the same architecture as the network proposed in [136], the ImageNet dataset (ILSVRC 2012) for pre-training [56] contains 1.2 million widely various high-resolution images with 1000 different classes. Compared to the ImageNet dataset, Places is a scene-centric dataset with 205 scene categories and 2.5 million images with category labels, and Places2 is also a scene-centric dataset but with 8 million images from 401 scene categories. From the global results in Table 4.4, we can find that Places-CNN and Places2-CNN perform much better. It proves that a CNN network trained using a scene-centric dataset is able to achieve a significant improvement on a scene benchmark in comparison with a network trained using an object-centric dataset.

4.4.4 Ablation study

In this section, we analyze the effectiveness of individual components in our pipeline. We discard one single component (*i.e.* region proposal screening, clustering, depth encoding and local fine-tuning) and keep other components untouched. Table 4.5 shows a summary of the comparison results on NYU Depth v1 dataset and SUN RGB-D dataset. The final results of our full pipeline are also shown. We conduct a comprehensive analysis of these comparison results in the following sections.

Table 4.5 Evaluation results of ablation studies on NYU Depth v1 and SUN RGB-D datasets.

Configuration	NYU Depth v1 dataset			SUN RGB-D dataset		
	RGB (%)	Depth (%)	RGB-D (%)	RGB (%)	Depth (%)	RGB-D (%)
Without region proposal screening	73.9	67.4	77.8	41.9	33.7	42.8
Without discriminative region proposals clustering	73.7	67.2	77.3	41.7	33.4	42.5
Without depth region proposal encoding	74.7	57.2	74.9	44.3	21.2	44.6
Without local fine-tuning	72.5	66.4	76.9	40.1	29.7	41.6
Our full pipeline LM-CNN	**74.7**	**67.8**	**79.3**	**44.3**	**34.6**	**47.2**

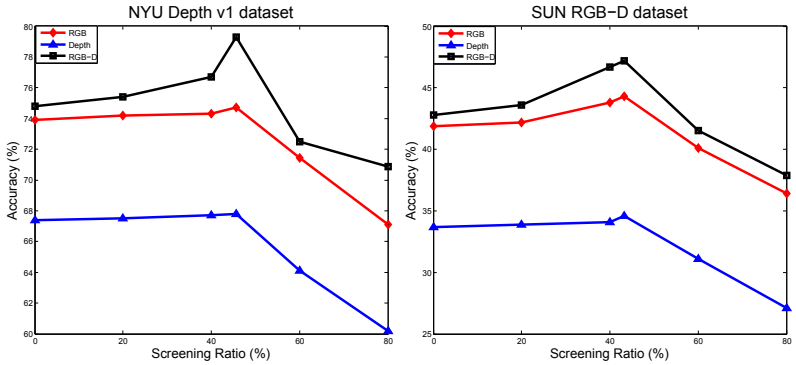


Fig. 4.7 Recognition accuracies with different screening ratios on NYU Depth v1 and SUN RGB-D.

Without region proposal screening

We directly feed all the region proposals without any screening into the subsequent components (clustering, depth encoding and local fine-tuning). During our region proposal screening step, we discard the region proposals with lower weights (see Eq. (4.3)). In this case, we define the screening ratio as the percentage of discarded region proposals in the screening step. It can be seen from Fig. 4.7 that when the screening ratio is 0, the recognition accuracies of RGB, Depth and RGB-D in NYU Depth v1 dataset are 73.9% 67.4% and 77.8%, in SUN RGB-D dataset are 41.9% 33.7% and 42.8%. All of the recognition accuracies without region proposal screening are lower than those with the full pipeline (Table 4.5). This is because that, although we have obtained reasonable region proposals through the Edge Boxes method, there still exist some proposals which are either false positives or unrepresentative objects shared by scene categories. On the contrary, a screening ratio that is too high will also result in a low recognition performance, because too high ratios will discard some discriminative region proposals. An optimal ratio can be obtained through cross validation on a subset of the training data.

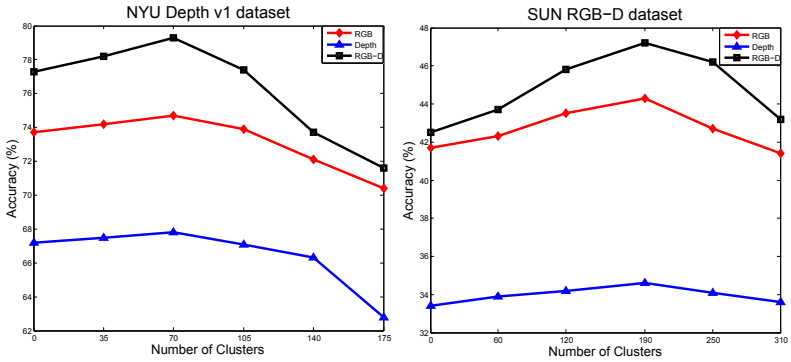


Fig. 4.8 Recognition accuracies with different number of clusters on NYU Depth v1 and SUN RGB-D.

Without discriminative region proposals clustering

The collected screened region proposals are directly considered as a large codebook, and each region proposal is treated as a visual word without region proposals clustering. Then we utilize LSAQ [164] (100 nearest neighbors) coding and SPM pooling for building an imagelevel representation of RGB and depth images. The recognition performances on both of NYU Depth v1 and SUN RGB-D datasets are lower than the performances of our method (Table 4.5), which illustrates that the discriminative region proposals clustering step in our pipeline is crucial. In this component, the common semantic meaning shared among similar proposals is emphasized, and the less important differences among them are tolerated. Therefore, discriminative region proposals clustering can improve the generality and representativeness of the discovered region proposals. Fig. 4.8 shows the recognition accuracies with different number of clusters on NYU Depth v1 and SUN RGB-D. It shows that an overly small number of clusters will result in the assignation of different semantic meanings into the same discriminative region proposals, which leads to a poor performance. On the other hand, too many clusters will also result in low recognition accuracies due to the poor generality of the semantic meanings of discriminative region proposals.

Without depth region proposal encoding

After obtaining the depth region proposals, we directly use these depth images as inputs for our multi-modal local fine-tuning architecture. It can be seen from Table 4.5 that the recognition accuracies of depth and RGB-D become low without depth encoding. In addition, we conduct experiments to compare two different depth encoding methods (HHA and Jet) described in section 3 (D). Both of these two encoding methods produce colorized images. Compared to HHA encoding which requires additional image preprocessing, the

Table 4.6 Comparison of different depth encoding methods on NYU Depth v1 and SUN RGB-D datasets.

Depth Encoding	NYU Depth v1 dataset		SUN RGB-D dataset	
	Depth (%)	RGB-D (%)	Depth (%)	RGB-D (%)
Without encoding	57.2	74.9	21.2	44.6
HHA	66.9	78.3	34.2	47.1
Jet	**67.8**	**79.3**	**34.6**	**47.2**

colorizing depth process of Jet encoding has negligible computational overhead. From the results, presented in Table 4.6, it is clear that using Jet encoding method yields slightly better performance than the HHA encoding method.

Without local fine-tuning

We directly utilize the responses from the RIM clustering model for pooling without local fine-tuning. From the comparison results of our experiments on NYU Depth v1 dataset in Table. 4.5, the recognition rate of RGB-D without local fine-tuning is 76.9%, which is around 2.4% lower than that with local fine-tuning. Similarly, the recognition rate of RGB-D on SUN RGB-D dataset without local fine-tuning is 41.6%, which is around 5.6% lower than that with local fine-tuning. This illustrates that in our pipeline, local fine-tuning is consistent with the common sense, which can better define separation boundaries between clusters.

4.5 Summary

In this chapter, we present a CNNs-based local multi-modal framework for RGB-D scene classification. Our pipeline is built on the top of the pre-trained CNNs model. We firstly perform a region proposal extraction method on an RGB-D dataset. Then we apply proposal screening on these generated region proposals to select the representative region proposals and group these selected proposals into clusters through the RIM algorithm. Aiming to leverage large CNNs trained for proposal recognition on the ImageNet dataset, we use an effective encoding method from depth to image data. Our local fine-tuning multi-modal model consists of a two-stream convolutional neural network that can learn fusion information from both RGB and depth proposals before classification. At last, the multi-level scene image representation can be built from the top of probability distribution of the region proposals. Experiments are conducted on both NYU v1 depth and SUN RGB-D Datasets to thoroughly test our method's performance. The experimental results show that our method

with local multi-modal CNNs greatly outperforms state-of-the-art approaches.

This chapter has the potential to significantly improve RGB-D scene understanding. An extended evaluation shows that our local fine-tuning method outperforms direct global fine-tuning methods. The experiments also show that CNNs trained using a scene-centric dataset is able to achieve an improvement on scene benchmarks compared to a network trained using an object-centric dataset.

Chapter 5

RGB-D Data Fusion in Complex Space

5.1 Overview

Single RGB image understanding has been studied very well over the past decades. Nevertheless, many challenges still exist in computer vision research area because of limited information provided by RGB images. With the invention of high-quality and low-cost depth sensor, a new stream of research turns to seek new types of image representations for overcoming the traditional hard tasks [32]. For example, Gupta *et al.* [92] propose algorithms that generalize the $gPb-ucm$ approach [6] through taking advantage of depth information for hierarchical segmentation and object boundary detection. Bo *et al.* [21] propose Hierarchical Matching Pursuit (HMP) which builds feature hierarchies with an increasing receptive field layer by layer to obtain representations of RGB-D data.

Beyond this, many RGB-D fusion methods have been proposed to extract RGB-D features. Lai *et al.* [144] use $3D$ spin images and SIFT descriptors for depth features, and texton, color histogram and standard deviation of each color channel for RGB features. Socher *et al.* [229] present a model which is based on the combination of Convolutional Neural Networks (CNNs) and Recursive Neural Networks (RNNs), but this model extracts features of RGB-D images separately. Song *et al.* [234] propose an approach which makes a $3D$ volumetric scene from RGB-D images as inputs and $3D$ object bounding box as output through Region Proposal Network (RPN) to learn objectness and a joint $2D$+$3D$ object recognition network to extract geometric features in $3D$ and color features in $2D$.

The rest of this chapter is organized in the following way. Sect. 5.2 gives the motivation and contributions. Sect. 5.3 introduces our fusion method and shows the observations and conjectures of this method from three aspects: *mutual information and independence*, *feature distribution* and *Euclidean KS-distance to uniformity*. In Sect. 5.4, we describe our $\mathbb{C}$-SIFT and show some example samples produced by $\mathbb{C}$-SIFT. Experimental setup, results

and the relevant experimental result analysis are comprehensively presented in Sect. 5.5. Finally, the summary of this chapter is given in Sect. 5.6.

5.2 Motivation and Contributions

Motivation. Above mentioned methods have not explored the correlation between raw RGB image and raw depth image. Most of the methods just learn features from RGB and depth data and then simply concatenate them together as RGB-D features or encode these two kinds of features. The major disadvantage is that the correlation and complementary property between RGB and depth are ignored, and learning procedures cannot be adjusted mutually. Even Gupta *et al.* [93] propose HHA coding approach which replaces the original depth map with three channels (horizontal disparity, angle between point normal and inferred gravity, height above ground) as RGB inputs, it still ignores the relationship between color and depth, especially the mutual information in the RGB-D pixel pairs. To better ex-

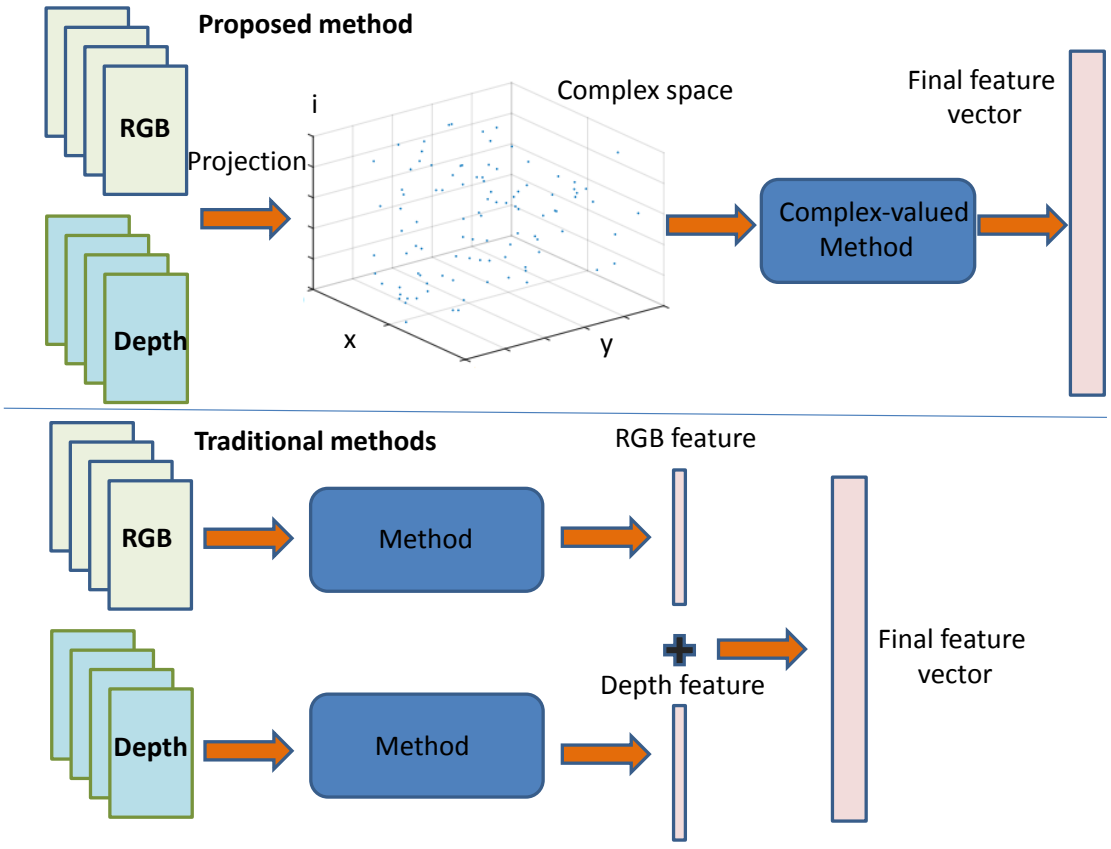


Fig. 5.1 The flow chart shows the difference between our fusion method and traditional fusion methods.

plore the correlation between the RGB pixel and the corresponding depth pixel, and take advantage of the complementary property, we first project raw RGB-D data into a complex space and then jointly learn features from the fused RGB-D images. The correlated and individual parts of the RGB-D information in the new feature space are well combined. The difference between our fusion method and traditional fusion methods is shown in Fig. 5.1. Our fusion method can also be considered as representing the data closer to the nature of the data. In physics, the range data correspond to the phase change and color information corresponds to the intensity. From computer vision view, the feature representations are expected to satisfy low mutual information and also show a lot of variations. The fused RGB-D data should be treated holistically. Therefore, it is reasonable to fuse RGB-D images to a complex-valued representation. Beyond this, we also modify the classical SIFT into complex-valued SIFT ($\mathbb{C}$-SIFT) to evaluate our fusion method. It is worthy to note that $\mathbb{C}$-SIFT is just an example to show the advantages of the fusion method. CNNs [136], Deep Belief Nets (DBNs) [100], GIST [191] or other methods can be introduced into complex space as well. For example, in one recent work, Chiheb Trabelsi et al. [251] use the atomic components for complex-valued deep neural networks and apply them to convolutional feed-forward networks. More specifically, Chiheb Trabelsi et al. rely on complex convolutions and present algorithms for complex batch-normalization, complex weight initialization strategies for complex-valued neural nets. The experimental results show that such complex-valued models achieves competing or better performance against their real-valued counterparts. Meanwhile, deep complex models are also tested on some computer vision tasks where achieve state-of-the-art performance. In this Chapter, our experimental results on two popular datasets also clearly show that our fusion method is advantageous.

Contributions. (1) We propose a new method which can better explore the correlation between the RGB pixel and the corresponding depth pixel for RGB-D data fusion. It makes the correlated and individual parts of the RGB-D information in the new feature space well combined. (2) We modify classical SIFT to complex-valued SIFT ($\mathbb{C}$-SIFT) to evaluate our fusion method resulting in higher performance compared to classical fusion methods.

5.3 Fusion Methodology

Different from other work which treat RGB and depth images separately, the fused images in our methodology are represented by complex values, which are closer to the nature of the data itself. This methodology makes the representations of RGB-D images greater distinctiveness, higher entropy on the whole images, higher entropy of the scale-space derivatives and larger feature quantity. Meanwhile, it is able to preserve the correlation information be-

tween the RGB image and the corresponding depth image. Since Kinect is able to provide RGB image and depth image synchronously, RGB image can be considered as amplitude measurements, which depends on the nature illumination such as sun light. According to depth images, no matter which sensing system is chosen for depth image representation, the pixel values of the depth images always mean the distance from the camera to the observed objects. The depth image is often considered as the phase change measurement, which depends on the measured scattering obtained from active illumination such as laser. The phase can be regarded as actual distance. The intensities of the active sensors can generally be considered as including two kinds of intensity: active intensity and passive intensity. The passive intensity can be calculated by the extraneous light, such as the sun light. The active intensity can be calculated through the active illumination from the sensor, such as the laser. For depth measurements with the uniqueness range, inverse-square law reveals the approximation: $I_D \approx I_R/\phi^2$, where I_R is the passive intensity of the RGB image, I_D is the active intensity of corresponding depth image, and ϕ is the phase which can be calculated from the depth values d. Therefore, it proves that it is reasonable to correlate I_R and I_D together through (I_R, I_D, ϕ). In physics, the phase difference can always be considered as a phase value from the mathematical concept, hence the representation of the RGB-D images with complex values becomes natural. It is worthy to note that all the RGB images mentioned in our methodology are first converted into gray images. We can define $I_R(x,y)$ as the RGB image, $I_D(x,y,d)$ as the depth image, where $d = d(x,y)$, x and y are the image coordinate points, d is the depth value on the coordinate (x,y). Combining the physical and mathematical concepts, the fused image function can be presented:

$$f(x,y,d) = I_R(x,y) + I_D(x,y,d)e^{i\phi(x,y,d)}, \tag{5.1}$$

where $\phi = \phi(x,y,d)$ can be considered through distance:

$$d = n \cdot 2\pi\ell + \phi\ell, \tag{5.2}$$

where $n \in \mathbb{N}$, $2\pi\ell$ is considered as the uniqueness range from camera ($\ell \in \mathbb{N}$). n is considered as the "wrapping number". ℓ is denoted as the multiple of some unit of length. In our definition, the real part is corresponding to the RGB image and the complex part is corresponding to the depth image. Each value on the depth image has a different ϕ. The ϕ has the range $[0, 2\pi)$. Moreover, the fused complex-valued image can be represented as Polar representation and Cartesian representation. Note that the following Polar representation

and Cartesian representation are the transformations of the definition:

$$I_f^P = |f|e^{iarg(f)}, \tag{5.3}$$

$$I_f^C = Re(f) + iIm(f), \tag{5.4}$$

where $|f| = \sqrt{I_R^2 + 2I_R I_D \cos\phi + I_D^2}$, $arg(f) = \arctan\frac{I_D \sin\phi}{I_R + I_D \cos\phi}$, $Re(f) = I_R + I_D \cos\phi$ and $Im(f) = I_D \sin\phi$. Since we normalize all complex-valued images, we can obtain $max|f| = 1$. Fig. 5.2 shows some random example images from 8 different scenes which include computer room, bedroom, classroom, dining room, kitchen, furniture room, lecture room and restroom from top to bottom. Since Kinect cannot perceive the light-reflecting area, we discard the scenes which include many mirrors and tiles, such as bathroom. Each row of Fig. 5.2 includes RGB image, gray image, depth image, fused image and 3D graphic simulation of one scene. From the images produced by our fusion function in the fourth column, we can visually find that the fused images do contain the depth data and hold the color information simultaneously. For the purpose of better understanding the fused images, $3D$ graphic simulations of the fused images are made in the fifth column. The size of the $2D$ plane which is at the bottom in the $3D$ graphic simulation is as the same as the size of the gray image. The third coordinate axis represents the depth of the corresponding pixels of the gray image.

The fused image can better explore the correlation between the RGB pixel and the corresponding depth pixel for RGB-D data fusion, which makes the correlated and individual parts of the RGB-D information in the new feature space well combined. RGB image can be considered as amplitude measurements, which depends on the nature illumination. According to the depth images, the pixel values of the depth images always mean the distance from the camera to the observed objects, which depends on the measured scattering obtained from active illumination. The depth image can often be considered as the phase change measurement. Therefore, the fused images can be generated by the strategy that each pixel on the RGB image has a phase change which can be calculated by the corresponding depth pixel: $e^{i\phi(x,y,d)}$.

Till now, we have obtained all our image representations in this chapter: $I_R(x,y)$ for RGB images, $I_D(x,y,d)$ for depth images, I_f^P for the fused RGB-D Polar representations and I_f^C for the fused RGB-D Cartesian representations. According to these two representations, the gray pixels on the gray image and the corresponding depth pixels on the depth image are corresponded point to point. In the following subsection, we will give the comparison among these representations from three aspects: *mutual information and independence*,

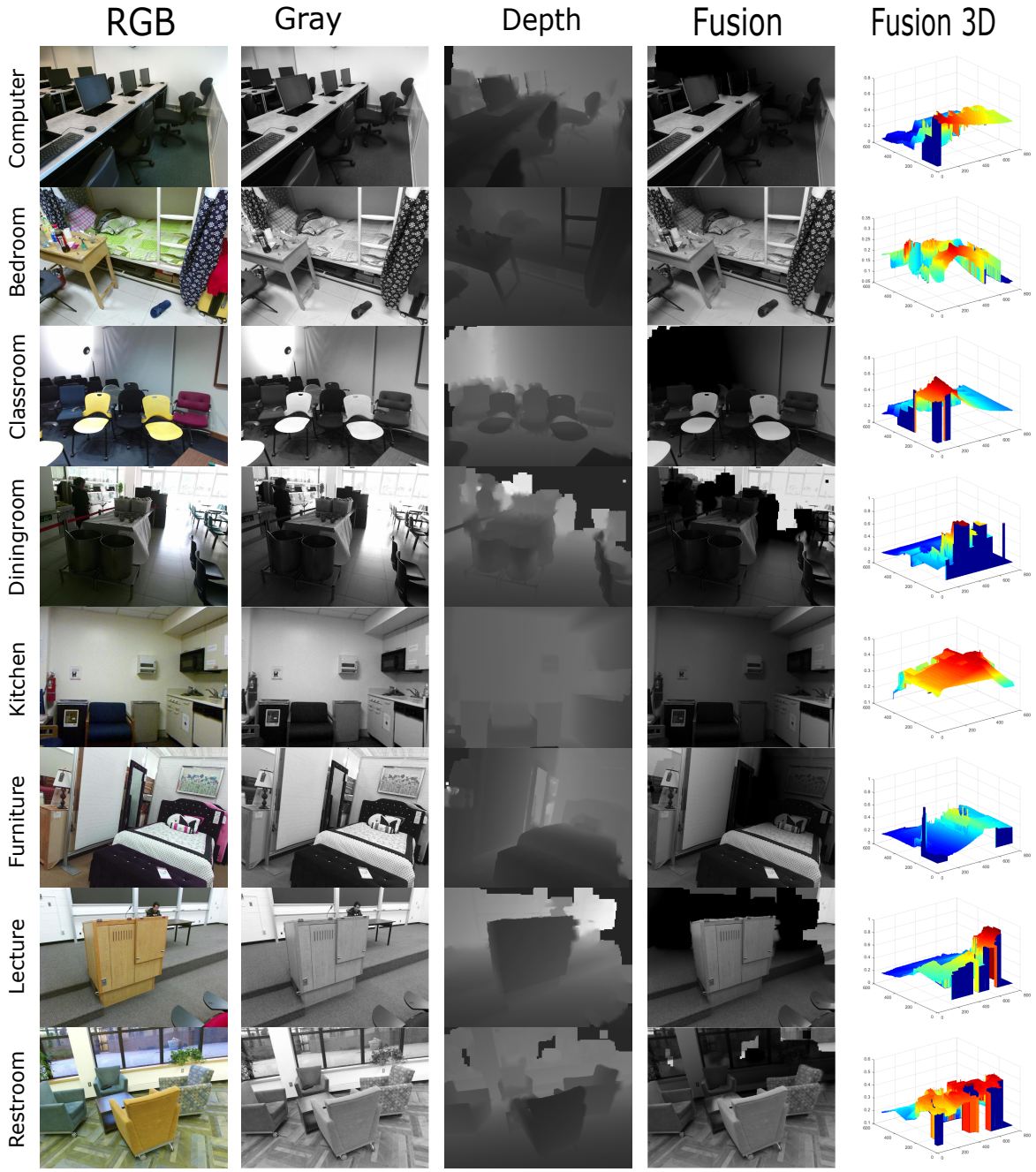


Fig. 5.2 Some random example images from 8 different scenes which include computer room, bedroom, classroom, dining room, kitchen, furniture room, lecture room and restroom from top to bottom. Each row in this figure includes RGB image, gray image, depth image, fused image and fused image 3*D* graphic simulations of each scene.

feature distribution and *Euclidean KS-distance to uniformity*.

5.3.1 Mutual Information and Independence

If the mutual information among the images is low, it means that these images have more independence. According to [252], good features should be distinctive. According to the information theory [170], the information content depends on the representation of the images. Therefore, we first expect that the fused RGB-D images have the property of distinctiveness while containing more entropy. Through above four image representations, we can easily obtain that $E(I_f^P)$ or $E(I_f^C) > E(I_R)$ or $E(I_D)$. Moreover, information theory describes that the image representation transformed from Polar coordinate to Cartesian coordinate increases entropy. From computer vision view point, the procedure of our fusion method equals to add structure information into the original gray images, which increases the entropy. Since $E(I_f^C) > E(I_f^P)$, I_f^C yields more structure than I_f^P. The choice of the coordinates can decide the entropy of one system. In addition, the Jacobian of the transformation can result in the change of entropy [102]. According to the complex-valued images, it can result in that Cartesian coordinate is better than Polar coordinate. Therefore, based on above several conclusions, we can observe that $E(I_f^C) > E(I_f^P) > E(I_R)$ or $E(I_D)$. In the following discussions, we only compare higher entropy image presentations: $E(I_f^C)$ and $E(I_f^P)$.

We define $E_{A,\omega}$ as the joint entropy of A and B, and $E_{R,I}$ as the joint entropy of $R = Re(f)$ and $I = Im(f)$. We can obtain:

$$E_{A,\omega} = E_{R,I} + \langle logA \rangle, \tag{5.5}$$

where from differential entropy representation,

$$\langle logA \rangle = \int \rho(R,I) logA(R,I) dRdI, \tag{5.6}$$

where $\rho(R,I)$ is the joint distribution function of R and I. Through the Jacobian transformation between the distributions, we can obtain $\rho(R,I) = \rho(A,\omega) \cdot |J|$. Under this circumstance, since $max|f| = 1$, $J = A(R,I) = \sqrt{R^2 + I^2} \leq 1$. Meanwhile, it proves $\langle logA \rangle < 0$.

The mutual information of (R,I) and (A,ω) are defined as:

$$MI(R,I) = E_R + E_I - E_{R,I}, \tag{5.7}$$

$$MI(A,\omega) = E_A + E_\omega - E_{A,\omega}. \tag{5.8}$$

Therefore, with Eq. (5.5), the difference between $MI(R,I)$ and $MI(A,\omega)$ is as following:

$$\begin{aligned} \nu &:= MI(R,I) - MI(A,\omega) \\ &= (E_R + E_I) - (E_A + E_\omega) + \langle logA \rangle. \end{aligned} \tag{5.9}$$

Here, ν is the measure for mutual information and independence between $MI(R,I)$ and $MI(A,\omega)$. Since the smaller $MI(x,y)$ is, the more independent x and y is, if $\nu < 0$, it means $MI(R,I)$ is more independent than $MI(A,\omega)$. From the information theory, the value of $(E_R + E_I) - (E_A + E_\omega)$ is really small, which hardly affects the value ν. Meanwhile, we have observed $\langle logA \rangle < 0$. Actually, in our experiments, both ν and $\langle logA \rangle$ are computed around -1. Therefore, we can confirm that I_f^C is more independent and contains less mutual information than I_f^P.

From above observation and conjecture, according to mutual information and independence, we consider I_f^C as the optimal fused image representation.

5.3.2 Feature Distribution

Better representation can be extracted larger number of features under the same method and is also more uniformly distributed on the image plane. With the increase of the entropy from different image representations, the number of the extracted features with the same method increases as well. Since we have observed that $E(I_f^C) > E(I_f^P) > E(I_R)$ or $E(I_D)$, I_f^C is supposed to have the most features among above four representations.

Take image scale-space feature detection for example, the entropy of the scale-space derivatives of the image representation from Polar to Cartesian is increaseed. With the increase of image derivative entropy, it leads to more persistent texture. The scale-space equation $\frac{\partial f}{\partial t} = \triangle f$ aims to find the persistent texture. Since I_f^C contains more entropy than I_f^P over the scale-space derivatives, we define:

$$E_{\dot{A},\dot{\omega}} = E_{\dot{R},\dot{I}} + \langle A \cdot |cos\dot{\omega} sin\omega - sin\dot{\omega} cos\omega| \rangle, \tag{5.10}$$

where $\dot{A}$, $\dot{\omega}$, $\dot{R}$ and $\dot{I}$ are the derivation of f on A, ω, R and I. In this case, $\langle A \cdot |cos\dot{\omega} sin\omega - sin\dot{\omega} cos\omega| \rangle < 0$.

Since $E(I_f^C) > E(I_f^P) > E(I_R)$ or $E(I_D)$, it follows from the Jacobian $J = A \cdot (cos\dot{\omega} sin\omega - sin\dot{\omega} cos\omega)$ of the transformation of derivatives. The RGB features and the depth features in real-valued image pairs are considered to be scale-space features for I_R and I_D. Similarly, for the scale-space feature, the Cartesian features and the Polar features of one complex-valued

image can be denoted for $Re(f)\&Im(f)$, $|f|\&arg(f)$ respectively. Therefore, we can obtain the comparison of the number of features about these four representations: $N(I_f^C) > N(I_f^P) > N(I_R)$ or $N(I_D)$.

From above observation and conjecture, according to feature distribution, we consider I_f^C as the optimal fused image representation.

5.3.3 Euclidean *KS*-distance to Uniformity

In this subsection, we will focus on the distribution of the features on the image grid. Robust features are always sampled from a uniform distribution. If the features are closer to the uniform, it means that the features contain more entropy. Therefore, we need to calculate the distance between the samples of Cartesian features and the uniform distributions on the image plane. We consider the extracted n features from a scene are independently from another, and these features are identically distributed. According to n independent and $\{X_1, X_2, \cdots, X_{i1}\} \in \mathbb{R}$ as identically distributed random variables with common cumulative distribution function, the empirical distribution function $F_n(X)$ can be denoted as:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mu_{(-\infty,x]}(X_i), \tag{5.11}$$

where $\mu_{(-\infty,x]}$ is the indicator function. The extracted n features mentioned above can be considered as be taken from $F(x)$. From Glivenko–Cantelli theorem, F_n uniformly converges to F:

$$\|F_n - F\|_\infty = \sup_{x\in\mathbb{R}} |F_n(x) - F(x)| \longrightarrow 0, \tag{5.12}$$

For arbitrary F, $\|F_n - F\|_\infty$ is Kolmogorov-Smirnov statistic [54], which has characters of distance among the cumulative distribution functions. We can also call it *KS*-distance. We can then define different image representation *KS*-distances through Euclidean norm:

$$d(S,\lambda) := \sqrt{\|F_n(y) - \lambda_y\|^2 + \|F_n(x) - \lambda_x\|^2}, \tag{5.13}$$

here λ and λ_i are cumulative distribution functions of uniform distributions from coordinate axis and plane separately. $d(S,\lambda)$ can be defined as Euclidean *KS*-distance to uniformity. S can be considered as n points on the plane. We define S_C as the Cartesian features sample, S_P as the Polar features sample, S_R as the RGB features sample and S_D as the depth features sample. Since according to independence, $S_C > S_P > S_R$ or S_D, and the number of features is:

$N(S_C) > N(S_P) > N(S_R)$ or $N(S_D)$, we can obtain:

$$d(S_C, \lambda) < d(S_P, \lambda) < d(S_R, \lambda) \; or \; d(S_D, \lambda), \tag{5.14}$$

where λ can be defined as the uniform distribution on the image plane.

From above observation and conjecture, according to Euclidean *KS*-distance to uniformity, we consider I_f^C as the optimal fused image representation.

Above three subsections show that I_f^C is the optimal image representation among four image representations: I_f^C, I_f^P, I_R and I_D. Therefore, in our RGB-D fusion methodology, I_f^C is chosen as the optimal image representation.

5.4 Complex-valued SIFT

As we have shown the observations and conjectures that the features extracted from our fusion method on Cartesian coordinate have more advantages, in this section, we modify classical SIFT into complex-valued SIFT ($\mathbb{C}$-SIFT) to evaluate our observations and conjectures. It is worthy to note that SIFT is just chosen for example in our article. Some famous algorithms and models such as Speeded Up Robust Features (SURF) [12], Maximally Stable Extremal Regions (MSER) [177], DBNs and CNNs can also be generalized for our fusion method.

The classical SIFT is designed for real-valued images, which makes it no sense on complex-valued images. A simple way is to calculate the real part $R = Re(f)$ and imaginary part $I = Im(f)$ separately. However, this goes against our target which hopes fused images computed together. Therefore, we modified SIFT from essences and propose $\mathbb{C}$-SIFT. Following the work in [167], we also use the only possible scale-space kernel Gaussian function. The scale space of an image can be defined as:

$$L(\sigma) = G(\sigma) * I, \tag{5.15}$$

where I is the input fused image, $*$ is the convolution operation. $G(\sigma)$ is a real-valued variable-scale Gaussian.

Aim to efficiently detect the stable feature point locations, the convolution of difference of Gaussian (DoG) function with the image can be computed:

$$D(k^i\sigma) = L(k^{i+1}\sigma) - L(k^i\sigma), \tag{5.16}$$

where k is the constant multiplicative factor.

Fig. 5.3 Examples of two pairs of RGB-D images by ℂ-SIFT. It is shown that the keypoints detected in the fused images are much more than the sum of keypoints in raw RGB images and raw depth images. The parameters of keypoint detection are always the same.

Then we follow the important step of SIFT to regularly sample the scale space. Following this, an image pyramid is built with different scales of the original images, which are grouped by octaves. Since each representation of the pixel on the fused images contains color and depth information simultaneously, it needs to explore a new calculation method to compute the local maxima. In the local extrema detection step, different from SIFT which detects the local extrema and minima of $D(\sigma)$ through comparing its 26 real-valued neighbors, our algorithm chooses to compare the module m among these neighbors. The module can be calculated as $m^2 = Re(f)^2 + Im(f)^2$. It can make sure that the color information and the depth information are all considered when choosing the keypoints. Examples of two pairs of RGB-D images by $\mathbb{C}$-SIFT are shown in Fig. 5.3. From Fig. 5.3, we can see that the keypoints detected in the fused images are much more than the sum of keypoints in raw RGB images and keypoints in raw depth images under the same parameters of keypoint detection. It shows the advantages of our fusion method. Most of the new keypoints cluster around the added depth profile parts on the fused images.

Once these key points have been found, we follow the steps in [167] to reject the key points which are with low contrast or poorly localized along an edge. At last, HOG 3D [130] is chosen in our algorithm to describe the key points. According to the original SIFT, there are only eight directions on the plane for each orientation histogram. However, since each pixel on the fused image has a third dimension which is the depth, it results in more directions in the space for each orientation histogram. Therefore, SIFT is not chosen in our algorithm to describe the key points. Instead of time being the third dimension, we choose depth as the third dimension. For details on HOG 3D, see [130].

5.5 Experimental Setup

Since the fused images produced by our fusion method have more advantages than RGB-D images, and the extracted features through our $\mathbb{C}$-SIFT are also robust, in this section we conduct experiments on computer vision tasks. We evaluate our work on the NYU Depth V1 dataset [223] and SUN RGB-D dataset [231]. Both of these two datasets are derived from the publicly available RGB-D sensor-based scene database. Compared to RGB-D object datasets, RGB-D indoor scene understanding tasks can show more advantages of depth information. In our experiments, our work only compares with SIFT algorithm for feature representations on scene classification application. It needs to note that it does not mean that our work cannot outperform the state-of-the-art. Some famous algorithms (GIST) and deep learning models (CNNs and DBNs) do have better performance on scene classification. But it is unfair to directly compare $\mathbb{C}$-SIFT with these methods. In this article, $\mathbb{C}$-SIFT

is an example to show the advantages of the fusion method. It inherits the principles of SIFT and fully takes advantages of depth information through extended to a complex space. Therefore, in fact, $\mathbb{C}$-SIFT is still based on SIFT method. If CNNs, DBNs, GIST or some other state-of-the-art methods can be extended into complex space, it becomes reasonable to compare with these methods. More precisely, we hope to show the advantages of our fused method under the same conditions without the influence from the methodological difference. The model for classification is linear SVM. Details of the datasets and experimental results are provided below.

5.5.1 NYU Depth v1

Aim to evaluate our work, we firstly conduct experiments on the NYU Depth V1 dataset. Since the standard classification protocol removes scene 'cafe' from the dataset, we use the remaining 6 different scenes. Since there are so many objects in one scene and the correlation between images in one scene is low, it makes NYU Depth v1 a very challenging dataset.

The baseline in [223] is calculated through SIFT extracted from RGB image and SIFT on the depth image. The RGB-D SIFT features are created by concatenating RGB features and depth features together. Our RGB-D $\mathbb{C}$SIFT features are from the images produced by our fusion method on the whole dataset. The comparison of the performance of our fusion method plus $\mathbb{C}$SIFT and the baseline with different K-means dictionary size is shown in Table. 5.1. From Table. 5.1, we can see that the classification accuracy of our fusion method plus $\mathbb{C}$SIFT outperforms RGBD-SIFT by around 4%, which proves that our method is significantly more effective than RGB-D SIFT on category classification tasks. The confusion matrix of our final best results is shown in Fig. 5.4. The classification accuracy for each category is represented on the diagonal elements.

Table 5.1 Scene classification performance on NYU Depth v1 dataset. A significant performance gain is obtained with a large dictionary.

Methods	**Accuracy on Different Dictionary Size** (%)				
	50	100	200	400	800
RGB-SIFT	51.2	54.7	54.8	55.0	54.9
Depth-SIFT	48.1	50.3	47.9	47.0	46.8
RGBD-SIFT	52.5	56.5	55.2	56.5	59.6
RGBD-$\mathbb{C}$SIFT	**54.7**	**58.1**	**58.3**	**60.6**	**63.4**

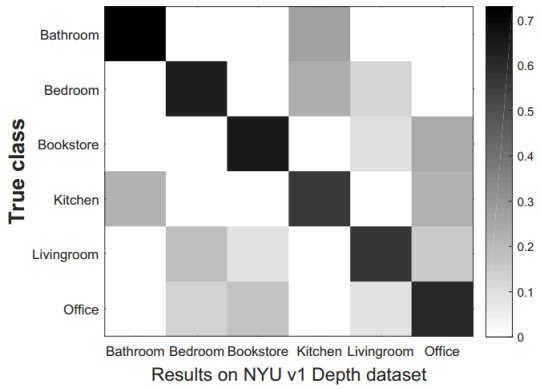


Fig. 5.4 Confusion matrix about our fusion method result on NYU Depth v1 dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

5.5.2 SUN RGB-D

SUN RGB-D dataset is captured by four different sensors and contains 10,335 RGB-D images. Following the setup in [231], we split the dataset to ensure about half for training and half for testing in each sensor. Same with the use of SUN RGB-D dataset in Chapter 4, these images, which are captured from the house with similar furniture styles or the same building, only all go into one of the training and testing sets.

Table 5.2 Scene classification performance on SUN RGB-D dataset. A significant performance gain is obtained with a large dictionary.

Methods	**Accuracy on Different Dictionary Size** (%)				
	50	100	200	400	800
RGB-SIFT	16.3	16.7	16.9	18.5	19.2
Depth-SIFT	14.6	14.2	16.8	17.7	17.5
RGBD-SIFT	18.1	19.4	18.2	20.8	21.3
RGBD-$\mathbb{C}$SIFT	**20.7**	**21.3**	**23.3**	**22.6**	**24.4**

We evaluate our fusion method following the experimental setup in [231] which also represents RGB-D features through concatenating RGB features and depth features. Our RGB-D $\mathbb{C}$SIFT features are from the fused images on the whole dataset. Table. 5.2 shows the comparison of the performance of our fusion method plus $\mathbb{C}$SIFT and the traditional SIFT baseline with different K-means dictionary size. From Table. 5.2, we can find that the classification accuracy of our fusion method plus $\mathbb{C}$SIFT outperforms RGBD-SIFT by around 3.1%, which proves that our method is significantly more effective than RGBD-SIFT

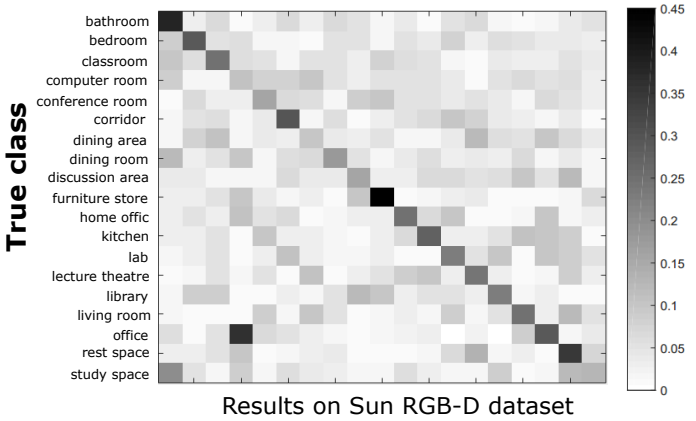


Fig. 5.5 Confusion matrix about our fusion method results on SUN RGB-D v1 dataset. The labels on the horizontal axis denote the predicted classes. The labels on the vertical axis express the true classes.

on classification tasks. The confusion matrix about our fusion method across 19 classes is shown in Fig. 5.5. The classification accuracy for each category is represented on the diagonal elements.

5.6 Summary

In this chapter, we have proposed a new RGB-D fusion method and $\mathbb{C}$-SIFT algorithm for fusing RGB-D images, which can better reveal the correlation between the RGB pixels and the corresponding depth pixels, taking advantage of the complementary property. Instead of concatenating extracted features separately before the classification, we firstly project raw RGB-D data into a complex space and then jointly extract features from the projected RGB-D images. The experimental results show that our method achieves competing performance against the classical SIFT.

Chapter 6

Recognizing RGB Information from RGB-D data

6.1 Overview

The problem of recognizing RGB images captured by conventional surveillance cameras through leveraging a set of labeled RGB-D data has been presented in [116] [37] [111]. This new task is considered as an unsupervised domain adaptation (UDA) problem, which aims to take advantage of the additional depth information in the source domain and reduce the data distribution mismatch between the source and target domains simultaneously. The training data in UDA consists of labeled RGB-D source data and unlabeled RGB target examples [174]. It is different from traditional classification problems which often assume that the labeled training data comes from the same distribution as that of the test data. In realistic scenarios, the source and target domains follow different distributions, especially when images are acquired from different cameras, or in various conditions. The classifier which is trained on the previous dataset would fail to classify the following dataset correctly without adaptation.

The rest of this chapter is organized in the following way. Sect. 6.2 gives a brief review of NMF. Sect. 6.3 gives the motivation and contributions of this chapter. Sect. 6.4 introduces our adaptive Visual-Depth Embedding method in detail. Experimental setup, results, parameter analysis and the relevant experimental result analysis are comprehensively presented in Sect. 6.5. Finally, the summary of this chapter is given in Sect. 6.6.

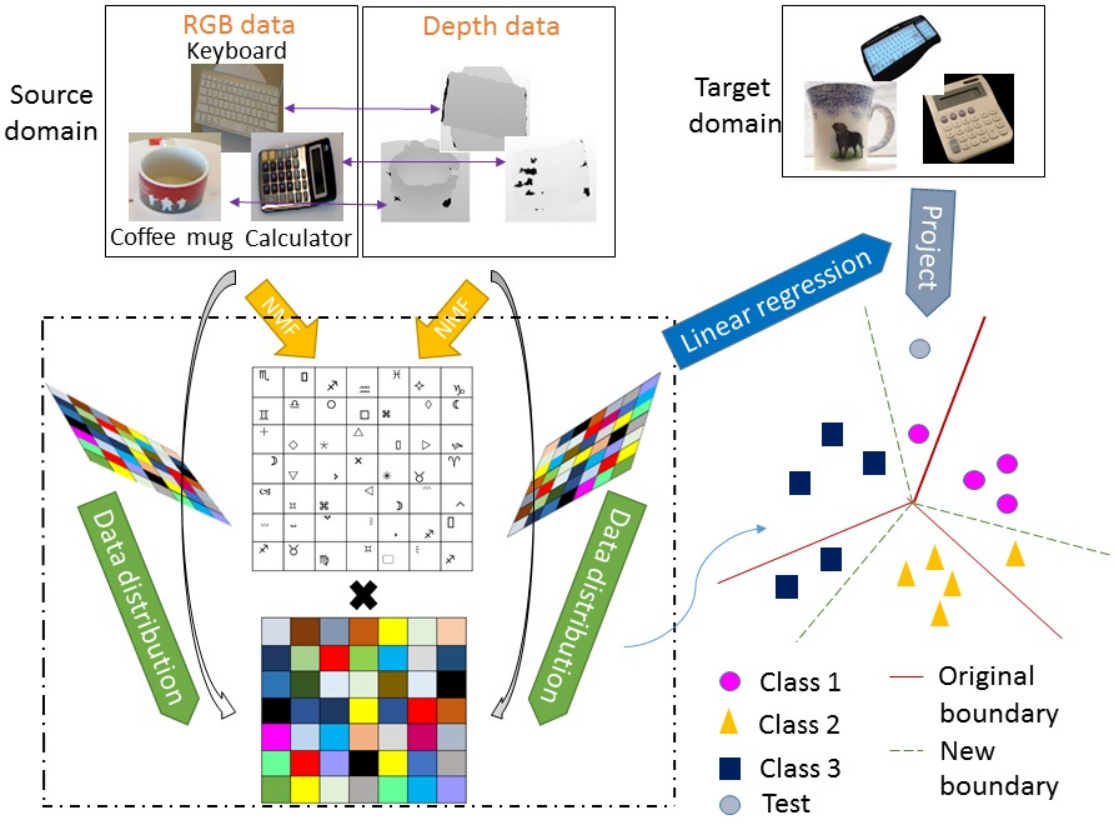


Fig. 6.1 The outline of the proposed method. We have RGB and depth features in the source domain, and RGB features in the target domain. Our main idea aims to find a shared latent space so that the shared parts between RGB and depth images can be preserved. Our aVDE can automatically adapt to the target latent space so as to further correct the classification errors. Examples from three classes are used to show the difference between the original decision boundaries and the new decision boundaries which are obtained by matching and reweighting.

6.2 Brief Review of NMF

As a matrix factorization algorithm which can learn the non-negative parts representation of objects, Non-negative Matrix Factorization (NMF) [150] plays a significant role in the fields of information retrieval and data mining. Suppose a non-negative matrix has M N-dimensional data vectors $X = [x_1, \cdots, x_n] \in \mathbb{R}_{\geq 0}^{M \times N}$. NMF tries to factorize this non-negative matrix into two non-negative matrices $U = [u_{id}] \in \mathbb{R}^{M \times D}$ and $V = [v_{jd}] \in \mathbb{R}^{D \times N}$ whose result can have a good approximation of the original matrix X. Lee and Seung [150] also proposed two objective functions to measure the distance between these two non-negative matrices X and UV. The objective function which is based on the difference can be represented as

$$O_F = \|X - UV\|^2 = \sum_{i,j} (x_{ij} - \sum_{d=1}^{D} u_{id} v_{jd})^2, \tag{6.1}$$

where $\|\cdot\|$ is the Frobenius norm.

For optimizing this objective function, the following iterative updating steps [150] are proposed to obtain local minima of O_F:

$$\begin{aligned} u_{jd}^{(t+1)} &= u_{jd}^{t} \frac{(XV^T)_{jd}}{(UVV^T)_{jd}} \\ v_{di}^{(t+1)} &= v_{di}^{t} \frac{(U^T X)_{di}}{(U^T UV)_{di}} \end{aligned} \tag{6.2}$$

It has been proved that the local minima of O_F can be effectively found by the iterative updating algorithm. U can be considered as the basis. Meanwhile, the matrix V, which can be obtained from NMF, can be regarded as the low-dimensional representation of X. Currently, there are many algorithms based on NMF. In [157], local non-negative matrix factorization (LNMF), which can obtain better locality of features in basis components, was proposed to learn parts-based and spatially localized representations of visual patterns. To improve LNMF, Cai et al. [29] proposed the Locality Preserving Non-negative Matrix Factorization (LPNMF) which analyzes the similarity between two data points on the hidden topics. Beyond these methods, in [282], Accelerated LPNMF (A-LPNMF) was proposed to solve the computational complexity problem of LPNMF and compress the data using an efficient landmark-based approach [39][166]. In order to detect the underlying manifold structure, Cai el al. presented Graph Regularized Non-negative Matrix Factorization (GNMF) in [28], which combines matrix factorization with the graph structure. Constrained Non-negative Matrix Factorization (CNMF) [276] considers the label information as additional hard constraints and the data points in the same class are merged in the new representation space.

Recently, motivated by the progress in the research of sparse coding, in order to ensure the sparseness of the obtained representation, Non-negative Local Coordinate Factorization (NLCF) adds the local coordinate constraint [41].

6.3 Motivation and Contributions

Motivation. There are two challenges in our task: 1) How to address the domain shifting problem between the source and target domains? 2) How to effectively explore the additional depth information to boost the performance further? A very fruitful line of work has been focusing on solving domain adaptation problem, where labeled target data is not needed, yielding excellent results [52, 77, 110]. However, none of them can incorporate depth information. On the other hand, many methods using the additional depth information have been proposed for classification tasks as well [20] [92]. However, these methods take the unrealistic assumption that the training and testing data are from the same domain.

In this chapter, we aim to solve above two challenges simultaneously by a novel RGB-D UDA method, referred to as **a**daptive **V**isual-**D**epth **E**mbedding (aVDE). Our method captures the shared latent bases and individual subspaces between two representations of labeled RGB and depth modalities in the source domain first. We utilize the advantages of Nonnegative Matrix Factorization (NMF) [108] [30] [218] for the discovery of the shared components between RGB and depth images. In addition, to preserve the significant structure of the original RGB-D data, and overcome the problem of NMF which cannot discover the intrinsic discriminating and geometrical structure in the data space, we solve this problem from the probability distribution perspective, *i.e.* to minimize the Jensen-Shannon divergence (JSD) between the probability distributions in RGB and depth spaces. Then we transfer the knowledge of depth information to the target dataset through an orthogonal projection to align the data in the shared latent feature space with the target domain. In the adaptive embedding step, we minimize the nonparametric Maximum Mean Discrepancy (MMD) in an infinite dimensional reproducing kernel Hilbert space (RKHS) [89] for feature matching, and minimize the $\ell_{2,1}$-norm structured sparsity penalty [175] on the shared latent space instances for instance reweighting. We match features and reweight instances jointly across the shared latent space and the projected target domain in a principled dimensionality reduction procedure for an adaptive classifier. Feature matching can discover a shared feature representation through the combination of the distribution difference reduction and the important properties of input data preservation (see Fig. 6.2 (c) (d)). However, when the domain difference is substantially large, some shared latent space instances are still not relevant to the projected target instances even in the feature-matching subspace. Therefore,

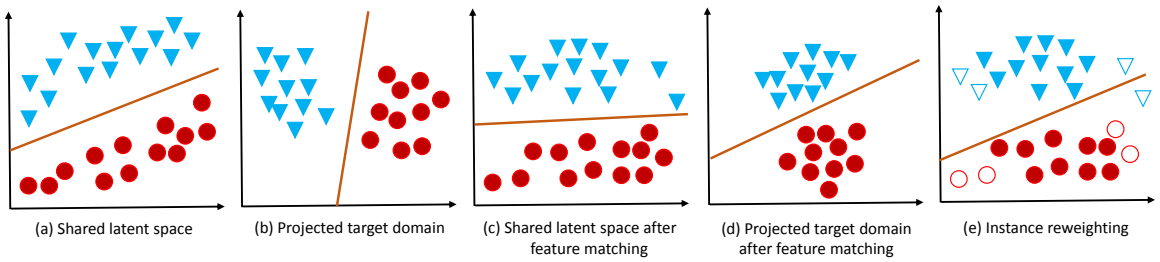


Fig. 6.2 Samples from the shared latent space and the projected target domain. (a) Shared latent space; (b) Projected target domain; (c) Shared latent space after feature matching; (d) Projected target domain after feature matching. The domain distance is still large after feature matching. (e) Further instance reweighting on shared latent space. The irrelevant shared latent space instances (shown as unfilled markers) are now down-weighted to further reduce the domain difference.

we introduce instance reweighting which can minimize the distribution difference through reweighting the shared latent data (see Fig. 6.2 (e)). The pipeline of our idea is described in Fig. 6.1. Comprehensive experiments for object recognition and scene classification on two pairs of real-world datasets show that aVDE can outperform state-of-the-art methods significantly.

Contributions. To summarize, our main contributions in this chapter are: i) we propose a novel UDA method which can effectively leverage depth information to recognize RGB images. The target domain does not contain the additional depth information. ii) A compact shared space uncovering the latent semantics can be learned by aVDE. Meanwhile, aVDE can preserve the data joint probability distribution in the source domain, then transfer the knowledge of depth information to the target dataset. iii) Through matching features and reweighting instances jointly across domains, a bridge between the source and target domains can be built.

6.4 Methodology

6.4.1 Notations

In this chapter, a vector is denoted by a lowercase letter in bold. The transpose of a matrix or a vector is denoted by the superscript T. We define I as an identity matrix. Besides, Table 6.1 shows the list of frequently used notations.

Problem (Adaptive Visual-Depth Embedding). Given two labeled modalities A and B in the source domain $\mathscr{D}_s$ with label set $Y = [y_1, \cdots, y_{N_s}]$ and an unlabeled target domain $\mathscr{D}_t$. To find the shared component space V and the projected target domain $\mathscr{P}_{\mathscr{D}_t}$ under the different

Table 6.1 Notations and descriptions.

Notation	Description	Notation	Description
$\mathscr{D}_s$, $\mathscr{D}_t$	Source/target domain	$\mathscr{P}_{\mathscr{D}_t}$	Projected target domain
A, B	RGB/depth modality	X	Input data matrix
V	Shared data space	K	Kernel matrix
P_A, P_B	Probability distributions	M	Adaptation matrix
$\mathscr{P}$	Orthogonal projection	Δ	MMD matrix
Λ	Connection matrix	$\widehat{\mathscr{G}}$	Diagonal sub-gradient matrix
D	Number of bases	k	Subspace bases
η, μ	Regularization parameter	Z	Subspace embedding

marginal probability distribution and conditional probability distribution, then learn a new feature space to reduce the domain distance by feature matching and instance reweighting across V and $\mathscr{P}_{\mathscr{D}_t}$.

6.4.2 Shared Component Problem Formulation

We use A and B to define the two modalities in the source domain $\mathscr{D}_s$ with dimensions and sample sizes $M_1 \times N_s$ and $M_2 \times N_s$ respectively: $A = [\mathbf{a}_1, \cdots, \mathbf{a}_{N_s}] \in \mathbb{R}_{\geq 0}^{M_1 \times N_s}$ and $B = [\mathbf{b}_1, \cdots, \mathbf{b}_{N_s}] \in \mathbb{R}_{\geq 0}^{M_2 \times N_s}$. NMF is used to find two nonnegative matrices from A: $V_1 \in \mathbb{R}_{\geq 0}^{D_1 \times N_s}$ and $U \in \mathbb{R}_{\geq 0}^{M_1 \times D_1}$ and two nonnegative matrices from B: $V_2 \in \mathbb{R}_{\geq 0}^{D_2 \times N_s}$ and $W \in \mathbb{R}_{\geq 0}^{M_2 \times D_2}$ with full rank whose product can represent the original matrix A and B approximately, *i.e.*, $A \approx UV_1$ and $B \approx WV_2$. In practice, we set $D_1 < \min(M_1, N_s)$ and $D_2 < \min(M_2, N_s)$. NMF aims to achieve the minimization of the following objective functions

$$\mathscr{L}_{NMF}^{A} = \|A - UV_1\|^2, s.t.\ U, V_1 \geq 0, \tag{6.3}$$

$$\mathscr{L}_{NMF}^{B} = \|B - WV_2\|^2, s.t.\ W, V_2 \geq 0, \tag{6.4}$$

where $\|\cdot\|$ is the Frobenius norm. The matrix V_1 and V_2 obtained from NMF are considered as low-dimensional representations. The matrix U and W denote the basis matrixes.

To learn fully shared spaces between RGB and depth modalities, the basic idea is to find suitable M_1 basis vectors for U and M_2 basis vectors for W via a shared coefficient matrix V. To learn the required shared space, we jointly optimize a convex combination of two constrained least squares problems: $V_1 = V_2 = V \in \mathbb{R}_{\geq 0}^{D \times N_s}$. The resulted objective function is:

$$\min_{U,W,V} \|A - UV\|^2 + \lambda \|B - WV\|^2, s.t.\ U, W, V \geq 0, \tag{6.5}$$

where parameter λ is given to balance the importance of the two terms. In this chapter, since RGB information and depth data are assumed equally important, for simplicity, we

set $\lambda = 1$. The training model is used to identify the latent shared bases determined via both RGB and depth data. Such jointed NMF can preserve shared components that make the model leads to a high-level representation V of the training RGB-D images in the bases space.

6.4.3 Data Distribution Divergency Reduction

NMF can learn a parts-based representation. We expect that the shared data space V obtained by our NMF-based shared structure learning algorithm can have locality structure from original data spaces A and B. However, NMF algorithm cannot find the intrinsic discriminating and geometrical structure in the data space which is important for our recognition task. For the preservation of the significant structure in the original RGB-D data, it is expected that the latent space can also balance the difference of data distribution between the RGB and depth modalities. We consider this problem from probability distribution aspect. Let P_A and P_B be the probability distributions in space A and B. We aim to find the joint probability distribution of the shared space Q which is shared by P_A and P_B as much as possible. In this chapter, we simply assume RGB and depth are equally important, *i.e.*, we hope the probability distribution Q in latent space V could be $Q = \frac{1}{2}(P_A + P_B)$. We can then minimize the Jensen-Shannon divergence (JSD) between P_A and P_B so that their structural difference can be mutually mitigated:

$$JSD(P_A||P_B) = \frac{1}{2}KL(P_A||Q) + \frac{1}{2}KL(P_B||Q), \tag{6.6}$$

where $KL(.||.)$ estimates the Kullback-Leibler divergence between the joint probability distributions.

P_A and P_B can be denoted as point-wise from p_A^{ij} and p_B^{ij}. Q can be represented as q_{ij}. The pairwise similarities in the original data space p_A^{ij} and p_B^{ij} are defined as:

$$p_A^{ij} = \frac{\exp\left(-\|\mathbf{a}^i - \mathbf{a}^j\|^2/2(\sigma_A^i)^2\right)}{\sum_{k\neq l}\exp\left(-\|\mathbf{a}_k - \mathbf{a}_l\|^2/2(\sigma_A^k)^2\right)}, \tag{6.7}$$

$$p_B^{ij} = \frac{\exp\left(-\|\mathbf{b}^i - \mathbf{b}^j\|^2/2(\sigma_B^i)^2\right)}{\sum_{k\neq l}\exp\left(-\|\mathbf{b}_k - \mathbf{b}_l\|^2/2(\sigma_B^k)^2\right)}, \tag{6.8}$$

where the conditional probability p_A^{ij} means the similarity between data points $\mathbf{a}^i$ and $\mathbf{a}^j$, and p_B^{ij} is the similarity between data points $\mathbf{b}^i$ and $\mathbf{b}^j$. $\mathbf{a}^j$ and $\mathbf{b}^j$ are chosen corresponding to their probability density under the Gaussian centered at $\mathbf{a}^i$ and $\mathbf{b}^i$ respectively. σ_A^k and σ_B^k are the variances of the Gaussian distribution that is centered on the data point a^i and b^i

respectively. Each data point a^i or b^i makes a significant contribution to the cost function. Through the probability distribution in the shared space, the joint probabilities q_{ij} is denoted as:

$$q_{ij} = \frac{(1+\|\mathbf{v}_i - \mathbf{v}_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|\mathbf{v}_k - \mathbf{v}_l\|^2)^{-1}}. \tag{6.9}$$

We set p_A^{ii}, p_B^{ii} and q_{ii} to zero for only significant points needed to model pairwise similarities. Meanwhile, it has the characteristics that $p_A^{ij} = p_A^{ji}$, $p_B^{ij} = p_B^{ji}$ and $q_{ij} = q_{ji}$ for $\forall i, j$. Since the definition in Eq. (6.9) is an infinite mixture of Gaussians which does not have an exponential, the evaluation of the density of a point becomes much faster than a single Gaussian. The significance of the data distribution can be measured effectively by the cost function based on JSD.

We use q_{ij} to jointly model p_A^{ij} and p_B^{ij}:

$$\begin{aligned} JSD = &\frac{1}{2}\sum_i\sum_j p_A^{ij}\log p_A^{ij} - p_A^{ij}\log q_{ij} \\ &+\frac{1}{2}\sum_i\sum_j p_B^{ij}\log p_B^{ij} - p_B^{ij}\log q_{ij}. \end{aligned} \tag{6.10}$$

Therefore, with this regularization, through combining the shared structure technique in Eq. (6.5) and the data structure preserving part in Eq. (6.10), the following objective function can be minimized:

$$\begin{aligned} &\min_{U,W,V} \|A-UV\|^2 + \|B-WV\|^2 \\ &+\eta JSD, s.t.\ U,W,V \geq 0, \end{aligned} \tag{6.11}$$

where $A \in \mathbb{R}^{M_1\times N_s}$, $V \in \{0,1\}^{D\times N_s}$, $B \in \mathbb{R}^{M_2\times N_s}$, $A,U,W,V,B \geqslant 0$, $U \in \mathbb{R}^{M_1\times D}$, $W \in \mathbb{R}^{M_2\times D}$, and η controls the smoothness of new representation.

For real world applications, the shared space data only from NMF-based shared structure algorithm is not effective and meaningful. Therefore, JSD is used to preserve the structure of original RGB-D data.

6.4.4 Relaxation and Optimization

We cannot directly calculate the discreteness condition V in Eq. (6.11) in the optimization procedure. Therefore, for the obtaination of real-values, the data $V \in \{0,1\}^{D\times N_s}$ is firstly

relaxed into the range $V \in \mathbb{R}^{D\times N_s}$. Therefore, the Lagrangian of our problem is:

$$\begin{aligned}\mathscr{L} = \|A-UV\|^2 + \|B-WV\|^2 + \eta JSD \\ + tr(\Phi U^T) + tr(\Theta W^T) + tr(\Psi V^T),\end{aligned} \tag{6.12}$$

where matrices Ψ, Θ and Φ are three Lagrangian multiplier matrices. In order to make the derivation clearer, ηJSD is simply denoted as G. Two auxiliary variables d_{ij} and Z are defined as follows:

$$d_{ij} = \|\mathbf{v}_i - \mathbf{v}_j\| \text{ and } Z = \sum_{k\neq l}(1+d_{kl}^2)^{-1}. \tag{6.13}$$

It needs to note that if $\mathbf{v}_i$ changes, the only pairwise distances that change are d_{ij} and d_{ji}. The gradient of G with respect to $\mathbf{v}_i$ is obtained:

$$\frac{\partial G}{\partial \mathbf{v}_i} = 2\sum_{j=1}^{N}\frac{\partial G}{\partial d_{ij}}(\mathbf{v}_i - \mathbf{v}_j). \tag{6.14}$$

Then $\frac{\partial G}{\partial d_{ij}}$ is calculated through Jensen-Shannon divergence in Eq. (6.10):

$$\frac{\partial G}{\partial d_{ij}} = -\frac{\eta}{2}\sum_{k\neq l}(p_A^{kl}+p_B^{kl})\left(\frac{1}{q_{kl}Z}\frac{\partial((1+d_{kl}^2)^{-1})}{\partial d_{ij}} - \frac{1}{Z}\frac{\partial Z}{\partial d_{ij}}\right). \tag{6.15}$$

Since $\frac{\partial((1+d_{kl}^2)^{-1})}{\partial d_{ij}}$ is nonzero when $l=j$, $k=i$, $\sum_{k\neq l} p_{kl} = 1$, we can simplify the gradient function:

$$\frac{\partial G}{\partial d_{ij}} = \eta(p_A^{ij} + p_B^{ij} - 2q_{ij})(1+d_{ij}^2)^{-1}. \tag{6.16}$$

Eq. (6.16) can be substituted into Eq. (6.14). We can obtain the gradient of the JSD between P and Q:

$$\frac{\partial G}{\partial \mathbf{v}_i} = 2\eta\sum_{j=1}^{N}\frac{(p_A^{ij}+p_B^{ij}-2q_{ij})(\mathbf{v}_i-\mathbf{v}_j)}{1+\|\mathbf{v}_i-\mathbf{v}_j\|^2}. \tag{6.17}$$

Then for minimizing O_f, each gradient of $\mathscr{L}$ are considered as zero:

$$\frac{\partial \mathscr{L}}{\partial V} = 2(-U^TA + U^TUV - W^TB + W^TWV) + \frac{\partial G}{\partial V} + \Psi = \mathbf{0}, \tag{6.18}$$

$$\frac{\partial \mathscr{L}}{\partial U} = 2(-AV^T + UVV^T) + \Phi = \mathbf{0}, \tag{6.19}$$

$$\frac{\partial \mathscr{L}}{\partial W} = 2(-BW^T + WVV^T) + \Theta = \mathbf{0}. \tag{6.20}$$

Moreover, we have KKT conditions: $\Phi_{ij}U_{ij} = 0$, $\Theta_{ij}W_{ij} = 0$ and $\Psi_{ij}V_{ij} = 0$, $\forall i, j$. Then V_{ij}, U_{ij} and W_{ij} are Multiplied in the corresponding positions on both sides of Eqs. (6.18), (6.19) and (6.20) respectively, we obtain

$$\left(2(-U^TA+U^TUV-W^TB+W^TWV)+\frac{\partial G}{\partial \mathbf{v}_i}\right)_{ij} V_{ij}=0, \tag{6.21}$$

$$2(-AV^T + UVV^T)_{ij}U_{ij} = 0, \tag{6.22}$$

$$2(-BV^T + WVV^T)_{ij}W_{ij} = 0. \tag{6.23}$$

Note that

$$\begin{aligned}\left(\frac{\partial G}{\partial \mathbf{v}_j}\right)_i &= \left(2\eta \sum_{k=1}^{N} \frac{(p_A^{jk} + p_B^{jk} - 2q_{jk})(\mathbf{v}_j - \mathbf{v}_k)}{1 + \|\mathbf{v}_j - \mathbf{v}_k\|^2}\right)_i \\ &= 2\eta \sum_{k=1}^{N} \frac{(p_A^{jk} + p_B^{jk} - 2q_{jk})(V_{ij} - V_{ik})}{1 + \|\mathbf{v}_j - \mathbf{v}_k\|^2}.\end{aligned}$$

The multiplicative update rules of bases of both W and U for any i and j are obtained:

$$U_{ij} \leftarrow \frac{(AV^T)_{ij}}{(UVV^T)_{ij}} U_{ij}, \tag{6.24}$$

$$W_{ij} \leftarrow \frac{(BV^T)_{ij}}{(WVV^T)_{ij}} W_{ij}. \tag{6.25}$$

The update rule of the shared space preserving coefficient matrix V between RGB and depth data spaces is:

$$V_{ij} \leftarrow \frac{(U^TA)_{ij}+(W^TB)_{ij}+\Upsilon}{(U^TUV)_{ij}+(W^TWV)_{ij}+\Gamma} V_{ij}, \tag{6.26}$$

where for simplicity, we let $\Upsilon = \eta \sum_{k=1}^{N} \frac{(p_A^{jk}+p_B^{jk})V_{ik}+2q_{jk}V_{ij}}{1+\|\mathbf{v}_j-\mathbf{v}_k\|^2}$, $\Gamma = \eta \sum_{k=1}^{N} \frac{(p_A^{jk}+p_B^{jk})V_{ij}+2q_{jk}V_{ik}}{1+\|\mathbf{v}_j-\mathbf{v}_k\|^2}$.

All the elements in U, V and W are nonnegative from the allocation. It proves that after every update of U, W and V, objective function is monotonically non-increasing. The proof for the convergence of U, W and V is similar with the proof in [28, 151, 293].

After U, W and V are converged, the real-valued shared structure representation by a linear projection matrix can be obtained. Since our algorithm is NMF-based, a direct projection from the target domain to the shared space does not exist for data embedding. Therefore, inspired by [27], linear regression is used to compute our projection matrix. It is equivalent to find a rotation to align the data in the current feature space with another, which is a classic Orthogonal Procrustes problem [211]. Through solving this problem, we can make the projection orthogonal:

$$\min_{\mathscr{P}} \|\mathscr{P}A - V\|, s.t.\ \mathscr{P}^T\mathscr{P} = I, \tag{6.27}$$

where $\mathscr{P}$ is the orthogonal projection for target domain. According to [290], the advantages on using orthogonal projection can be summarized as: 1) It preserves the Euclidean distance between points; 2) It distributes the variance across the dimensions more evenly; 3) Since orthogonal projection learns maximally uncorrelated dimensions, it can lead more compact representations. For the optimal solution, the singular value decomposition algorithm is firstly used to decompose the matrix: $A^TV = Q\Sigma S^T$. Then we calculate $\mathscr{P} = S\Lambda Q^T$, where Λ is the connection matrix $\Lambda = [I, \mathbf{0}] \in \mathbb{R}^{D\times M}$ where $\mathbf{0}$ is a zero matrix. Once we obtain the orthogonal projection $\mathscr{P}$, RGB data in the target domain $\hat{\mathbf{a}} \in \mathbb{R}^{M_1\times 1}$ can be projected into the latent space:

$$\mathbf{v}_{\hat{\mathbf{a}}} = \mathscr{P}\hat{\mathbf{a}}. \tag{6.28}$$

6.4.5 Adaptive Embedding

Although our above Visual-Depth Embedding (VDE) can correct the noise by projecting RGB into the shared space, the domain shifting problem remains unsolved. In the following, we propose an adaptive strategy to make VDE adaptive to target domain RGB data. In aVDE, we define the target domain as $\mathscr{D}_t = [\hat{\mathbf{a}}_1, \cdots, \hat{\mathbf{a}}_{N_t}]$. The projected target domain is defined as $\mathscr{P}_{\mathscr{D}_t} = [\mathbf{v}_{\hat{\mathbf{a}}_1}, \cdots, \mathbf{v}_{\hat{\mathbf{a}}_{N_t}}] \in \mathbb{R}^{D\times N_t}$. The shared component space is $V = [\mathbf{v}_1, \cdots, \mathbf{v}_{N_s}] \in \mathbb{R}^{D\times N_s}$.

The model proposed above learns the relationship between RGB data space and shared bases, and the shared bases are determined via both RGB data space and depth data space in the source domain. Since exploring feature matching and instance reweighting independently may not be effective enough when the domain difference is substantially large, we match features and reweight instances jointly across the latent shared space V and the new space projected from the target domain to the shared space in a principled dimensionality reduction procedure for an accurate classifier. If we only consider matching the feature distributions based on MMD minimization, it is not good enough for domain adaptation.

This strategy only matches the first- and high-order statistics, and the distribution matching is far from perfect. When the domain difference is large, there will still exist some shared latent space instances that are not relevant to the projected target instances even in the feature matching subspace. Therefore, combining feature matching and instance reweighting procedures should be considered to handle this difficult setting. But it is difficult to reweight source instances when we match the feature distributions in the infinite dimensional RKHS simultaneously. In this step, we impose the $\ell_{2,1}$-norm structured sparsity regularizer on the transformation matrix for Kernel PCA M, which can introduce row-sparsity to the transformation matrix.

We first mix projected target data with the source data, on which we perform PCA for data reconstruction. Let $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] = [\mathbf{v}_1, \cdots, \mathbf{v}_{N_s}, \mathbf{v}_{\hat{\mathbf{a}}_1}, \cdots, \mathbf{v}_{\hat{\mathbf{a}}_{N_t}}] \in \mathbb{R}^{D\times n}$ as the input data matrix, and $H = I - \frac{1}{n}\mathbf{1}$ as the centering matrix, where $n = N_s + N_t$ and $\mathbf{1}$ indicates all ones matrix, then the covariance matrix can be computed as XHX^T. PCA can find an orthogonal transformation matrix $T \in \mathbb{R}^{D\times k}$, where k is the subspace bases such that embedded data variance is maximized

$$\max_{T^T T=I} tr(T^T XHX^T T). \tag{6.29}$$

Above optimization problem can be efficiently solved by eigen-decomposition $XHX^T T = T\Omega$, where $\Omega = \text{diag}(\omega_1, \cdots, \omega_k) \in \mathbb{R}^{k\times k}$ are the k largest eigenvalues. Then we find the optimal k-dimensional representation by $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_n] = T^T X$.

To work in the RKHS $\mathscr{H}$, consider kernel mapping φ: $x \to \varphi(\mathbf{x})$, $\varphi(X) = [\varphi(\mathbf{x}_1), \cdots, \varphi(\mathbf{x}_n)]$, and kernel matrix $K = \varphi(X)^T \varphi(X) \in \mathbb{R}^{n\times n}$. We utilize the Representer theorem $T = \omega(X)M$ to kernelize PCA as

$$\max_{M^T M=I} tr(M^T KHK^T M), \tag{6.30}$$

where $M \in \mathbb{R}^{n\times k}$ is the transformation matrix for Kernel PCA. We also call M is an adaptation matrix. The subspace embedding becomes $Z = M^T K$.

Then we adopt the empirical MMD [194] as the nonparametric distance measure for comparing distributions based on the RKHS. Through k-dimensional embeddings extracted by Kernel-PCA, MMD computes the distance between the empirical expectations of shared component source and target data:

$$\left\| \frac{1}{N_s} \sum_{i=1}^{N_s} M^T k_i - \frac{1}{N_t} \sum_{j=N_s+1}^{N_s+N_t} M^T k_j \right\|_{\mathscr{H}}^2 = tr(M^T K \Delta K^T M), \tag{6.31}$$

where Δ is the MMD matrix and can be computed as

$$\Delta_{ij} = \begin{cases} \frac{1}{N_s N_s}, & x_i, x_j \in V \\ \frac{1}{N_t N_t}, & x_i, x_j \in \mathscr{P}_{\mathscr{D}_t} \\ -\frac{1}{N_s N_t}, & otherwise \end{cases} \tag{6.32}$$

Through minimizing Eq. (6.31) such that Eq. (6.30) is maximized, the first-order and high-order statistics of feature distributions are matched in the new representation $Z = M^T K$.

To impose the $\ell_{2,1}$-norm structured sparsity regularizer on the transformation matrix M, we introduce row-sparsity to the transformation matrix. The instance reweighting regularizer can be defined as

$$\|M_s\|_{2,1} + \|M_t\|_F^2, \tag{6.33}$$

where $M_s := M_{1:N_s,:}$ is the transformation matrix corresponding to the source instances, and $M_t := M_{1:N_s,:}$ is the transformation matrix corresponding to the target instances. Note that the Frobenius norm: $\|M\|_F = \sqrt{\sum_{i=1}^n \|m^i\|_2^2}$. The $\ell_{2,1}$-norm: $\|M\|_{2,1} = \sum_{i=1}^n \|m^i\|_2$. Therefore, by combining Eq. (6.31) and Eq. (6.33) into Eq. (6.30), the optimization problem is defined as

$$\min_{M^T X H X^T M = I} tr(M^T K \Delta K^T M) + \mu(\|M_s\|_{2,1} + \|M_t\|_F^2), \tag{6.34}$$

where μ is the regularization parameter to trade off feature matching and instance reweighting. In aVED, when $\mu \to 0$, aVDE optimization problem degenerates. When $\mu \to \infty$, the joint feature matching and instance reweighting is not performed. Therefore, we set $\mu = 1$.

Since $\Omega = \text{diag}(\omega_1, \cdots, \omega_k) \in \mathbb{R}^{k \times k}$ is denoted as the Lagrange multiplier, through deriving the Lagrange function of problem Eq. (6.34) as

$$\begin{aligned} \mathscr{L} = & tr(M^T K \Delta K^T M) + \|M_s\|_{2,1} + \|M_t\|_F^2 \\ & + tr((I - M^T K \Delta K^T M)\Omega). \end{aligned} \tag{6.35}$$

Let $\frac{\partial \mathscr{L}}{\partial M} = 0$, we obtain generalized eigendecomposition

$$(K \Delta K^T + \widehat{\mathscr{G}}) M = K H K^T M \Omega. \tag{6.36}$$

$\widehat{\mathscr{G}}$ is a diagonal sub-gradient matrix with ith element equal to

$$\widehat{\mathscr{G}}_{ii} = \begin{cases} \dfrac{1}{2\|m^i\|}, & x_i \in V, \quad m^i \neq 0 \\ 0, & x_i \in V, \quad m^i = 0 \\ 1, & x_i \in \mathscr{P}_{\mathscr{D}_t} \end{cases} \tag{6.37}$$

The optimal adaptation matrix M is then reduced to solve Eq. (6.36) for the k smallest eigenvectors. An adaptive classifier f can be obtained by training on $\{M^T k_i, yi\}_{i=1}^{N_s}$. The convergence analysis of our adaptive embedding is similar to the methods in [189] [281]. Finally, Algorithm 1 provides the details of aVDE. Since labeled data and unlabeled data are from different distributions that leads to impossibly tuning the optimal parameters using cross validation, following [82], nearest neighbor classifier (NN) which does not require tuning cross-validation parameters is chosen as the base classifier.

Algorithm 1 adaptive Visual-Depth Embedding (aVDE)

Require:
The source domain $\mathscr{D}_s$: $A \in \mathbb{R}^{M_1 \times N}$ and $B \in \mathbb{R}^{M_2 \times N}$; the target domain $\mathscr{D}_t$; number of bases D; the subspace bases k; the regularization parameter $\{\eta, \mu\}$; ground truth Y in source domain;.

Ensure: The basis matrix U, W, adaptation matrix M, embedding Z, adaptive classifier f.

1: Initialize V, W and U with random uniformly distributed values.
2: **repeat**
3: Calculate the matrixes U, W and the shared structure representation matrix V via Eqs. (6.24), (6.25) and (6.26);
4: **until** convergence
5: The matrix $A^T V$ is decomposed by SVD for $Q\Sigma S^T$ and compute $\mathscr{P} = S\Omega Q^T$.
6: Shared component embedded representation of the coming target domain data $\mathbf{v}_{\hat{\mathbf{a}}} \in \mathbb{R}^{D\times 1}$ is defined in Eq. (6.28).
7: Compute MMD matrix Δ by Eq. (6.32), and kernel matrix K by $K_{ij} \leftarrow K(x_i, x_j)$ where $K(\cdot,\cdot)$ is a predefined kernel. Set $\Delta \leftarrow \Delta / \|\Delta\|_F, \widehat{\mathscr{G}} \leftarrow I$;
8: **repeat** Solve Eq. (6.36) and choose the k smallest eigenvectors to construct the adaptation matrix M, and $Z \leftarrow M^T K$. Update $\widehat{\mathscr{G}}$ by Eq. (6.37);
9: **until** convergence
10: Obtain an adaptive classifier f by training on $\{M^T k_i, yi\}_{i=1}^{N_s}$.

6.4.6 Computational Complexity Analysis

The computational complexity of aVDE consists of three parts. We compare our shared component part in Eq. (6.5) and the cost of the basic NMF algorithm in [151]. Assuming that the shared latent space dimensionality for decomposition of an $M_1 \times N$ matrix A and

an $M_2 \times N$ matrix B is D, the computational complexity of the shared component part is $O(max\{M_1ND, M_2ND\})$ per iteration. In [151], through the basic NMF algorithm, A and B have the complexity $O(M_1ND)$ and $O(M_2ND)$ respectively. The first part of aVDE has the same complexity as the basic NMF. The second part is the computation of matrices P_A, P_B and Q with the complexity $O(2N^2D)$. The adaptive embedding step with the complexity $O(kn^2 + mn^2)$ is the third part. We can then obtain the final computational complexity of aVDE is: $O(max\{M_1ND, M_2ND\}t_1 + 2N^2D + t_2kn^2 + mn^2)$, where t_1 is the number of iterations when learning shared source space V, *i.e.*, from Line 1 to Line 4. t_2 is the number of iterations when learning adaptive classifier f, *i.e.*, from Line 5 to Line 9.

6.5 Experiments and Results

In this section, we evaluate the effectiveness of our aVDE for object recognition and scene classification on two pairs of datasets. The detailed experimental settings, relevant experimental results, important parameter analysis and algorithm analysis are shown in the rest of this section. All experiments are performed using Matlab 2014a on a server configured with a 16-core processor and 500G of RAM running the Linux OS.

6.5.1 Datasets

Object Recognition The RGB-D Object dataset is chosen as the source domain and the Caltech-256 dataset is considered as the target domain for object recognition. Caltech-256 dataset contains only RGB images. They share ten common categories: "calculator", "ball", "coffee mug", "cereal box", "Flashlight", "light bulb", "keyboard", "mushroom", "tomato" and "soda can". Since the RGB-D Object dataset is recorded as video sequences, we uniformly choose images with an interval of two seconds for each category resulting in 2059 training samples in the source domain. Note that each RGB image corresponds to a depth image. The 1131 RGB images in the Caltech-256 dataset are used as the target domain to evaluate our aVDE.

Scene Classification For scene classification, the NYU Depth v1 dataset is selected as the source domain and the Scene-15 dataset is chosen as the target domain. Scene-15 dataset contains only RGB images. We use the same four categories of NYU Depth v1 and Scene-15 datasets, "bedroom", "kitchen", "living room" and "office" to demonstrate our proposed algorithm. Finally, we have 907 RGB-D training image pairs in the source domain and 930 RGB images in the target domain to evaluate the performance of aVDE.

6.5.2 The Selected Methods and Settings

In our experiment, for a comprehensive and fair comparison, we select following five categories as the baselines including: 1) Naive Approach: SVM_A and 1-Nearest Neighbor Classifier which are trained by the RGB features in the source domain without considering the domain adaptation and the depth information compensation; 2) Multi-view Learning: Kernelisation of Canonical Correlation Analysis (KCCA) [96] and SVM2K [65] which use the two-view data in the source domain for training; 3) Learning Using Privileged Information: SVM+ [255] and Rank Transfer (RT) [217] which use the additional depth features in the source domain as privileged information; 4) Unsupervised Domain Adaptation: Kernel Mean Matching (KMM) [110], Domain Adaptation Machine (DAM) [59], Sampling Geodesic Flow (SGF) [85], TCA [194], Landmark (LMK) [81], Subspace Alignment (SA) [69], Geodesic Flow Kernel (GFK) [82], UNE [201] and Domain Invariant Projection (DIP) [10] which use the visual features from both domains for training the classifiers, and then predict target data based on the visual features. 5) Using Privileged Information and Unsupervised Domain Adaptation: Domain Adaptation from Multi-view to Single-view (DAM2S) which uses the additional depth features in the source domain as privileged information and reduces the data distribution mismatch between the source and target domains.

We take the factor of feature performance into consideration, and then choose shallow features and deep features to evaluate aVDE respectively. For shallow features, we extract Gradient kernel descriptors (KDES) features and LBP KDES features [20] which are successful in RGB-D object dataset from each pair of RGB/depth images. The vocabulary size is set as 1000. Three level of pyramids (1×1, 2×2, 3×3) are used. For deep features, we choose ImageNet-CNN features [136] which are learned from the pre-trained Caffe model [123] on image classification dataset (*i.e.* ImageNet) for object classification, and the Places-CNN [294] scene features which are learned from the pre-trained Caffe model on scene classification dataset (*i.e.* Places dataset) for scene classification. Both of these two kinds of models obtain great success for object and scene classification respectively. The feature dimension after CNN is 4096. Note that the depth image is encoded as HHA image as in [93] before extracting the features.

From Eq. (6.11) and algorithm 1, the size of matrices $U \in \mathbb{R}^{M_1 \times D}$, $W \in \mathbb{R}^{M_2 \times D}$ and $V \in \mathbb{R}^{D \times N_s}$ should be predefined. M_1, M_2 and N_s are known when the data is given. However, the value of number of latent bases D is difficult to be pre-determined. In aVDE, an improper D will result in the limitation of identification of latent topics or the increase of possibility of overfitting. In order to investigate the effects of D, we choose different number of bases, *e.g.*, 40, 60, 80, 100, 120 and 140. We also explore the sensitivity of the parameter η in Eq. (6.11) on the performance of aVDE. We set the parameter η by searching $\eta \in$

$\{0, 1/8, 1/4, 1/2, 1, 2, 4, 8\}$. In particular, although the subspace bases k is also related, we can find that when k is small, data reconstruction is accurate in general. Therefore, when comparing with the baseline methods, we set $k = 20$. We consider the maximum number of t_1 as 1000, and let t_2 = 10 in aVDE learning phase.

6.5.3 Experimental Results

We evaluate all selected methods by strictly choosing the parameters according to their original papers, and then report the best results of each method. The two pairs of datasets which are used to validate the proposed method share some common objects and scene categories. However, these common objects and scene categories still suffer from the domain shifting problem due to conventional surveillance cameras and depth sensors are using different mechanisms. Meanwhile, in realistic scenarios, the source and target domains follow different distributions, especially when images are acquired from different cameras, or in various conditions. The selected two pairs of datasets are from different cameras. RGB-D datasets are created by Kinect. RGB datasets which are chosen as the target domain are created by some RGB camera. Therefore, these common objects and scene categories are from different domains. Some related papers which focus on domain adaptation can be found in [82] [81]. The experimental results of aVDE compared with the 16 baseline methods discussed before on the two pairs of source and target domains are reported in Table 6.2. In Table 6.2, the first column is the number corresponding to the category of the selected methods, the second column indicates method names, the third and forth column present the recognition results when the RGB-D object dataset is for training and the Caltech-256 dataset is for testing, and the fifth and sixth column give recognition rate when the NYU Depth v1 is for training and the Scene-15 dataset is for testing. We test the shallow and deep features on both of these two pairs of datasets. In addition, we also illustrate some samples with highest recognition accuracies from selected datasets in Fig. 6.3.

(a) tomato, shallow features accuracy = 70.31%, deep features accuracy = 92.86%

(b) bedroom, shallow features accuracy = 69.78%, deep features accuracy = 89.66%

Fig. 6.3 Example images with highest accuracy results from five selected dataset pairs.

From Table 6.2, we observe that our method outperforms all other baseline methods,

Table 6.2 Accuracies (%) for object recognition and scene classification with shallow and deep features (bold numbers indicate the best results).

	Methods	**Object→ Caltech-**256		**NYU v1→ Scene-**15	
		KDES	**ImageNet -CNN**	**KDES**	**Places -CNN**
1	SVM_A	18.21	47.21	17.42	49.46
	1-NN	18.30	48.36	19.78	50.75
2	KCCA	18.39	49.60	19.68	53.33
	SVM2K	20.79	51.72	21.61	53.23
3	SVM+	18.57	48.63	19.46	51.94
	RT	17.15	46.51	16.77	49.03
4	KMM	18.13	47.21	17.53	49.57
	DAM	18.21	49.60	17.10	49.25
	SGF	19.27	50.04	19.25	55.27
	TCA	25.11	56.23	22.04	59.03
	LMK	19.45	52.34	25.81	54.73
	SA	21.13	54.64	27.42	62.69
	UNE	24.76	56.23	26.34	59.68
	GFK	18.48	51.02	24.19	53.23
	DIP	25.46	57.38	25.48	58.60
5	DA-M2S	30.06	61.54	31.08	64.52
	aVDE	**35.75**	**70.18**	**33.98**	**69.46**

sometimes by a large margin. It demonstrates the effectiveness of our method by exploring additional depth images in the source domain and reducing the domain distribution mismatch between the source and target domains. From the results, we find that RT performs the worst which possibly because it is based on Rank SVM which is designed for ranking task rather than classification task. SVM_A and 1-NN which do not consider the depth information and domain discrepancy perform poorly. KCCA, SVM2K and SVM+ obtain better performance generally when compared with SVM_A and 1-NN by utilizing the additional depth features. However, these three methods do not reduce the distribution mismatch between the source and target domains. The domain adaptation methods as KMM and DAM perform in a general way or even worse than SVM_A and 1-NN, which maybe because both approaches are unsuitable in this application. SGF, TCA, LMK, SA, UNE, GFK and DIP perform better than other nonadaptation methods, which reveals that considering the domain mismatch across domains is useful. Our proposed aVDE also outperforms DA-M2S which uses privileged information and unsupervised domain adaptation as well. It is possible because the domain mismatch between our shared latent space and the projected target domain is less than the domain mismatch in DA-M2S.

Additionally, from the comparison of shallow and deep features, we can observe that all deep features have higher classification performances than shallow features. For example, the accuracy of our aVDE method on object classification task increases from 35.75% to 70.18%, which indicates that the deep features can effectively remove the domain bias. It is possible because that the deep learning models (*i.e.* ImageNet model and Places model) are pre-trained by abundant images which are from different datasets and webs. Note that the proposed method still outperforms other methods with deep features.

6.5.4 Parameter Sensitivity Analysis

In the proposed aVDE, there are two parameters D and η involved for model tuning. We demonstrate the accuracies with different values of D from {40, 60, 80, 100, 120, 140} and different values of η from $\{0, 1/8, 1/4, 1/2, 1, 2, 4, 8\}$ on two pairs of datasets with the shallow and deep features in Fig. 6.4. From Fig. 6.4, we can find that with the increase of number of bases, the performance of aVDE (with $\eta \in \{0, 1/8, 1/4, 1/2, 1, 2, 4, 8\}$) becomes better and better until around 100 bases in general. The former three cases reach the best point when $\eta = 1/2$ and $D = 100$, and the forth case achieves the highest point when $\eta = 1/2$ and $D = 120$. In addition, when η is zero, the accuracy is lowest which indicates that learning without this regularization leads to poor performance. Therefore, we can conclude that the regularization term is important for our algorithm.

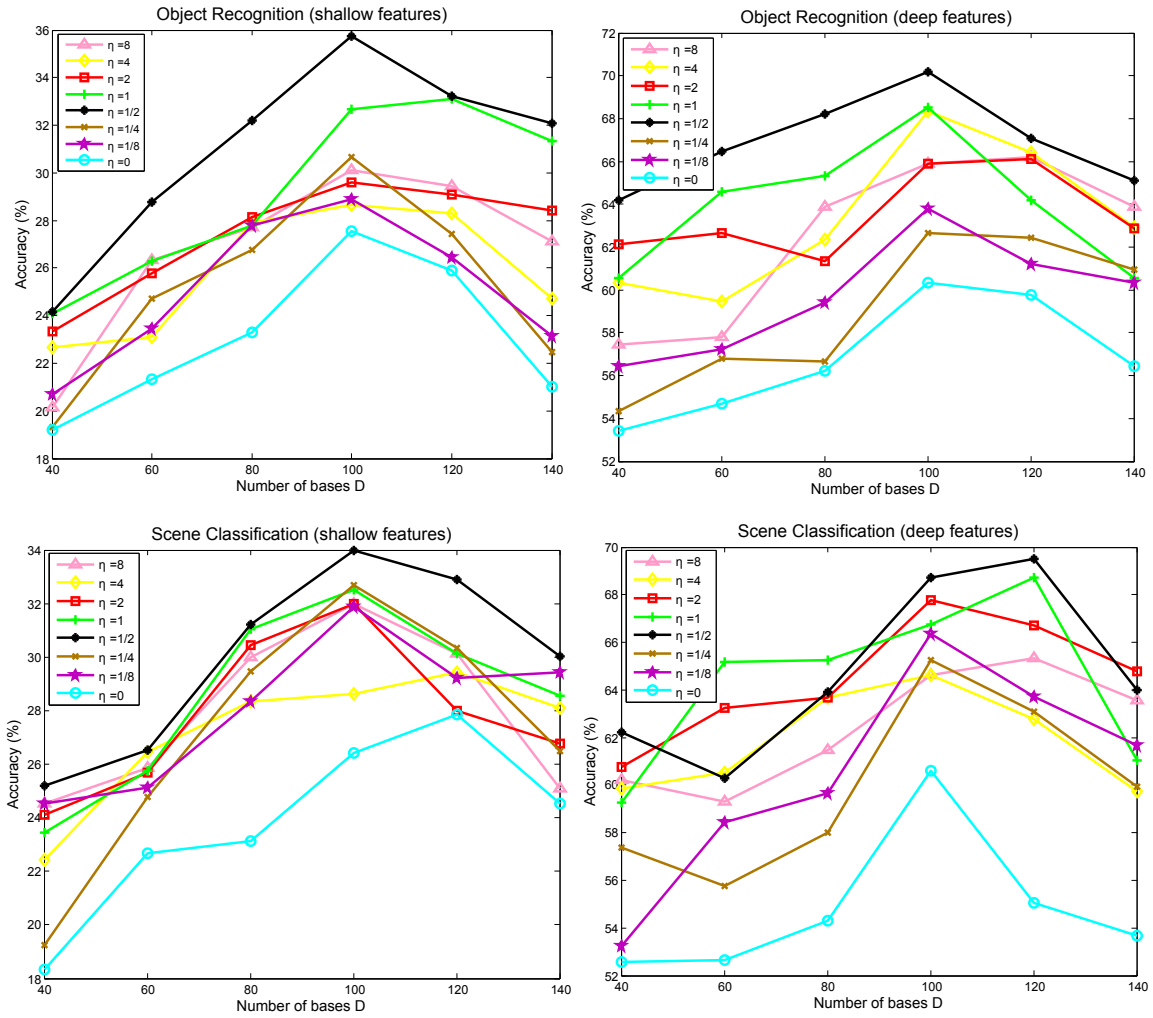


Fig. 6.4 Parameter sensitivity analysis on the considered datasets with the shallow and deep features.

Table 6.3 Comparison of accuracies (%) between aVDE and two special cases.

	Object→ Catech-256		**NYU v1→ Scene-**15	
	KDES	**ImageNet -CNN**	**KDES**	**Places -CNN**
aVE	27.67	59.15	27.31	62.04
VDE	22.81	53.58	23.76	55.70
aVDE	**35.75**	**70.18**	**33.98**	**69.46**

6.5.5 Analysis on aVDE

We explore two special cases of our aVDE for a better understanding of our algorithm.
Case1: We do not consider depth information, which is denoted as aVE. We remove $\|B - WV\|^2$ in Eq. (6.5) and $KL(P_B\|Q)$ in Eq. (6.6) which result in the minimization of another objective function as:

$$\min_{U,V} \|A - UV\|^2 + \frac{\eta}{2} KL(P_A\|Q), s.t.\ U,V \geq 0. \tag{6.38}$$

Case2: We do not consider domain adaptation, which is denoted as VDE. We directly use the V which is acquired from Eq. (6.26) to build a NN classifier. Then the embedded representation of the coming RGB target domain data $\hat{\mathbf{a}} \in \mathbb{R}^{M_1 \times 1}$ can be obtained as $\mathbf{v}_{\hat{\mathbf{a}}}$ by Eq. (6.28).

From Table 6.3, we can find that the results of the special cases are worse than aVDE, which shows it is beneficial to exploit the additional depth features and domain adaptation for learning an adaptive classifier.

6.6 Summary

In this chapter, we have proposed a novel method aVDE which can utilize the additional depth information in the source domain and simultaneously reduce the domain mismatch between the source and target domains. The latent shared space is identified in Visual-Depth embedding. Aiming to alleviate the mismatch between data distributions, aVDE matches features and reweights instances jointly across the shared latent space and the projected target domain in a principled dimensionality reduction procedure. On two real-world image datasets, the experimental results illustrate that the proposed method significantly outperforms the state-of-the-art methods.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis mainly focuses on the exploration of RGB-D data. We solve the problems in RGB-D areas: performance of deep learning models in RGB-D datasets, hyper-parameter optimization, RGB-D data fusion and recognizing RGB information from RGB-D data in the previous chapters. In this section, the contributions of this thesis are briefly concluded.

In the first part, four prevalent deep learning models (*i.e.*, Deep Belief Networks (DBNs), Stacked Denoising Auto-Encoders (SDAE), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) Neural Networks) are chosen for evaluation. We conduct extensive experiments on five popular RGB-D datasets including three image datasets and two video datasets. A detailed analysis about the comparison between the learned feature representations from the four deep learning models is presented. In addition, a few suggestions on how to adjust hyper-parameters for learning deep neural networks are made in this part. In conclusion, we note that RGB-D fusion methods using CNNs with numerous training samples always outperform our other selected methods (DBNs, SDAE and LSTM). Since LSTM can learn from experience to classify, process and predict time series, it achieved better performances than DBN and SDAE in video classification tasks. According to the extensive experimental results, this evaluation will provide insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion.

The second part presents a simple and efficient framework for improving the efficiency and accuracy of hyper-parameter optimization by considering the classification complexity of a particular dataset. This new framework bridges dataset classification complexity measures and hyper-parameters optimization in deep learning models. Through this framework, we can quickly choose an initial set of hyper-parameters that are suitable for a new classification task, thus reducing the number of trials in the hyper-parameter space. This

new framework is easy and practical which improves the efficiency and accuracy of hyper-parameter optimization by combining classification complexity and hyper-parameter optimization. To verify our assumption, we choose three representative deep learning models on six real-world RGB-D datasets. After the analysis of experiments, we confirm that our framework can provide deeper insights into the relationship between dataset classification tasks and hyperparameters optimization, thus quickly choosing an accurate initial set of hyper-parameters for a new coming classification task.

In the third part, based on the observation of the category of one unknown scene mainly relying on the object-level information which includes appearance, texture, shape and depth of each object and the structural distribution of different objects, we propose a new Convolutional Neural Networks (CNNs)-based local multi-modal feature learning framework (LM-CNN) for RGB-D scene classification. This method can effectively capture much of the local structure from the RGB-D scene images and automatically learn a fusion strategy for the object-level recognition step instead of simply training a classifier on top of features extracted from both modalities. The experimental results show that our method with local multi-modal CNNs greatly outperforms state-of-the-art approaches. This part has the potential to significantly improve RGB-D scene understanding. An extended evaluation shows that our local fine-tuning method outperforms direct global fine-tuning methods. The experiments also show that CNNs trained using a scene-centric dataset is able to achieve an improvement on scene benchmarks compared to a network trained using an object-centric dataset.

The forth part is motivated by the physical concept that range data correspond to the phase change and color information corresponds to the intensity. We propose a new method which can better explore the correlation between the RGB pixel and the corresponding depth pixel for RGB-D data fusion. It makes the correlated and individual parts of the RGB-D information in the new feature space well combined. The experimental results show that our method achieves competing performance against the classical SIFT.

In the fifth part, we aim to simultaneously solve two challenges in recognizing RGB images from RGB-D data: 1) how to take advantage of the additional depth information in the source domain? 2) how to reduce the data distribution mismatch between the source and target domains? We propose a novel method called adaptive Visual-Depth Embedding (aVDE) which learns the compact shared latent space between two representations of labeled RGB and depth modalities in the source domain first. Then the shared latent space can help the transfer of the depth information to the unlabeled target dataset. At last, aVDE matches features and reweights instances jointly across the shared latent space and the projected target domain for an adaptive classifier. On two pairs of real-world image datasets,

the experimental results illustrate that the proposed method significantly outperforms the state-of-the-art methods.

In addition, the computational efficiency of recent deep learning methods are also discussed. Though deep learning methods have shown a lot of advantages in many applications, the computational efficiency is still a difficult task for large scale data. Deep learning methods can achieve higher accuracies through a large number of hidden neurons. The iterative computations in most of the deep learning methods are always extremely difficult to be parallelized. The amount of recent commercial and academic data increases quickly. Above conditions result in that deep learning models need a significant amount of computational burden to reach state-of-the-art performances on large size datasets.

Deep learning methods consider many training parameters, for example the learning rate, number of hidden layers, units for each layer and initial weights. Parameter optimization by sweeping through the parameter space is not suitable on account of the computational resources and time cost. Some tricks which are proposed in Chapter 2 such as batching which can compute the gradient on a number of training examples can speed up the computation. Since the computation of the vector and matrix is well-suited for GPUs, GPUs based deep learning models can speed up the training significantly. In the experiments of Chapter 2, though the size of the selected datasets is not large scale, however, the implementation of the deep learning methods is based on CPU, therefore, the test for one set of hyperparameters will still cost over 12 hours.

Some researchers have focused on large-scale learning. Deng et al. [57] present the Deep Stacking Network (DSN), which can overcome the problem of parallelizing learning algorithms for deep architectures. The DSN can stack simple processing modules in building deep architectures with a convex learning problem in each module. Following above work, a novel deep architecture called Tensor Deep Stacking Network (T-DSN) [112] is presented, which is based on the DSN. T-DSN is implemented using CPU clusters for scalable parallel computing. Great computing power for speeding up the training process in large scale data deep learning obtains a lot of attention. The use of multiple CPU cores can scale up DBNs, with each core dealing with a subset of training data. In [254], some implementations which can enhance the performance of modern CPUs more for deep learning are proposed. Moreover, [199] shows that the GPU-based implementation can increase the speed of DBN with one million samples and 45 million parameters in a RBM. In their experiments, it is around one day for CPU-based implementation and 30 minutes for GPU-based implementation.

In summary, large-scale data deep learning has continuous progress in recent years. It shows that single CPU is inefficient for deep learning with a large model and large-scale data

in [199]. However, the running time will not be a big concern with many machines [55]. Some significant research progress in large-scale data deep learning can be found in [46] [136]. From above discussions, we can conclude that major research efforts direct towards experiments with GPUs.

7.2 Future Work

This thesis starts with the introduction of low-cost Kinect and various RGB-D datasets, then provides some ground work to explore the applications of RGB-D information in computer vision field. In this section, we will give several extensions, potential new frontiers and applications about these works in the future.

RGB-D datasets. With the release of Microsoft Kinect v1, from 2011 to 2017, the background, illumination conditions and occlusion levels in the datasets become more and more challengeable [94]. Therefor, there are a great many RGB-D datasets for benchmarking. In the future, I will solve the lack of one comprehensive and systematic description about the popular RGB-D datasets. Each dataset from our collection is still allocated into one of five categories, object detection and tracking, human activity analysis, object and scene recognition, SLAM (Simultaneous Localization and Mapping) and hand gesture analysis. The characteristics, such as ground truth, number of references about each dataset also need to be carefully collected. This work will show the direction of the future RGB-D research in computer vision and robotics. Following the steps of the release of Microsoft Kinect v2, an increasing number of new available RGB-D datasets will be created for the research of unsolved vision problems.

Correlation between dataset complexity and deep learning model hyper-parameters. Since it has been verified that our framework plays a significant role on hyper-parameter optimization among a small scale of datasets, to extent this work, we illustrate a scenario for further research around our framework. In Chapter 3, we can only obtain six complexity feature vectors through our six datasets. It is still not enough. It needs us and other researchers to collect more complexity feature vectors from other datasets, which can make the selection of initial set of hyper-parameters closer to the optimal set. Meanwhile, more complexity feature vectors can speed up the optimization process. Furthermore, the collection of more datasets can make the hyper-parameter sets heterogeneous on different kinds of machine learning methods. Through the contribution from other researchers, deep learning will become easier to use under this framework in the future. We will collect more complexity feature vectors from other datasets to implement the aim.

Collection of large scale RGB-D dataset. To outperform hand-crafted features, deep learn-

ing models need numerous training samples (over one million samples) to train the models for extracting robust features. Some popular large scale RGB datasets such as imageNet always have several million images. However, we still cannot find a large scale RGB-D dataset till now. It is because that though we have convenient RGB-D sensor Kinect to obtain RGB-D images now, it is still impossible to shoot that huge number of images for a dataset in real world. Recently, some research proposed that it can estimate the depth of a single RGB image without any additional knowledge [298] [165]. These proposed research result in that we can obtain RGB-D images from large scale RGB datasets, which can significantly save the time. In the future, we will focus on collecting a large-scale RGB-D dataset for better gauging new algorithms and finding convenient ways to adjust hyper-parameters. We confirm that this dataset should contain three parts: 1) Urban areas. Urban areas can make this dataset have more different kinds of scenes. Then this dataset can be applied into both indoor scene tasks and outdoor scene tasks. 2) Indoor scenes. Indoor scene classification has received increasing attention in both academia and industry over the past few years. It plays an important role for a wide range of practical applications. 3) Maya software. Through Maya software, we can gain some synthetic depth images. Compare to the depth images from Kinect, these images do not contain any noise. One example image which is created by us is shown in Fig. 7.1. There are still some challenges in the creation of synthetic depth images. We will focus more on the follow-up work. Furthermore, the preparing large scale RGB-D dataset should have a rich scene taxonomy. The preparing large scale RGB-D dataset will have the same list of scene categories (about 476 scene categories) with the Places dataset. At the end, the benchmarks including classification accuracy and top-5 error rate on this large scale RGB-D dataset will be provided.

RGB-D fusion in complex space. As we have mentioned in Chapter 5 that proposed $\mathbb{C}$-SIFT is just an example to show the advantages of the fusion method. CNNs, DBNs, GIST or other methods can also be introduced into complex space as well. To extend the proposed RGB-D fusion method into deep learning methods can be a direction in the future. Take CNNs for example, a complex valued CNN model can be built with complex input and weights. Specifically, the optimization method for this network can be handled, and the back propagation algorithm can also be modified. The general way to train neural networks is through the iterative gradient descent algorithm. However, the loss function is real-valued with complex weights in the complex case. This function cannot be differentiable everywhere. Meanwhile, steepest descent direction is not able to be calculated using the gradient. The Wirtinger derivatives can be used to adjust gradient based methods to the complex domain. In addition, the labels in many applications are real-valued. Also the network's output is real-valued. To that end, a projection layer can be added as a special case of an activation

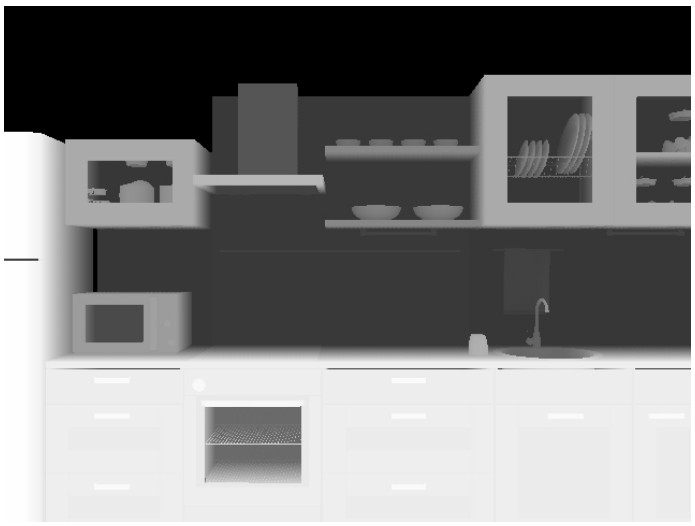


Fig. 7.1 One example scene image from Maya software.

function layer.

Domain adaptation. Though the RGB-D domain adaptation method aVDE which we proposed in Chapter 6 outperforms the state-of-the-art methods on two real-world image datasets, it can still be improved. The proposed aVDE learns the compact shared latent space between two representations of labeled RGB and depth modalities in the source domain first. Then the shared latent space can help the transfer of the depth information to the unlabeled target dataset. At last, aVDE matches features and reweights instances jointly across the shared latent space and the projected target domain for an adaptive classifier. The limitation is that it does not consider the correlation among source RGB images, source depth images and target RGB images. In the future, we can extend aVDE into a method which can reduce the mismatch between source RGB images and target RGB images, source depth images and target RGB images simultaneously instead of a medium process which projects target domain into the shared latent space. Besides the extension of aVDE, some direction with potential scientific contributions about the domain adaptation are also provided. It shows that addressing domain adaptation in recent few papers only focuses on recognition and detection tasks. Other tasks such as object segmentation, human pose or action analysis can also be extended into domain adaptation area. Above challenging tasks are addressed by deep learning methods recently. These deep learning methods need large amount of labeled data. How to adapt these deep learning models between domains with limited amount of data is one challenge that should be addressed by the visual domain adaptation community in the future.

In addition, domain shifts are usually due to some physical causes (sensor changes,

illumination changes, etc.). Incorporating these physical priors into statistical adaptation approaches may lead to performance increase. This can better explore the knowledge about image formation and better integrate other domain knowledge implied by noisy, partial and imagery. It is expected that through incorporating physically informed adaptation paradigm appropriately, distributional changes among different sensors can be handled.

References

[1] Abdallah, D. and Charpillet, F. (2015). Pose estimation for a partially observable human body from rgb-d cameras. In *International Conference on Intelligent Robots and Systems*, page 8.

[2] Aggarwal, J. K. and Cai, Q. (1997). Human motion analysis: A review. In *Nonrigid and Articulated Motion Workshop*, pages 90–102.

[3] Aldoma, A., Tombari, F., Di Stefano, L., and Vincze, M. (2012). A global hypotheses verification method for 3d object recognition. In *European Conference on Computer Vision*, pages 511–524.

[4] Allen, P. (2012). *Robotic object recognition using vision and touch*, volume 34.

[5] Arandjelovic, R. and Zisserman, A. (2013). All about vlad. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1578–1585.

[6] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.

[7] Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335.

[8] Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2011). Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39.

[9] Bai, J., Wu, Y., Zhang, J., and Chen, F. (2015). Subset based deep learning for rgb-d object recognition. *Neurocomputing*.

[10] Baktashmotlagh, M., Harandi, M. T., Lovell, B. C., and Salzmann, M. (2013). Unsupervised domain adaptation by domain invariant projection. In *IEEE International Conference on Computer Vision*, pages 769–776.

[11] Barbosa, B. I., Cristani, M., Del Bue, A., Bazzani, L., and Murino, V. (2012). Re-identification with rgb-d sensors. In *First International Workshop on Re-Identification*, pages 433–442.

[12] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Europen Conference on Computer Vision*, pages 404–417.

[13] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[14] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153.

[15] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

[16] Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

[17] Bian, W. and Tao, D. (2010). Biased discriminant euclidean embedding for content-based image retrieval. *IEEE Transactions on Image Processing*, 19(2):545–554.

[18] Blum, M., Springenberg, J. T., Wülfing, J., and Riedmiller, M. (2012). A learned feature descriptor for object recognition in rgb-d data. In *IEEE International Conference on Robotics and Automation*, pages 1298–1303.

[19] Bo, L., Lai, K., Ren, X., and Fox, D. (2011a). Object recognition with hierarchical kernel descriptors. In *Conference on Computer Vision and Pattern Recognition*, pages 1729–1736.

[20] Bo, L., Ren, X., and Fox, D. (2011b). Depth kernel descriptors for object recognition. In *IEEE International Conference on Intelligent Robots and Systems*, pages 821–826.

[21] Bo, L., Ren, X., and Fox, D. (2013). Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402.

[22] Borràs, R., Lapedriza, À., and Igual, L. (2012). Depth information in human gait analysis: an experimental study on gender recognition. In *Image Analysis and Recognition*, pages 98–105.

[23] Bosch, A., Zisserman, A., and Muñoz, X. (2008). Scene classification using a hybrid generative/discriminative approach. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 30(4):712–727.

[24] Boureau, Y.-l., Cun, Y. L., et al. (2008). Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1185–1192.

[25] Brochu, E., Cora, M., and de Freitas, N. (2009). November.¡°a tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning¡±. Technical report, Technical Report TR-2009-023, Department of Computer Science, University of British Columbia.

[26] Browatzki, B., Fischer, J., Graf, B., Bulthoff, H., and Wallraven, C. (2011). Going into depth: Evaluating 2d and 3d cues for object classification on a new, large-scale object dataset. In *International Conference on Computer Vision Workshops*, pages 1189–1195.

[27] Cai, D., He, X., and Han, J. (2007). Spectral regression for efficient regularized subspace learning. In *IEEE International Conference on Computer Vision*, pages 1–8.

[28] Cai, D., He, X., Han, J., and Huang, T. S. (2011). Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560.

[29] Cai, D., He, X., Wang, X., Bao, H., and Han, J. (2009). Locality preserving nonnegative matrix factorization. In *International Joint Conferences on Artificial Intelligence*, volume 9, pages 1010–1015.

[30] Cai, D., He, X., Wu, X., and Han, J. (2008). Non-negative matrix factorization on manifold. In *IEEE International Conference on Data Mining*, pages 63–72.

[31] Cai, L., Shuangjie, C., Min, X., Yu, J., and Zhang, J. (2017). Dynamic hand gesture recognition using RGB-D data for natural human-computer interaction. *Journal of Intelligent and Fuzzy Systems*, 32(5):3495–3507.

[32] Cai, Z., Han, J., Liu, L., and Shao, L. (2016). Rgb-d datasets using microsoft kinect or similar sensors: a survey. *Multimedia Tools and Applications*, pages 1–43.

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

[34] Chen, C., Jafari, R., and Kehtarnavaz, N. (2015a). Improving human action recognition using fusion of depth camera and inertial sensors. *IEEE Transactions on Human-Machine Systems*, 45(1):51–61.

[35] Chen, C., Jafari, R., and Kehtarnavaz, N. (2015b). A real-time human action recognition system using depth and inertial sensor fusion.

[36] Chen, C., Jafari, R., and Kehtarnavaz, N. (2015c). Utd-mad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *IEEE International Conference on Image Processing*.

[37] Chen, L., Li, W., and Xu, D. (2014). Recognizing rgb images by learning from rgb-d data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1418–1425.

[38] Chen, L., Wei, H., and Ferryman, J. (2013a). A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 34(15):1995–2006.

[39] Chen, X. and Cai, D. (2011). Large scale spectral clustering with landmark-based representation. In *AAAI*.

[40] Chen, X. and Koskela, M. (2014). Using appearance-based hand features for dynamic RGB-D gesture recognition. In *International Conference on Pattern Recognition*, pages 411–416.

[41] Chen, Y., Zhang, J., Cai, D., Liu, W., and He, X. (2013b). Nonnegative local coordinate factorization for image representation. *IEEE Transactions on Image Processing*, 22(3):969–979.

[42] Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. (2014). Bing: Binarized normed gradients for objectness estimation at 300fps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3286–3293.

[43] Chéron, G., Laptev, I., and Schmid, C. (2015). P-cnn: Pose-based cnn features for action recognition. In *IEEE International Conference on Computer Vision*, pages 3218–3226.

[44] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[45] Chua, C.-S., Guan, H., and Ho, Y.-K. (2002). Model-based 3d hand posture estimation from a single 2d image. *Image and Vision computing*, 20(3):191–202.

[46] Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., and Andrew, N. (2013). Deep learning with cots hpc systems. In *International Conference on Machine Learning*, pages 1337–1345.

[47] Coates, A. and Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, pages 921–928.

[48] Coates, A., Ng, A. Y., and Lee, H. (2011a). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223.

[49] Coates, A., Ng, A. Y., and Lee, H. (2011b). An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223.

[50] Cong, Y., Liu, J., Yuan, J., and Luo, J. (2013). Self-supervised online metric learning with low rank constraint for scene categorization. *IEEE Transactions on Image Processing*, 22(8):3179–3191.

[51] Cruz, L., Lucio, D., and Velho, L. (2012). Kinect and rgbd images: Challenges and applications. In *Conference on Graphics, Patterns and Images Tutorials*, pages 36–49.

[52] Cui, Z., Li, W., Xu, D., Shan, S., Chen, X., and Li, X. (2014). Flowing on riemannian manifold: Domain adaptation by shifting covariance. *IEEE transactions on cybernetics*, 44(12):2264–2273.

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

[54] Daniel, W. W. et al. (1990). Applied nonparametric statistics.

[55] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.

[56] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

[57] Deng, L., Yu, D., and Platt, J. (2012). Scalable stacking and learning for building deep architectures. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2133–2136.

[58] Deselaers, T., Alexe, B., and Ferrari, V. (2010). Localizing objects while learning their appearance. In *European Conference on Computer Vision*, pages 452–466.

[59] Duan, L., Tsang, I. W., and Xu, D. (2012). Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479.

[60] émi Bardenet, R., ty ás Brendel, M., Kégl, B., et al. (2013). Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, pages 199–207.

[61] Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the rgb-d slam system. In *IEEE International Conference on Robotics and Automation*, pages 1691–1696.

[62] Erdogmus, N. and Marcel, S. (2013). Spoofing in 2d face recognition with 3d masks and anti-spoofing with kinect. pages 1–6.

[63] Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. (2014). Tts synthesis with bidirectional lstm based recurrent neural networks. In *Interspeech*, pages 1964–1968.

[64] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Van Gool, L. (2013). Random forests for real time 3d face analysis. *International Journal on Computer Vision*, 101(3):437–458.

[65] Farquhar, J., Hardoon, D., Meng, H., Shawe-taylor, J. S., and Szedmak, S. (2005). Two view learning: Svm-2k, theory and practice. In *Advances in Neural Information Processing Systems*, pages 355–362.

[66] Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531.

[67] Feng, D., Barnes, N., You, S., and McCarthy, C. (2016). Local background enclosure for rgb-d salient object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[68] Fernandez, R., Rendel, A., Ramabhadran, B., and Hoory, R. (2014). Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In *Interspeech*, pages 2268–2272.

[69] Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. (2013). Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision*, pages 2960–2967.

[70] Förster, A., Graves, A., and Schmidhuber, J. (2007). Rnn-based learning of compact maps for efficient robot localization. In *European Symposium on Artificial Neural Networks*, pages 537–542.

[71] Fothergill, S., Mentis, H. M., Kohli, P., and Nowozin, S. (2012). Instructing people for training gestural interactive systems. In *Conference on Human Factors in Computer Systems*, pages 1737–1746.

[72] Frank, L. and Hubert, E. (1996). Pretopological approach for supervised learning. In *International Conference on Pattern Recognition*, volume 4, pages 256–260.

[73] Gao, Y., Wang, M., Tao, D., Ji, R., and Dai, Q. (2012). 3-d object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9):4290–4303.

[74] Garcia, J. and Zalevsky, Z. (2008). Range mapping using speckle decorrelation. United States Patent 7,433,024.

[75] Gasparrini, S., Cippitelli, E., Spinsante, S., and Gambi, E. (2014). A depth-based fall detection system using a kinect® sensor. *Sensors*, 14(2):2756–2775.

[76] Geem, Z. W., Kim, J. H., and Loganathan, G. (2001). A new heuristic optimization algorithm: harmony search. *simulation*, 76(2):60–68.

[77] Geng, B., Tao, D., and Xu, C. (2011). Daml: Domain adaptation metric learning. *IEEE Transactions on Image Processing*, 20(10):2980–2989.

[78] Geng, J. (2011). Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160.

[79] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

[80] Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *International Conference on Machine Learning*, pages 513–520.

[81] Gong, B., Grauman, K., and Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 222–230.

[82] Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073.

[83] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407.

[84] Gonzalez-Dominguez, J., Lopez-Moreno, I., Sak, H., Gonzalez-Rodriguez, J., and Moreno, P. J. (2014). Automatic language identification using long short-term memory recurrent neural networks. In *INTERSPEECH*, pages 2155–2159.

[85] Gopalan, R., Li, R., and Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. In *International Conference on Computer Vision*, pages 999–1006.

[86] Gossow, D., Weikersdorfer, D., and Beetz, M. (2012). Distinctive texture features from perspective-invariant keypoints. In *International Conference on Pattern Recognition*, pages 2764–2767.

[87] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[88] Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

[89] Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2006). A kernel method for the two-sample-problem. In *Neural Information Processing Systems*, pages 513–520.

[90] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset.

[91] Gu, C., Lim, J. J., Arbeláez, P., and Malik, J. (2009). Recognition using regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1030–1037.

[92] Gupta, S., Arbeláez, P., Girshick, R., and Malik, J. (2015). Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *International Journal of Computer Vision*, 112(2):133–149.

[93] Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *Europen Conference on Computer Vision*, pages 345–360.

[94] Han, J., Shao, L., Xu, D., and Shotton, J. (2013). Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334.

[95] Handa, A., Whelan, T., McDonald, J., and Davison, A. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *International Conference on Robotics and Automation*, pages 1524–1531.

[96] Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

[97] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., , and Navab, N. (2012). Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. pages 548–562.

[98] Hinton, G. (2010). A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926.

[99] Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pages 599–619.

[100] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

[101] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[102] Hnizdo, V. and Gilson, M. K. (2010). Thermodynamic and differential entropy under a change of variables. *Entropy*, 12(3):578.

[103] Ho, T. K. and Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300.

[104] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[105] Hoekstra, A. and Duin, R. P. (1996). On the nonlinearity of pattern classifiers. In *International Conference on Pattern Recognition*, volume 4, pages 271–275.

[106] Hornung, A., Wurm, K. M., and Bennewitz, M. (2010). Humanoid robot localization in complex indoor environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1690–1695.

[107] Hosang, J., Benenson, R., and Schiele, B. (2014). How good are detection proposals, really? *arXiv preprint arXiv:1406.6962*.

[108] Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5(Nov):1457–1469.

[109] Hu, G., Huang, S., Zhao, L., Alempijevic, A., and Dissanayake, G. (2012). A robust rgb-d slam algorithm. In *International Conference on Intelligent Robots and Systems*, pages 1714–1719.

[110] Huang, J., Gretton, A., Borgwardt, K. M., Schölkopf, B., and Smola, A. J. (2006). Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608.

[111] Huang, Y., Zhu, F., Shao, L., and Frangi, A. F. (2016). Color object recognition via cross-domain learning on rgb-d images. In *IEEE International Conference on Robotics and Automation*, pages 1672–1677.

[112] Hutchinson, B., Deng, L., and Yu, D. (2013). Tensor deep stacking networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1944–1957.

[113] Hutter, F. (2009). Automated configuration of algorithms for solving hard computational problems.

[114] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011a). Bayesian optimization with censored response data. In *Neural Information Processing Systems workshop*.

[115] Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011b). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523.

[116] Huynh, T., Min, R., and Dugelay, J.-L. (2012). An efficient lbp-based descriptor for facial depth images applied to gender recognition using rgb-d face data. In *Asian Conference on Computer Vision*, pages 133–145.

[117] Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430.

[118] Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80.

[119] Janoch, A., Karayev, S., Jia, Y., Barron, J. T., Fritz, M., Saenko, K., and Darrell, T. (2013). A category-level 3d object dataset: Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision, Rsearch Topics and Applications*, pages 141–165.

[120] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311.

[121] Jhuo, I.-H., Gao, S., Zhuang, L., Lee, D., and Ma, Y. (2015). Unsupervised feature learning for rgb-d image classification. In *Asian Conference on Computer Vision*, pages 276–289.

[122] Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

[123] 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. In *International Conference on Multimedia*, pages 675–678.

[124] Jin, L., Gao, S., Li, Z., and Tang, J. (2014). Hand-crafted features or machine learnt features? together they improve rgb-d object recognition. In *International Symposium on Multimedia*, pages 311–319.

[125] Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383.

[126] Jones, G. A., Paragios, N., and Regazzoni, C. S. (2012). *Video-based surveillance systems: computer vision and distributed processing*.

[127] Karpathy, A., Miller, S., and Fei-Fei, L. (2013). Object discovery in 3d scenes via shape analysis. In *International Conference on Robotics and Automation (ICRA)*, pages 2088–2095.

[128] Kennedy, J. (2011). Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766.

[129] Kepski, M. and Kwolek, B. (2014). Fall detection using ceiling-mounted 3d depth camera. In *International Conference on Computer Vision Theory and Applications*, volume 2, pages 640–647.

[130] Klaser, A., Marszałek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, pages 275–1.

[131] Koppula, H. S., Anand, A., Joachims, T., and Saxena, A. (2011). Semantic labeling of 3d point clouds for indoor scenes. In *Advances in Neural Information Processing Systems*, pages 244–252.

[132] Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970.

[133] Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871.

[134] Krause, A., Perona, P., and Gomes, R. G. (2010). Discriminative clustering by regularized information maximization. In *Advances in Neural Information Processing Systems*, pages 775–783.

[135] Krizhevsky, A., Hinton, G. E., et al. (2010). Factored 3-way restricted boltzmann machines for modeling natural images. In *International Conference on Artificial Intelligence and Statistics*, pages 621–628.

[136] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

[137] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114.

[138] Kumatani, K., Arakawa, T., Yamamoto, K., McDonough, J., Raj, B., Singh, R., and Tashev, I. (2012). Microphone array processing for distant speech recognition: Towards real-world deployment. *Asia Pacific Signal and Information Processing Association Conference*, pages 1–10.

[139] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g 2 o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613.

[140] Kurakin, A., Zhang, Z., and Liu, Z. (2012). A real time system for dynamic hand gesture recognition with a depth sensor. In *European Signal Processing Conference (EUSIPCO)*, pages 1975–1979.

[141] Kwitt, R., Vasconcelos, N., and Rasiwasia, N. (2012). Scene recognition on the semantic manifold. In *European Conference on Computer Vision*, pages 359–372.

[142] Kwolek, B. and Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, 117(3):489–501.

[143] Lai, K., B, L., Ren, X., and Fox, D. (2013). Rgb-d object recognition: Features, algorithms, and a large scale benchmark. In *Consumer Depth Cameras for Computer Vision*, pages 167–192.

[144] Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *International Conference on Robotics and Automation*, pages 1817–1824.

[145] Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, pages 473–480.

[146] Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113.

[147] Le, Q. V., Karpenko, A., Ngiam, J., and Ng, A. Y. (2011). Ica with reconstruction cost for efficient overcomplete feature learning. In *Advances in Neural Information Processing Systems*, pages 1017–1025.

[148] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[149] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[150] Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562.

[151] Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562.

[152] Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104.

[153] Lee, T.-k., Lim, S., Lee, S., An, S., and Oh, S.-y. (2012). Indoor mapping using planes extracted from noisy rgb-d sensors. In *International Conference on Intelligent Robots and Systems*, pages 1727–1733.

[154] Leibe, B., Cornelis, N., Cornelis, K., and Van Gool, L. (2007). Dynamic 3d scene analysis from a moving vehicle. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[155] Lenz, I., Lee, H., and Saxena, A. (2015). Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724.

[156] Leyva, E., Gonzalez, A., and Perez, R. (2015). A set of complexity measures designed for applying meta-learning to instance selection. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):354–367.

[157] Li, S. Z., Hou, X., Zhang, H., and Cheng, Q. (2001). Learning spatially localized, parts-based representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1.

[158] Liang, H., Yuan, J., and Thalmann, D. (2012). 3d fingertip and palm tracking in depth image sequences. In *ACM International Conference on Multimedia*, pages 785–788.

[159] Liao, Y., Kodagoda, S., Wang, Y., Shi, L., and Liu, Y. (2016). Understand scene categories by objects: A semantic regularized scene classifier using convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 2318–2325.

[160] Lin, L., Wang, K., Zuo, W., Wang, M., Luo, J., and Zhang, L. (2015). A deep structured model with radius–margin bound for 3d human activity recognition. *International Journal of Computer Vision*, pages 1–18.

[161] Lin, L., Wang, K., Zuo, W., Wang, M., Luo, J., and Zhang, L. (2016). A deep structured model with radius-margin bound for 3d human activity recognition. *International Journal of Computer Vision*, 118(2):256–273.

[162] Liu, K., Chen, C., Jafari, R., and Kehtarnavaz, N. (2014a). Fusion of inertial and depth sensor data for robust hand gesture recognition. *Sensors Journal*, 14(6):1898–1903.

[163] Liu, L. and Shao, L. (2013). Learning discriminative representations from rgb-d video data. In *International joint conference on Artificial Intelligence*, pages 1493–1500.

[164] Liu, L., Wang, L., and Liu, X. (2011). In defense of soft-assignment coding. In *IEEE International Conference on Computer Vision*, pages 2486–2493.

[165] Liu, M., Salzmann, M., and He, X. (2014b). Discrete-continuous depth estimation from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723.

[166] Liu, W., He, J., and Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *International Conference on Machine Learning*, pages 679–686.

[167] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[168] Luo, J., Wang, W., and Qi, H. (2013). Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. pages 1809–1816.

[169] Luo, Y., Wen, Y., Tao, D., Gui, J., and Xu, C. (2016). Large margin multi-modal multi-task feature extraction for image classification. *IEEE Transactions on Image Processing*, 25(1):414–427.

[170] MacKay, D. J. (2003). *Information theory, inference and learning algorithms*.

[171] Mansilla, E. B. and Ho, T. K. (2004). On classifier domains of competence. In *International Conference on Pattern Recognition*, volume 1, pages 136–139.

[172] Mantecon, T., del Bianco, C. R., Jaureguizar, F., and Garcia, N. (2014). Depth-based face recognition using local quantized patterns adapted for range data. In *International Conference on Image Processing*, pages 293–297.

[173] Marchi, E., Ferroni, G., Eyben, F., Gabrielli, L., Squartini, S., and Schuller, B. (2014). Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2164–2168.

[174] Margolis, A. (2011). A literature review of domain adaptation with unlabeled data. *Tec. Report*, pages 1–42.

[175] Masaeli, M., Dy, J. G., and Fung, G. M. (2010). From transformation-based dimensionality reduction to feature selection. In *International Conference on Machine Learning*, pages 751–758.

[176] Mason, J., Marthi, B., and Parr, R. (2012). Object disappearance for object discovery. In *International Conference on Intelligent Robots and Systems*, pages 2836–2843.

[177] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767.

[178] Meister, S., Izadi, S., Kohli, P., Hämmerle, M., Rother, C., and Kondermann, D. (2012). When can we use kinectfusion for ground truth acquisition? In *Workshop on Color-Depth Camera Fusion in Robotics*.

[179] Mian, A. S., Bennamoun, M., and Owens, R. A. (2006). Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1584–1601.

[180] Min, R., Kose, N., and Dugelay, J.-L. (2014). Kinectfacedb: A kinect database for face recognition. *IEEE Transactions on Cybernetics*, 44(11):1534–1548.

[181] Mockus, J., Tiesis, V., and Zilinskas, A. (1978). The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2.

[182] Mohamed, A.-r. and Hinton, G. (2010). Phone recognition using restricted boltzmann machines. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 4354–4357.

[183] Mollineda, R. A., Sánchez, J. S., and Sotoca, J. M. (2005). Data characterization for effective prototype selection. In *Pattern Recognition and Image Analysis*, pages 27–34.

[184] Monay, F. and Gatica-Perez, D. (2003). On image auto-annotation with latent space models. In *International Conference on Multimedia*, pages 275–278.

[185] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760.

[186] Negin, F., Özdemir, F., Akgül, C. B., Yüksel, K. A., and Erçil, A. (2013). A decision forest based feature selection framework for action recognition from rgb-depth cameras. In *Image Analysis and Recognition*, pages 648–657.

[187] Ngiam, J., Chen, Z., Koh, P. W., and Ng, A. Y. (2011). Learning deep energy models. In *International Conference on Machine Learning*, pages 1105–1112.

[188] Ni, B., Wang, G., and Moulin, P. (2013). Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *Consumer Depth Cameras for Computer Vision*, pages 193–208.

[189] Nie, F., Huang, H., Cai, X., and Ding, C. H. (2010). Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In *Neural Information Processing Systems*, pages 1813–1821.

[190] Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2011). Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference*, pages 1–11.

[191] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175.

[192] Oreifej, O. and Liu, Z. (2013). Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723.

[193] Ouyang, W., Luo, P., Zeng, X., Qiu, S., Tian, Y., Li, H., Yang, S., Wang, Z., Xiong, Y., Qian, C., Zhu, Z., Wang, R., Loy, C. C., Wang, X., and Tang, X. (2014). Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *IEEE Conference on Computer Vision and Pattern Recognition*.

[194] Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

[195] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[196] Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol*, 5(11):e1000579.

[197] Pomerleau, F., Magnenat, S., Colas, F., Liu, M., and Siegwart, R. (2011). Tracking a depth camera: Parameter exploration for fast icp. In *International Conference on Intelligent Robots and Systems*, pages 3824–3829.

[198] Poultney, C., Chopra, S., Cun, Y. L., et al. (2006). Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144.

[199] Raina, R., Madhavan, A., and Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. In *International Conference on Machine Learning*, pages 873–880.

[200] Ranzato, M., Poultney, C. S., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144.

[201] Redko, I. and Bennani, Y. (2016). Non-negative embedding for fully unsupervised domain adaptation. *Pattern Recognition Letters*, 77:35–41.

[202] Richtsfeld, A., Morwald, T., Prankl, J., Zillich, M., and Vincze, M. (2012). Segmentation of unknown objects in indoor environments. In *International Conference on Intelligent Robots and Systems*, pages 4791–4796.

[203] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, pages 833–840.

[204] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, pages 1–4.

[205] Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

[206] Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455.

[207] Sarikaya, R., Hinton, G. E., and Deoras, A. (2014). Application of deep belief networks for natural language understanding. *Transactions on Audio, Speech, and Language Processing*, 22(4):778–784.

[208] Schindler, K. and Van Gool, L. (2008). Action snippets: How many frames does human action recognition require? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[209] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[210] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

[211] Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

[212] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[213] Shantia, A., Timmers, R., Schomaker, L., and Wiering, M. (2015). Indoor localization by denoising autoencoders and semi-supervised learning in 3d simulated environment. In *International Joint Conference on Neural Networks*, pages 1–7.

[214] Shao, L., Zhen, X., Tao, D., and Li, X. (2014). Spatio-temporal laplacian pyramid coding for action recognition. *IEEE Transactions on Cybernetics*, 44(6):817–827.

[215] Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., and Guo, B. (2012). An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics*, 31(6):136.

[216] Shao, Y., Zhou, Y., He, X., Cai, D., and Bao, H. (2009). Semi-supervised topic modeling for image annotation. In *International Conference on Multimedia*, pages 521–524.

[217] Sharmanska, V., Quadrianto, N., and Lampert, C. H. (2013). Learning to rank using privileged information. In *IEEE International Conference on Computer Vision*, pages 825–832.

[218] Shen, B. and Si, L. (2010). Non-negative matrix factorization clustering on multiple manifolds. In *AAAI Conference on Artificial Intelligence*.

[219] Shen, W., Deng, K., Bai, X., Leyvand, T., Guo, B., and Tu, Z. (2012). Exemplar-based human action pose correction and tagging. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1784–1791.

[220] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). Learning semantic representations using convolutional neural networks for web search. In *International Conference on World Wide Web*, pages 373–374.

[221] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *Conference on Computer Vision and Pattern Recognition*, pages 2930–2937.

[222] Silberman, N. and Fergus, R. (2011a). Indoor scene segmentation using a structured light sensor. In *International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, pages 601–608.

[223] Silberman, N. and Fergus, R. (2011b). Indoor scene segmentation using a structured light sensor. In *International Conference on Computer Vision Workshops*, pages 601–608.

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

[225] Singh, A., Sha, J., Narayan, K. S., Achim, T., and Abbeel, P. (2014). Bigbird: A large-scale 3d database of object instances. In *International Conference on Robotics and Automation*, pages 509–516.

[226] Siva, P. and Xiang, T. (2011). Weakly supervised object detector learning with model drift detection. In *International Conference on Computer Vision*, pages 343–350.

[227] Smith, F. W. (1968). Pattern classifier design by linear programming. *IEEE Transactions on Computers*, 100(4):367–372.

[228] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

[229] Socher, R., Huval, B., Bath, B., Manning, C. D., and Ng, A. Y. (2012). Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673.

[230] Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *International Conference on Machine Learning*, pages 129–136.

[231] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576.

[232] Song, S. and Xiao, J. (2013). Tracking revisited using rgbd camera: Unified benchmark and baselines. In *IEEE International Conference on Computer Vision*, pages 233–240.

[233] Song, S. and Xiao, J. (2016a). Deep sliding shapes for amodal 3d object detection in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[234] Song, S. and Xiao, J. (2016b). Deep sliding shapes for amodal 3d object detection in rgb-d images.

[235] Spinello, L. and Arras, K. O. (2011). People detection in rgb-d data. In *International Conference on Intelligent Robots and Systems*, pages 3838–3843.

[236] Spinello, L., Stachniss, C., and Burgard, W. (2012). Scene in the loop: Towards adaptation-by-tracking in rgb-d data. In *Proc. Workshop RGB-D, Adv. Reason. Depth Cameras*.

[237] Srinivasan, A. and Ramakrishnan, G. (2011). Parameter screening and optimisation for ilp using designed experiments. *Journal of Machine Learning Research*, 12:627–662.

[238] Stein, S. and McKenna, S. J. (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *International joint conference on Pervasive and ubiquitous computing*, pages 729–738.

[239] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *International Conference on Intelligent Robot Systems*, pages 573–580.

[240] Sung, J., Ponce, C., Selman, B., and Saxena, A. (2011). Human activity detection from rgbd images. *plan, activity, and intent recognition*, 64.

[241] Susanto, W., Rohrbach, M., and Schiele, B. (2012). 3d object detection with multiple kinects. In *European Conference on Computer Vision Workshops and Demonstrations*, pages 93–102.

[242] Sutskever, I. and Hinton, G. E. (2007). Learning multilevel distributed representations for high-dimensional sequences. In *Artificial Intelligence and Statistics*, volume 2, pages 548–555.

[243] Sutton, M. A., Orteu, J. J., and Schreier, H. (2009). *Image correlation for shape, motion and deformation measurements: basic concepts, theory and applications*.

[244] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[245] Tang, K., Paluri, M., Fei-Fei, L., Fergus, R., and Bourdev, L. (2015). Improving image classification with location context. In *IEEE International Conference on Computer Vision*, pages 1008–1016.

[246] Tang, S., Wang, X., Lv, X., Han, T. X., Keller, J., He, Z., Skubic, M., and Lao, S. (2013). Histogram of oriented normal vectors for object recognition with a depth sensor. In *Asian Conference on Computer Vision*, pages 525–538.

[247] Tao, D., Cheng, J., Lin, X., and Yu, J. (2015). Local structure preserving discriminative projections for rgb-d sensor-based scene classification. *Information Sciences*.

[248] Tao, D., Jin, L., Yang, Z., and Li, X. (2013). Rank preserving sparse learning for kinect based scene classification. *IEEE Transactions on Cybernetics*, 43(5):1406–1417.

[249] Taylor, G. W., Hinton, G. E., and Roweis, S. T. (2006). Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, pages 1345–1352.

[250] Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. (2012). The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110.

[251] Trabelsi, C., Bilaniuk, O., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., and Pal, C. J. (2017). Deep complex networks.

[252] Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280.

[253] 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.

[254] Vanhoucke, V., Senior, A., and Mao, M. Z. (2011). Improving the speed of neural networks on cpus. In *NIPS Workshop*, volume 1, page 4.

[255] Vapnik, V. and Vashist, A. (2009). A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557.

[256] Vaufreydaz, D. and Nègre, A. (2014). Mobilergbd, an open benchmark corpus for mobile rgb-d related algorithms. In *International Conference on Control, Automation, Robotics and Vision*.

[257] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine learning*, pages 1096–1103.

[258] Vishwakarma, S. and Agrawal, A. (2013). A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29(10):983–1009.

[259] Vogel, J. and Schiele, B. (2007). Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision*, 72(2):133–157.

[260] Wallach, I., Dzamba, M., and Heifets, A. (2015). Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*.

[261] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066.

[262] Wang, A., Cai, J., Lu, J., and Cham, T. (2016a). Modality and component aware feature fusion for RGB-D scene classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5995–6004.

[263] Wang, A., Cai, J., Lu, J., and Cham, T.-J. (2016b). Modality and component aware feature fusion for rgb-d scene classification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[264] Wang, H., Shi, X., and Yeung, D.-Y. (2015). Relational stacked denoising autoencoder for tag recommendation. In *AAAI Conference on Artificial Intelligence*, pages 3052–3058.

[265] Wang, J., Liu, Z., Wu, Y., and Yuan, J. (2012). Mining actionlet ensemble for action recognition with depth cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297.

[266] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3360–3367.

[267] Wang, X., Hou, Z.-G., Tan, M., Wang, Y., and Wang, X. (2008). Corridor-scene classification for mobile robot using spiking neurons. In *International Conference on Natural Computation*, volume 4, pages 125–129.

[268] Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013). Regionlets for generic object detection. In *IEEE International Conference on Computer Vision*, pages 17–24.

[269] Wohlkinger, W., Aldoma, A., Rusu, R. B., and Vincze, M. (2012). 3dnet: Large-scale object class recognition from cad models. In *International Conference on Robotics and Automation*, pages 5384–5391.

[270] Wollmer, M., Blaschke, C., Schindl, T., Schuller, B., Farber, B., Mayer, S., and Trefflich, B. (2011). Online driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):574–582.

[271] Wu, D., Pigou, L., Kindermans, P., Le, N. D., Shao, L., Dambre, J., and Odobez, J. (2016). Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 38(8):1583–1597.

[272] Wu, D. and Shao, L. (2014). Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–731.

[273] Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *IEEE International Conference on Robotics and Automation workshop*, volume 2.

[274] Xiao, J., Owens, A., and Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In *International Conference on Computer Vision*, pages 1625–1632.

[275] Xing, C., Ma, L., and Yang, X. (2015). Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *Journal of Sensors*, 2016.

[276] Xu, B., Lu, J., and Huang, G. (2003). A constrained non-negative matrix factorization in information retrieval. In *IEEE International Conference on Information Reuse and Integration*, pages 273–277.

[277] Xu, Y., Mo, T., Feng, Q., Zhong, P., Lai, M., Chang, E. I., et al. (2014). Deep learning of feature representation with multiple instance learning for medical image analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1626–1630.

[278] Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1794–1801.

[279] Yang, S. and Rahimi, A. (2011). Structure learning for optimization. In *Neural Information Processing Systems*, pages 1044–1052.

[280] Yang, W., Jin, L., Tao, D., Xie, Z., and Feng, Z. (2016). Dropsample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten chinese character recognition. *Pattern Recognition*, 58:190–203.

[281] Yang, Y., Shen, H. T., Ma, Z., Huang, Z., and Zhou, X. (2011). $\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In *International Joint Conference on Artificial Intelligence*, volume 22, page 1589.

[282] Yao, G. and Deng, C. (2012). Accelerating locality preserving nonnegative matrix factorization. In *ACM International Conference on Information and Knowledge Management*, pages 2271–2274.

[283] Yoo, D., Park, S., Lee, J.-Y., and Kweon, I. S. (2014). Fisher kernel for deep neural activations. *arXiv preprint arXiv:1412.1628*.

[284] Yu, G., Liu, Z., and Yuan, J. (2015). Discriminative orderlet mining for real-time recognition of human-object interaction. In *Asian Conference on Computer Vision*, pages 50–65.

[285] Yu, M., Liu, L., and Shao, L. (2016). Structure-preserving binary representations for rgb-d action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1651–1664.

[286] Zaki, H. F. M., Shafait, F., and Mian, A. S. (2016). Convolutional hypercube pyramid for accurate RGB-D object category and instance recognition. In *IEEE International Conference on Robotics and Automation*.

[287] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference Computer Vision*, pages 818–833.

[288] Zhang, C., Li, Z., Cai, R., Chao, H., and Rui, Y. (2016). Joint multiview segmentation and localization of rgb-d images using depth-induced silhouette consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[289] Zhang, Q., Song, X., Shao, X., Shibasaki, R., and Zhao, H. (2013). Category modeling from just a single labeling: Use depth information to guide the learning of 2d models. In *Conference on Computer Vision and Pattern Recognition*, pages 193–200.

[290] Zhang, X., Yu, F. X., Guo, R., Kumar, S., Wang, S., and Chang, S.-F. (2015). Fast orthogonal projection based on kronecker product. In *IEEE International Conference on Computer Vision*, pages 2929–2937.

[291] Zheng, A. X. and Bilenko, M. (2013). Lazy paired hyper-parameter tuning. In *International Joint Conference on Artificial Intelligence*, pages 1924–1931.

[292] Zheng, J., Feng, Z., Xu, C., Hu, J., and Ge, W. (2016). Fusing shape and spatio-temporal features for depth-based dynamic hand gesture recognition. *Multimedia Tools and Applications*, pages 1–20.

[293] Zheng, W., Qian, Y., and Tang, H. (2011). Dimensionality reduction with category information fusion and non-negative matrix factorization for text categorization. In *International Conference on Artificial Intelligence and Computational Intelligence*, pages 505–512.

[294] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Neural Information Processing Systems*, pages 487–495.

[295] Zhou, Q.-Y. and Koltun, V. (2013). Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, 32(4):112–117.

[296] Zhu, H., Weibel, J.-B., and Lu, S. (2016). Discriminative multi-modal feature fusion for rgbd indoor scene recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[297] Zhuang, Y., Jiang, N., Hu, H., and Yan, F. (2013). 3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments. *IEEE Transactions on Instrumentation and Measurement*, 62(2):438–450.

[298] Zhuo, W., Salzmann, M., He, X., and Liu, M. (2015). Indoor scene structure analysis for single image depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 614–622.

[299] Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405.

Appendix A

Abbreviations

PCA: Principal Component Analysis
DBN: Deep Belief Networks
SDAE: Stacked Denoising Auto-Encoders
CNN: Convolutional Neural Network
SLAM: Simultaneous Localization and Mapping
SIFT: Scale-invariant Feature Transform
HOG: Histogram of Oriented Gradient
SURF: Speeded-Up Robust Features
NUI: Natural User Interface
PCL: Point Cloud Library
RBM: Restricted Boltzmann Machines
MST: Minimum Spanning Tree
SVM: Support Vector Machine
NMF: Non-negative Matrix Factorization
BP: Backpropagation
RNN: Recursive Neural Networks
NN: Neural Networks
VLAD: Vector of Locally Aggregated Descriptors
UDA: Unsupervised Domain Adaptation
RKHS: Reproducing Kernel Hilbert Space
MMD: Maximum Mean Discrepancy
LSTM: Long Short-Term Memory
SPM: Spatial Pyramid Matching