Article:

Cai, Z. (2017) Performance evaluation of deep feature learning for RGB-D image/video classification. Information Sciences, 385. pp. 266-283. ISSN 0020-0255

https://doi.org/10.1016/j.ins.2017.01.013

Performance Evaluation of Deep Feature Learning for RGB-D Image/Video Classification

Ling Shao[a,b,*], Ziyun Cai[c], Li Liu[d], Ke Lu[e,f]

[a] *College of Electronic and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China.*
[b] *School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K.*
[c] *Department of Electronic and Electrical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, U.K.*
[d] *Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K.*
[e] *University of Chinese Academy of Sciences, Beijing 100049, China.*
[f] *Beijing Center for Mathematics and Information Interdisciplinary Sciences, Beijing, China.*

Abstract

Deep Neural Networks for image/video classification have obtained much success in various computer vision applications. Existing deep learning algorithms are widely used on RGB image or video data. Meanwhile, with the development of **low-cost** RGB-D sensors (such as Microsoft Kinect and Xtion Pro Live), high-quality RGB-D data can be easily acquired and used to enhance computer vision algorithms [29]. It would be interesting to investigate how deep learning can be employed for extracting and fusing features from RGB-D data. In this paper, after briefly reviewing the basic concepts of RGB-D information and 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), we conduct extensive experiments on five popular RGB-D datasets including three image datasets and two video datasets. We then present a detailed analysis about 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 paper. According to the extensive experimental results, we

*Corresponding author. Tel.: +44 (0)114 222 5841; E-mail: ling.shao@ieee.org

believe that this evaluation will provide insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion.

Keywords: Deep neural networks, RGB-D data, Feature learning, Performance evaluation.

1. Introduction

Learning good feature representations from input data for high-level tasks receives much attention in computer vision, robotics and medical imaging [52, 53, 93, 97]. Image/video classification is a classic and challenging high-level task, which has many practical applications, such as robotic vision [1], image annotation [63, 71] and video surveillance [41, 85]. 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 [16]. It has been proved that combining RGB and depth information in image/video classification tasks can significantly improve the classification accuracy [29, 36, 43]. Therefore, an increasing number of RGB-D datasets have been created as benchmarks [13]. Moreover, Deep Neural Networks for high-level tasks obtain great success in recent years. Different from hand-crafted feature representations such as SIFT [60], HOG [17] and STLPC [70], 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 [87]. 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 evaluation will provide insights and a deeper understanding of different deep learning algorithms for RGB-D feature extraction and fusion.

1.1. 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. This drawback significantly impedes the usage of RGB based vision algorithms in real-world 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. Compared to early expensive and inconvenient range sensors (such as Konica Minolta Vivid 910), the **low-cost** 3D Microsoft Kinect sensor makes the acquisition of RGB-D data cheaper and easier. Therefore, 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. [9] 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. [81] 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 [8], 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 [77] and [78]. In [18], Endres et al. introduced an approach which describes a volumetric voxel representation [95] through optimizing the 3D pose graph using the g^2o [46] framework which can be directly used for path planning, robot localization and navigation [35]. More papers on combining color and depth channels from multiple scenes using RGB-D perception can be found in [83], [72], [55].

1.2. Related Work to Deep Learning Methods

According to our evaluation, we select four representative deep learning methods including Deep Belief Networks (DBNs), Stacked Denoising Auto-Encoders (SDAE), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) Neural Networks for our experiments. 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 [33]. Meanwhile, it achieved good results on phoneme recognition, with an error rate of 26.7% on the TIMIT core test set [62]. SDAE was first introduced in [84] as an extension of Stacked auto-encoder (SAE) [48]. BP-trained CNNs [50] achieved a new MNIST record of 0.39% [64]. In 2012, GPU-implemented CNNs achieved the best results on the ImageNet classification benchmark [45]. LSTM won the ICDAR handwriting competition in 2009 and achieved a record 17.7% phoneme error rate on the TIMIT natural speech dataset in 2013. More relevant work and history on these four deep learning methods can be found in [68].

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) [76] to obtain higher order features. In CNN-RNN, C-NN 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) [3] 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 the Convolution Neural Networks (CNNs) to extract the machine learned representation and Locality-constrained Linear Coding (LLC) based spatial pyramid matching for hand-crafted features [40]. This new feature representation method can obtain both the advantages of hand-crafted features and machine learned 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.

1.3. Deep learning methods for RGB-D Data Analysis

Since deep learning methods have shown to be useful for standard RGB vision tasks like object detection, image classification and semantic segmentation, more works on RGB-D perception naturally consider neural networks for learning representations from depth information [15] [76]. In general, the RGB-D vision problems that can be addressed or enhanced by means of the deep learning methods are summarized from four aspects: object detection and tracking, object and scene recognition, human activity analysis and indoor 3-D mapping. In this paper, our experiments focus on object and scene recognition, and human activity analysis.

1.3.1. Object Detection and Tracking

The depth information of an object is immune to object appearance changes, environmental illumination and subtle movements of the background. With the invention of the low-cost Kinect depth camera, researchers immediately realized that features based on depth information can significantly improve detecting and tracking objects in the real world where all kinds of variations occur. Depth-RCNN [27] [28] is the first object detector using deep convolutional nets on RGB-D data, which is an extension of the RCNN framework [22]. The depth map is encoded as three extra channels (with Geocentric Encoding: Disparity, Height, and Angle) appended to the color images. Furthermore, Depth-RCNN was extended to generate 3D bounding boxes through aligning 3D CAD models to the recognition results. Tracking via deep learning methods in RGB-D data is also an important topic. In [98], Xue et al. proposed to train a deep convolutional neural network, which improves tracking performance, to classify people in RGB-D videos. RGB-D based object detection and tracking through deep learning methods have attracted great attention in recent few years.

1.3.2. Object and Scene Recognition

The conventional RGB-based deep learned features may suffer from the distortions of an object. RGB information is less capable of handling these environmental variations. Fortunately, the combination of RGB and depth information can potentially enhance the robustness of the deep learned features. Zaki et al. [99] presented an RGB-D object recognition framework which employed a CNN pre-trained on RGB data as feature extractors for both color and depth channels. Then they proposed a rich coarse-to-fine feature representation scheme, called Hypercube Pyramid, which can capture discriminatory information at different levels of detail. Zhu et al. [100] introduced a novel discriminative multi-modal fusion framework for RGB-D scene recognition which simultaneously considered the inter- and intra-modality correlation for all samples and meanwhile regularizing the learned features to be discriminative and compact. Then the results from the multimodal layer can be back-propagated to the lower CNN layers. Many object/scene recognition deep learning methods based on RGB and depth information have been proposed recently [88] [59].

1.3.3. Human Activity Analysis

Apart from outputting both RGB and depth information, another contribution of Kinect is a fast human-skeletal tracking algorithm. This tracking algorithm can provide the exact location of each joint of the human body over time, which makes the representation of complex human activities easier. Wu et al. [92] proposed a novel method called Deep Dynamic Neural Networks (DDNN) for multimodal gesture recognition, which learns high-level spatiotemporal representations using deep neural networks suited to the input modality: a Gaussian-Bernouilli Deep Belief Network (DBN) to handle skeletal dynamics, and a 3D Convolutional Neural Network (3DCNN) to manage and fuse batches of depth and RGB images. Li et al. [54] proposed a feature learning network which is based on sparse auto-encoder (SAE) and principal component analysis for recognizing human actions. Many new deep learning methods are devoting to deducing human activities from depth information or the combination of depth and RGB data [56] [57].

1.3.4. Indoor 3-D Mapping

The emergence of Kinect boosts the research for indoor 3-D mapping through deep learning methods due to its capability of providing depth information directly. Zhang et al. [42] proposed an approach to embed 3D context into the topology of a neural network trained for the performance of holistic scene understanding. After a 3D scene is depicted by a depth image, the network can align the observed scene with a predefined 3D scene template and then reason about the existence and location of each object within the scene template. To recover full 3D shapes from view-based depth images, Wu et al. [94] proposed to represent a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid through a Convolutional Deep Belief Network. Over the last few years, many excellent works about deep learning for indoor 3-D mapping have been published [69] [30].

Aiming to make a comprehensive performance evaluation, we collect five representative datasets including two RGB-D object datasets [12, 47], an

RGB-D scene dataset [74], an RGB-D gesture dataset [58] and an RGB-D activity dataset [90] 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, the pre-trained CNNs model through abundant RGB data and the RGB-D coding methods are not included. It is because that not all of these four deep learning methods can use other RGB data for pre-training and the particular RGB-D coding methods may not be suitable for all of the four 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, which is introduced in detail in Section 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 evaluation.

The rest of this paper is organized as follows. In Section 2, we briefly review the deep learning models which we use for evaluation in our experiments. In Section 3, we present the data pre-processing techniques on deep learned features. Section 4 describes experimental analysis, results and some tricks on our selected RGB-D datasets. Finally, we draw the conclusion in Section 5.

2. Deep Learning Models

In recent years, many successful deep learning methods [10, 32, 49, 84] as efficient feature learning tools have been applied to numerous areas. The aim of deep nets is to learn high-level features at each layer from the features learned at the previous layers. Some methods (such as DBNs [32] and SDAE [84]) 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 minima when doing supervised gradient descent. Therefore, we can consider that unsupervised pre-training is a regularizer. In the fine-tuning step, the 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 [45] 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 convolutions but also can be extended into 3D-CNNs [39].

In this section, we will briefly introduce four deep learning models which are used in our experiments, DBNs, SDAE, CNNs and LSTM.

2.1. Deep Belief Networks

Deep Belief Networks (DBNs) stack many layers of unsupervised Restricted Boltzmann Machines (RBMs) in a greedy manner which was first introduced by Hinton et al. [32]. An RBM consists of visible layers and hidden layers. Each neuron on the layers is fully connected to all the neurons on the next layer. But there are no connections in the same layer. The learned weights are used to initialize a multi-layer neural network and then adjusted to the current task through supervised information for classification. A schematic representation of DBNs can be found in Fig. 1.

In practice, the joint distribution $p(\mathbf{v}, \mathbf{h}; \theta)$ over the visible units $\mathbf{v}$ and hidden units $\mathbf{h}$ can be expressed as:

$$p(\mathbf{v}, \mathbf{h}; \theta) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}; \theta))}{Z}, \tag{1}$$

where the model parameters $\theta = \mathbf{w}, \mathbf{a}, \mathbf{b}$ and $Z = \sum_v \sum_h \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$ is the normalization factor. The energy $E(\mathbf{v}, \mathbf{h}; \theta)$ of the joint configuration $(\mathbf{v},\mathbf{h})$ is defined as:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{V} \sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j, \tag{2}$$

where V and H are the numbers of the visible and hidden units. w_{ij} is the symmetric interaction between visible unit v_i and hidden unit h_j. b_i and a_j are the bias terms.

The marginal probability of the model to a visible vector $\mathbf{v}$ is expressed as:

$$p(\mathbf{v}; \theta) = \frac{\sum_h \exp(-E(\mathbf{v}, \mathbf{h}; \theta))}{Z}. \tag{3}$$

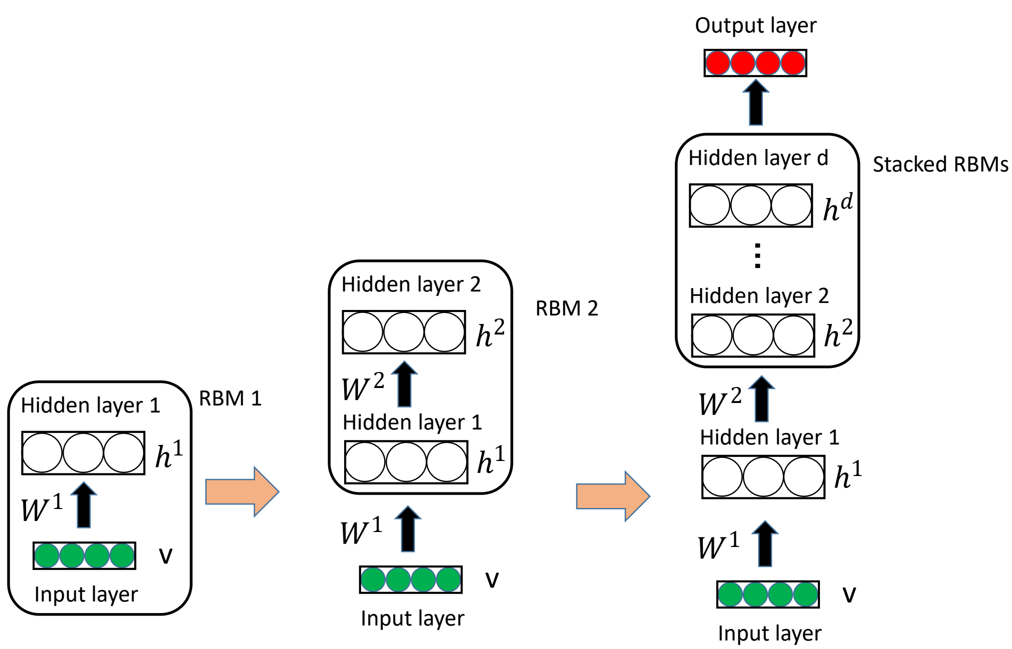


Figure 1: The schematic representation of DBNs. It is just an example of DBNs structure. In practice, the number of units on each hidden layer is flexible.

Therefore, according to the gradient of the joint likelihood function of data and labels, we can get the update rule of the **v**-**h** weights as

$$\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} . \tag{4}$$

The greatest advantage of DBNs is the capability of "learning features" in a "layer-by-layer" manner. The higher-level features are learned from the previous layers. 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 used for generating and recognizing images [5], video sequences [79], motion-capture data [82] and natural language understanding [66].

2.2. Stacked Denoising Auto-Encoders

The Stacked Denoising Auto-Encoders (SDAE) [84] is an extension of the Stacked auto-encoder [48]. This model works in much the same way with DBNs. It also uses the greedy principle but stacks denoising auto-encoders to initialize a deep network. An auto-encoder consists of an encoder $h(\cdot)$ and a decoder $g(\cdot)$. Therefore, the reconstruction of the input x can be expressed as $Re(x) = g(h(x))$. Through minimizing the average reconstruction error $loss(x, Re(x))$, the reconstruction accuracy is able to be improved. This unsupervised pre-training is done on one layer at one time.

Same as DBNs, after all layers have been pre-trained, the parameters which can describe levels of representation about x are used as initialization to the deep neural network optimized with a supervised training criterion. In

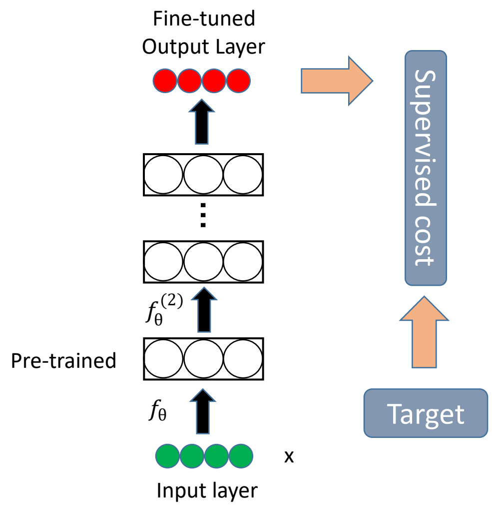


Figure 2: A diagram of Stacked Denoising Auto-Encoders which includes an unsupervised pre-training step and a supervised fine-tuning step. Through performing gradient descent, the parameters are fine-tuned to minimize the error with the supervised target.

the fine-tuning stage, an output logistic regression layer is added to the top of the unsupervised pre-trained machine. Then, the classifier is fine-tuned through the design data set $D_x = \{d_{x_1}, \cdots, d_{x_n}\}$ and the corresponding set of label codes $L_y = \{l_{y_1}, \cdots, l_{y_n}\}$ to minimize the entropy loss function between the correct labels and the classifier's predictions. A schematic diagram of Stacked Denoising Auto-Encoders is shown in Fig. 2.

For binary $\mathbf{x}$, the cross-entropy loss of the input vector $\mathbf{x} \in \{0,1\}^d$ and the reconstructed d-dimensional vector $\hat{\mathbf{x}}$ is expressed as:

$$CEL(\mathbf{x} \| \hat{\mathbf{x}}) = \sum_i CEL(x_i \| \hat{x_i}) = -\sum_i (x_i log \hat{x_i} + (1 - x_i) log(1 - \hat{x_i})), \quad (5)$$

where $\hat{\mathbf{x}} = sigmoid(c + w^T h(c(x)))$, c is the bias, and w is the transpose of the feed-forward weights. Additionally, another option is to use a Gaussian model.

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 instead of RBMs as the unsupervised building block of a deep architecture is that as long as the training criterion is continuous in the parameters, almost any parametrization of the layers is possible [4]. 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 DBNs which is with generative models, samples cannot be drawn to check qualitatively what has been learned.

SDAE is currently applied to many areas such as domain adaptation [23], images classification [96] and text analysis [89].

2.3. Convolutional Neural Networks

Convolutional Neural Networks [51] obtain much success in many visual processing tasks in recent years. This deep learning model is motivated by Hubel and Wiesel's work [37] on the cat's visual cortex. This visual cortex includes some cells which are sensitive to small sub-regions of the visual field. It can be called a receptive field. In practice, these cells can be considered as filters on the input space in the CNNs model. It has been proved that it is well-suited to extract the local correlation in natural images/videos.

Convolutional Neural Network consists of one image processing layer, one or more convolutional layers and fully connected layers and one classification layer. A classical schematic representation of CNNs is shown in Fig. 3. The image processing layer is a designed pre-processing layer which can keep being fixed in the training step. We introduce the pre-processing layer in Section 3 in detail. The convolutional layer applies a set of kernels of size $n \times n \times c$ that are able to process small local parts of the input. For most of the 2D-CNNs experiments, the input color images are often processed into gray images to enhance the efficiency and accuracy, therefore, the kernel size is often expressed as $n \times n$. Pooling is another important concept. It is a form of non-linear down-sampling where each map is sub-sampled with mean or max pooling over $m \times m$ contiguous regions (usually, m is from 2 to 5). It can improve translation invariance and tolerance to small differences of positions about object parts, at the same time, lead to faster convergence. The classification layer is fully connected which combines the outputs from the topmost convolutional layer into a feature vector, with one output unit per class label. Additionally, weight sharing is a significant principle since it is able to reduce the number of trainable parameters. More details concerning

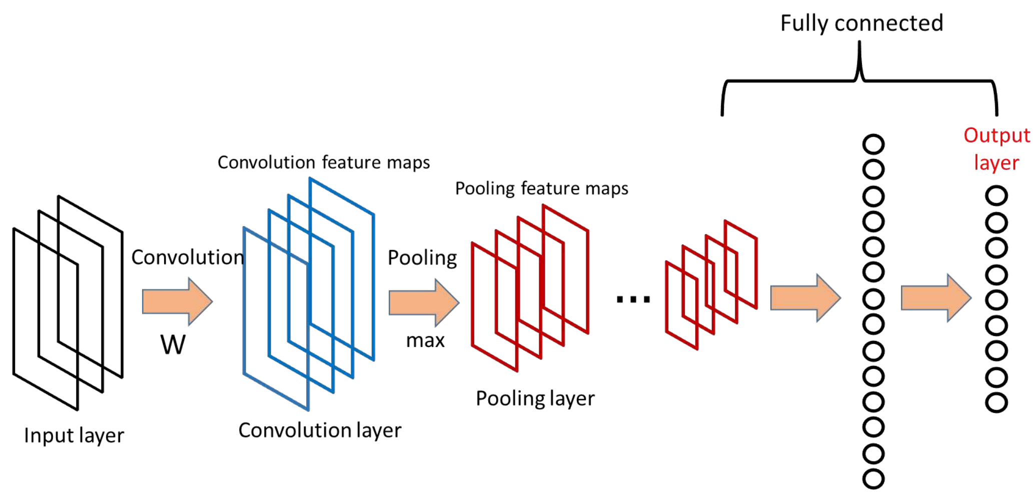


Figure 3: The classical schematic representation of CNNs which includes an input layer, convolutional layers, max-pooling layers and an output layer. The fully connected part is also presented in the figure.

CNNs can be found in [11]. For a multi-label classification problem with F training examples and M classes, the squared-error is expressed as:

$$E^F = \frac{1}{2}\sum_{f=1}^{F}\sum_{m}^{M}(t_m^f - y_m^f)^2, \tag{6}$$

where t_m^f is the value of the m-th dimension about f-th pattern's corresponding label, and y_m^f is the m-th output layer unit related to f-th input pattern. In our experiments, for better results, we use 2D-CNNs for image datasets and 3D-CNNs for video datasets. Due to the space limitation, we do not give a detailed review of 3D-CNNs. More details can be found in [39].

One major advantage of CNNs is the use of shared weights in convolutional layers. The same filter is used for each pixel in the layer, which leads to the reduction of memory footprint and the improvement of result performance. For image classification applications, CNNs use relatively little pre-processing, which means that the network in CNNs is responsible to learn the filters. Without dependence on prior knowledge and human effort for designing features is another major advantage of CNNs. Besides, compared to traditional neural networks, CNN is more robust towards variation of input features. The neurons in the hidden layers are connected to the neurons that are in the same spatial area instead of being connected to all of the nodes in the previous layer. Furthermore, the resolution of the image data is reduced when calculating to higher layers in the network. 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 is considered as one of the most attractive supervised feature learning methods nowadays. CNNs have achieved superior performance for different tasks such as image recognition [80], video analysis [39], Natural language processing [73] and drug discovery [86]. Especially, CNNs based on GoogLeNet increased the mean average precision of object detection to 0.439 and reduced classification error to 0.067 [80]. Both of the performances are the best results up to now.

2.4. Long Short-Term Memory Neural Networks

Long short-term memory (LSTM) is an extension of recurrent neural network (RNN) architecture which was first proposed in [34] 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, taking advantages of multiplicative gates can control the access to the CEC.

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. 4. Fig. 4 shows that the memory block assemblies are composed of multiple separate layers: the input gate layer (ι), the forget gate layer (ϕ), the memory cell layer (c), and the output gate layer (ω). The input layer projects all of the connections to each of these layers. The memory cell layer projects all of the connections to the output layer (θ). Moreover, each memory cell c_j projects a single ungated peephole connection to each of its associated gates. A diagram of

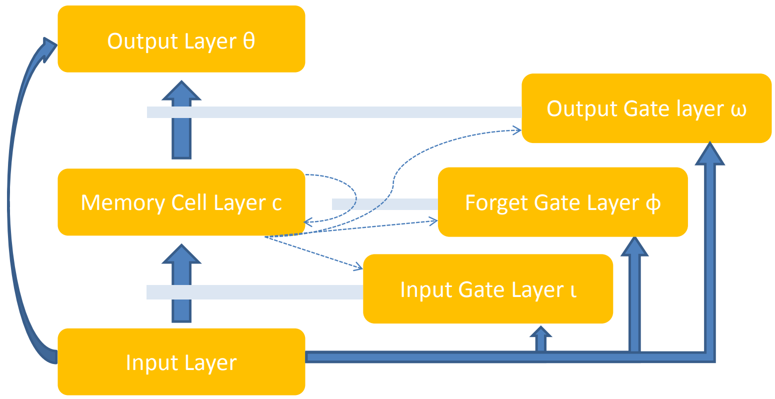


Figure 4: The standard LSTM architecture. The memory block assemblies contain separate layers of memory cells, input gates, forget gates and output gates, in addition to the input layers and output layers. Blue solid arrows show full all-to-all connectivity between units in a layer. Blue dashed arrows mean connectivity only between the units in the two layers that have the same index. The light gray bars denote gating relationships.

a 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. 5. 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. An LSTM network computes a mapping from an input sequence $x = (x_1, \cdots, x_T)$ to an output sequence $y = (y_1, \cdots, y_T)$ through computing the network unit activations through the following equations iteratively from $t = 1$ to T [65]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i), \tag{7}$$
$$f_t = \sigma(W_{fx}x_t + W_{mf}m_{t-1} + W_{cf}c_{t-1} + b_f), \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c), \tag{9}$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o), \tag{10}$$
$$m_t = o_t \odot h(c_t), \tag{11}$$
$$y_t = W_{ym}m_t + b_y, \tag{12}$$

where the W terms denote weight matrices, the b terms denote bias vectors, σ is the logistic sigmoid function, and i, f, c and o represent the input gate, forget gate, cell activation vectors and output gate respectively, all of which are the same size as the cell output activation vector m. $\odot$ is the element-wise

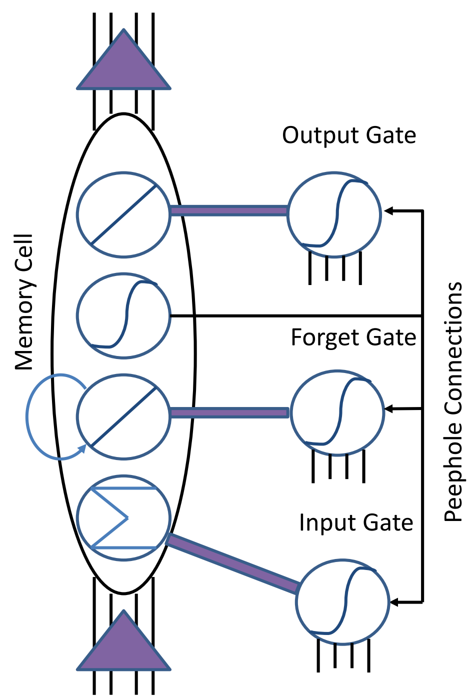


Figure 5: A cross-section of an LSTM network, with a single memory block, and connections from the input layer (bottom) to the output layer (top).

product of the vectors. g and h are the cell input and cell output activation functions, generally $tanh$.

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 [34]. 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 [25], handwriting recognition [26] and human action recognition [2]. Besides, LSTM is also applicable to robot localization [21], online driver distraction detection [91] and many other tasks. Specially, LSTM RNN/HMM hybrids obtained best known performance on medium-vocabulary [24] and large-vocabulary speech recognition. Moreover, LSTM-based methods set benchmark records in audio onset detection [61], prosody contour prediction [20] and text-to-speech synthesis [19]. 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 experiments, we only show the performance about LSTM on a gesture recognition dataset (SKIG dataset) and an action recognition dataset (MSRDailyActivity3D dataset).

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 [14]. 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.

3.1. Normalization

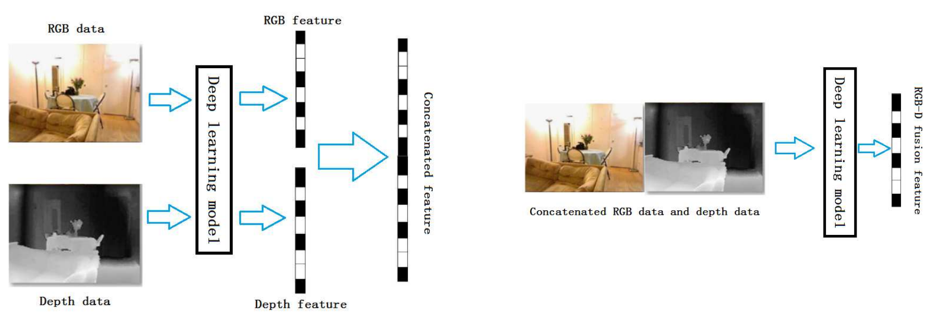
General normalization approaches include simple rescaling, per-example mean subtraction and feature standardization. The choice of these methods mainly depends on the data. In our experiments, since feature standardization is able to set every dimension of raw data to have zero-mean and unit-variance, 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.

3.2. PCA/ZCA Whitening

Following the step of feature standardization, we apply PCA/ZCA whitening to the entire dataset [38]. This is commonly used in deep learning tasks (e.g., [44]). 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 [38].

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 [12], RGB-D object dataset [47], NYU Depth v1 indoor scene segmentation dataset [74], Sheffield Kinect Gesture dataset (SKIG) [58] and MSRDailyActivity3D dataset [90]. 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. 6. 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

Figure 6: Illustration about two experimental procedures used in our evaluation work.

4.1. 2D&3D Object Dataset

We evaluate deep feature learning for object category recognition on the 2D&3D object dataset [12]. This dataset includes 18 different categories (*i.e.*, binders, books and scissors) with each of them containing 3 to 14 objects resulting in 162 objects. The views of each object are recorded every 10 degrees

Figure 7: Example images in the 2D&3D Object dataset, which contains 14 object classes (binder, books, bottles, cans, coffee pots, cups, dishes, dish liquids, mice, pens, scissors, monitors, silverware and drink cartons). 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.

along the vertical axis. Therefore, there are totally $162 \times 36 = 5832$ RGB images and $162 \times 36 = 5832$ depth images respectively. For the consistency with the setup in [12], 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'. Example images from this dataset are given in Fig. 7. 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 1.

Table 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

The hyper-parameters of the DBNs, SDAE and CNNs models are described in Table. 2, Table. 3 and Table. 4. Fig. 8 shows confusion matrixes about our three deep learning models across 14 classes on the 2D&3D dataset.

Table 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

Table 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

From the comparison results of our experiments about three selected deep learning models on 2D&3D dataset in Table. 1, it can be seen that the accuracy of RGB, depth and 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. 8, 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 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

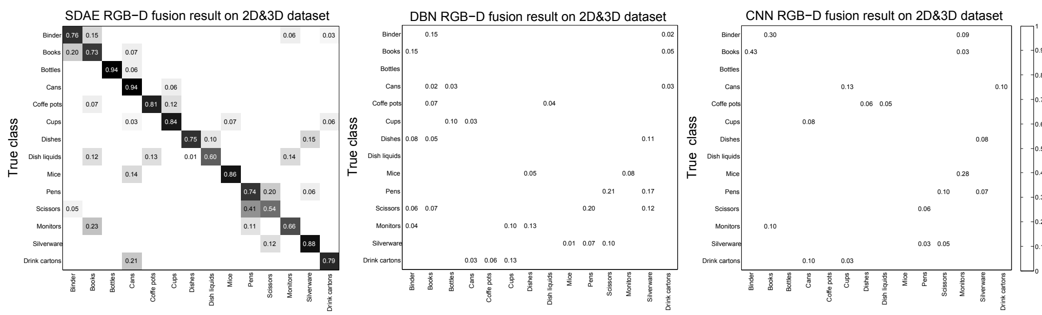


Figure 8: Confusion matrixes about three deep learning models on the 2D&3D dataset. The labels on the vertical axis express the true classes and the labels on the horizontal axis denote the predicted classes.

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.

4.2. Object RGB-D Dataset

We test these deep learning algorithms on the second dataset called RGB-D object dataset. This dataset contains 41877 images which are organized into 51 categories about 300 everyday objects such as apples, mushrooms and notebooks. All of the objects are segmented from the background through combining color and depth cues. Fig. 9 shows some segmentation objects from this dataset. Every shown object is from one of the 51 object categories. Following the setup in [47], we choose to run category recognition experiments

Figure 9: Some example images in Object RGB-D dataset. We can find 20 paired samples shown in this figure. In each pair, the segmented RGB image is shown on the top and the corresponding depth image is on the bottom.

by randomly selecting one object from the categories for testing. Each image in object RGB-D dataset is resized into 56×56 pixels for consistency with the 2D&3D dataset. Table 5 summarizes the comparison between neural-network classifier and SVM.

Table 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**

The hyper-parameters of three deep learning models DBNs, SDAE and CNNs are shown in Table 6, Table 7 and Table 8.

As we can see from Table 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. 10

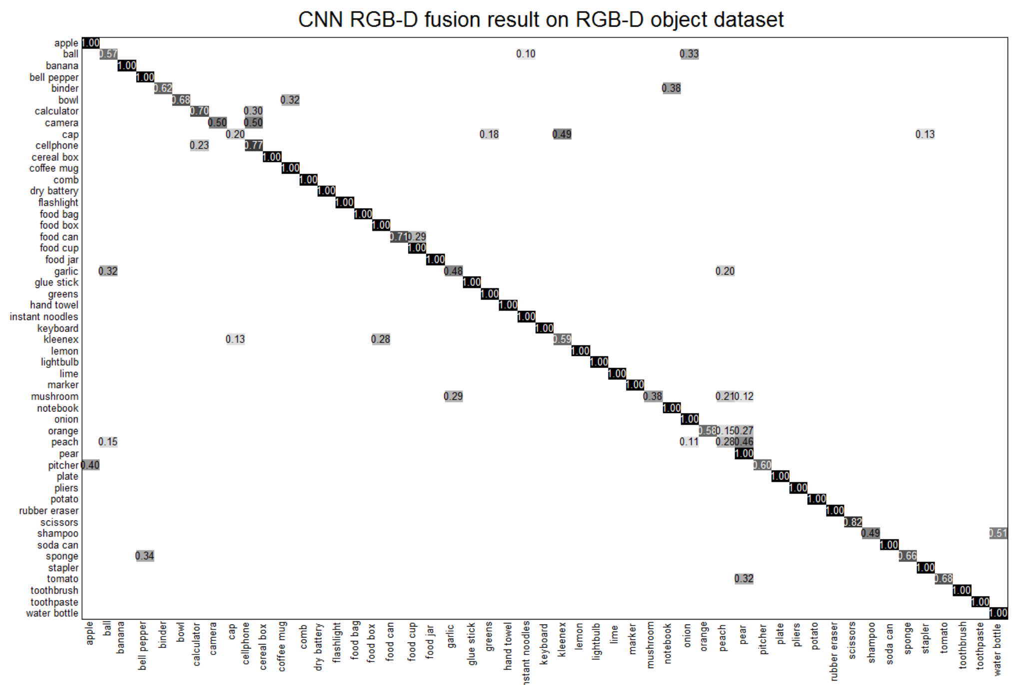


Figure 10: Confusion matrix about CNNs on Object RGB-D Dataset. The labels on the vertical axis express the true classes and the labels on the horizontal axis denote the predicted classes.

Table 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

Table 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

shows the confusion matrix about CNNs across 51 classes over object RGB-D dataset.

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

Table 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

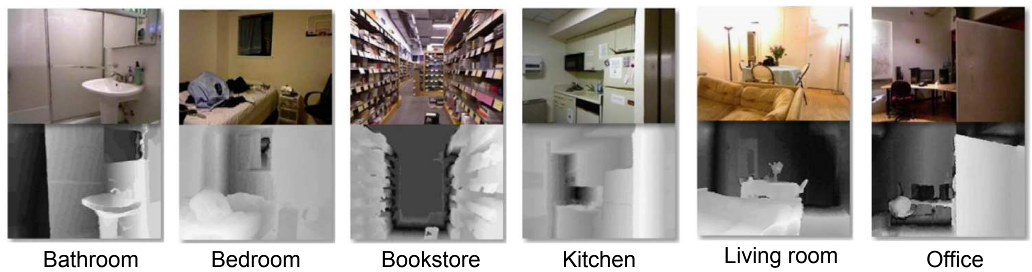


Figure 11: Some example images in the NYU Depth v1 dataset. It includes 6 object classes (bathroom, bedroom, bookstore, kitchen, living room and office). We can find 6 paired samples shown in this figure. In each pair, the segmented RGB image is shown on the top and the corresponding depth image is on the bottom.

the dataset, we use the remaining 6 different scenes. Example images in the NYU Depth v1 dataset are shown in Fig. 11. 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% [74]. Table 9 shows the performance comparison between neural-network classifier and SVM on this dataset.

Table 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 10, Table 11 and Table 12. Fig. 12 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 challeng-

Table 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 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

ing. 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 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

Table 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

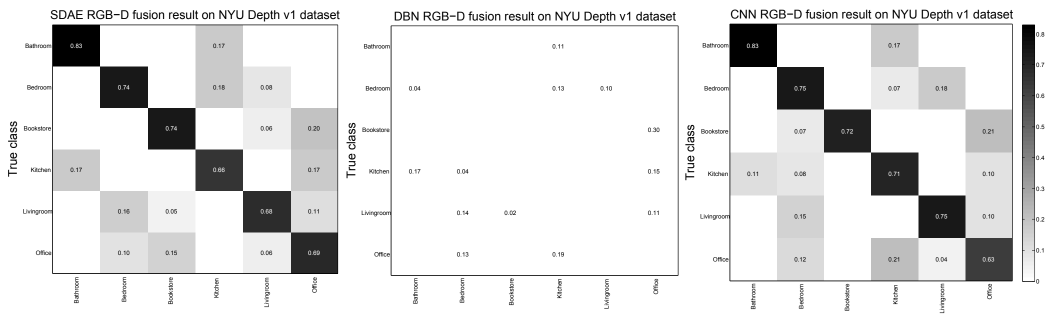


Figure 12: Confusion matrixes about three deep learning models on NYU Depth v1 dataset. The labels on the vertical axis express the true classes and the labels on the horizontal axis denote the predicted classes.

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.

4.4. Sheffield Kinect Gesture (SKIG) Dataset

We also evaluate these four deep 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. Fig. 13 shows some frames in this dataset. In our experiments, since it has been proved that 5$\sim$7 frames (0.3$\sim$0.5 seconds of video) are enough to have the similar performance with the one obtainable with the entire video sequence [67]. Therefore, each video sequence is resized into $64 \times 48 \times 13$. Following the experimental setting in [58], we choose four objects as the training set and test on the remaining data. Table 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. We also compare LSTM Neural Networks experimentally in this subsection.

The hyper-parameters of DBNs, SDAE, 3D-CNNs and LSTM can be found in Table 14, Table 15, Table 16 and Table 17.

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

Fig. 14 shows confusion matrixes about our four deep learning models across 10 classes on the SKIG dataset.

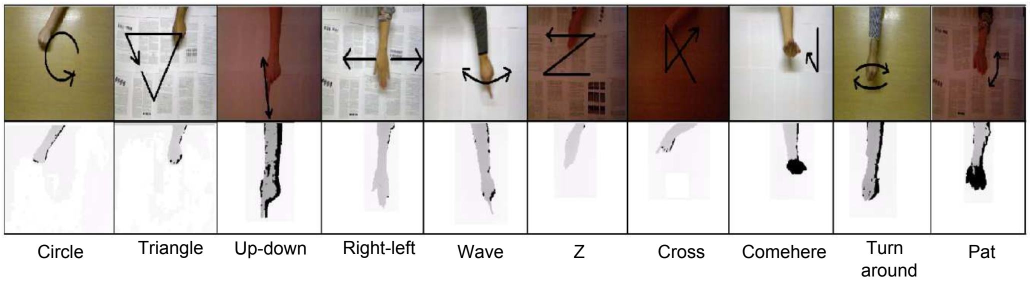
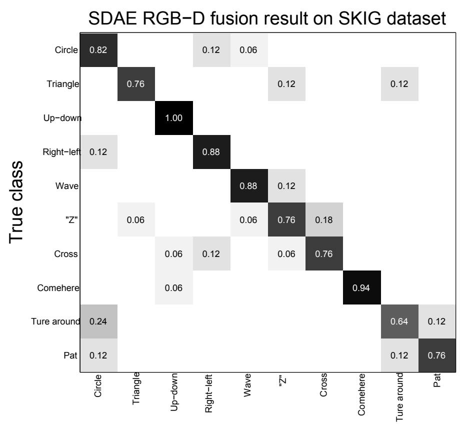


Figure 13: Example frames from Sheffield Kinect gesture dataset and the descriptions of 10 different categories: circle (clockwise), triangle (anti-clockwise), up and down, right and left, wave, hand signal "Z", cross, comehere, turn around and pat. In each pair, the segmented RGB image is shown on the top and the corresponding depth image is on the bottom.

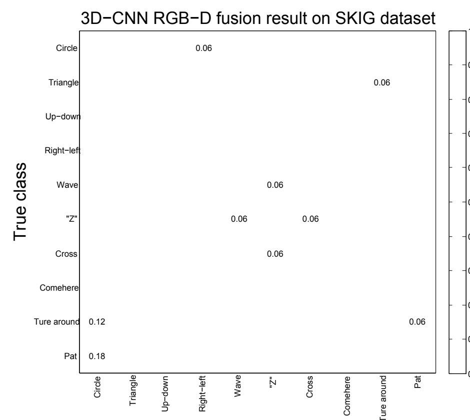
From the comparison of these four deep learning models in Table 13, we can see that 3D-CNNs achieve the best performance among four - 93.3%. It may be because that 3D-CNNs consider the more temporal correlation between video frames [39]. Sequence learning method LSTM with raw pixel features achieves 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.

4.5. MSRDailyActivity3D Dataset

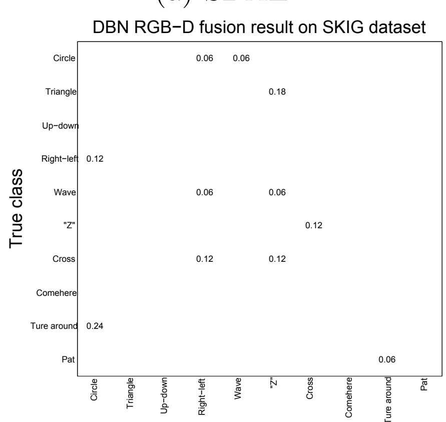
The last dataset which we test on is MSRDailyActivity3D dataset [90]. It is a daily activity dataset which contains 16 activity types (*e.g.*, drink, eat, play game). There are 10 subjects with each of them performs each activity twice, once in standing position, and once in sitting position. Examples of RGB images, raw depth images in this dataset are illustrated in Fig. 15. 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



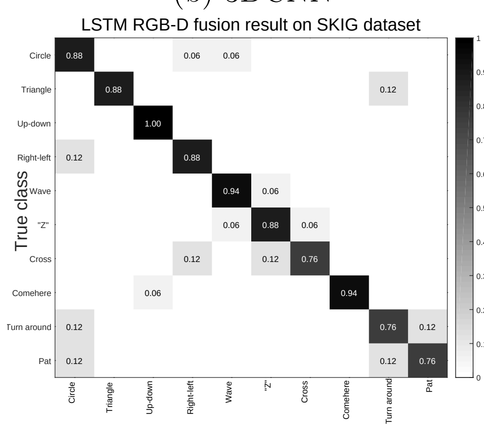
(a) SDAE



(b) 3DCNN



(d) DBN



(e) LSTM

Figure 14: Confusion matrixes about four deep learning models on SKIG 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) SDAE, (b) 3DCNN, (c) DBN, (d) LSTM.

Table 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**

Table 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

are used for evaluation. Table 18 shows the accuracies of four deep learning methods.

The hyper-parameters of DBNs, SDAE, 3D-CNNs and LSTM are shown in Table 19, Table 20, Table 21 and Table 22.

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%), SDAE (66.3%) and LSTM (68.1%). But compared to the performances of SKIG dataset, we only obtain lower accuracies. There are two main reasons. First, it is a very challenging video dataset. According to this dataset, inter-class distance is

Table 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 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

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. 16 shows confusion matrixes about our four deep learning models across 16 classes over MSRDailyActivity3D dataset.

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,

Table 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

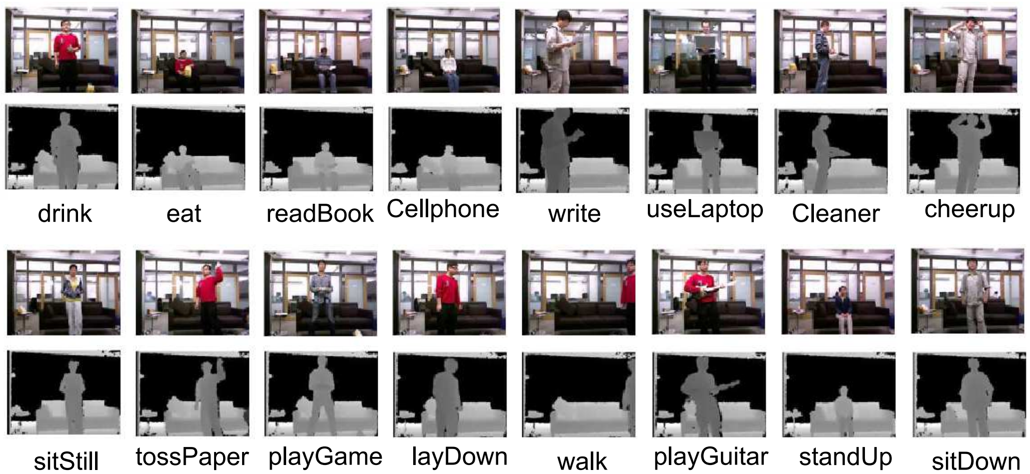
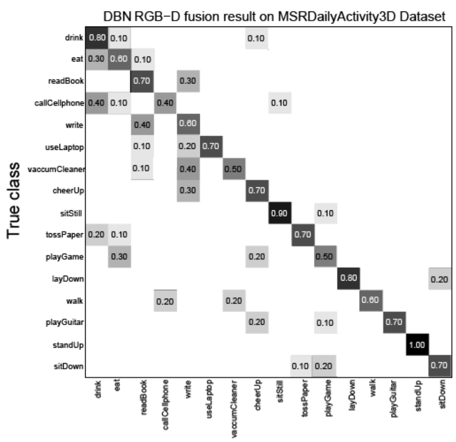


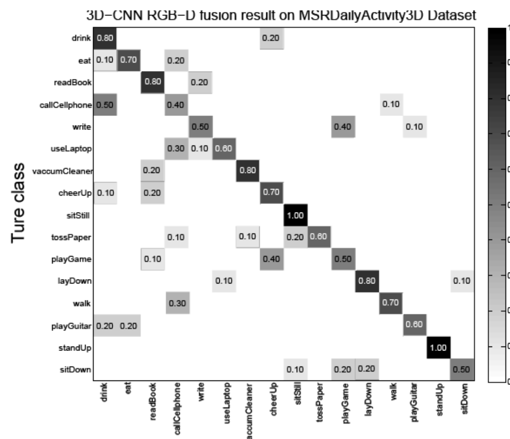
Figure 15: Selected examples of RGB images and raw depth images in MSRDailyActivity3D dataset.

inappropriate parameters may result in overfitting or convergence to a locally optimal solution, so it requires a strong practical experience. 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 [6] 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 [7]. This work is successfully extended in [75] which considers the hyper-parameters optimization problem through the framework of Bayesian optimization. In this paper, we give some tricks about how to choose hyper-parameters in our experiments. It can help other researchers use deep neural networks.

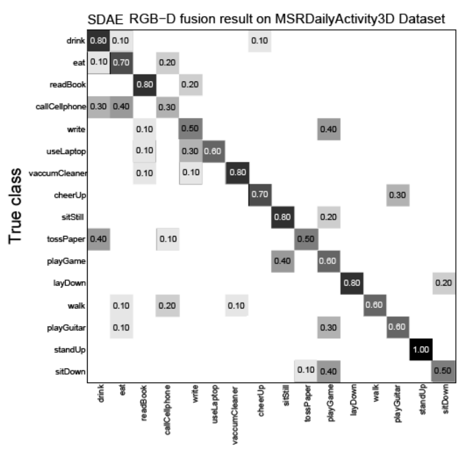
During our experiments, we find that DBNs are more difficult than CNNs and SDAE in hyper-parameter optimization. With inappropriate parameters, DBNs easily converge to locally optimal solutions. According to DBNs, CNNs, SDAE and LSTM, the reconstruction error always increases remarkably if the learning rate is too large. Therefore, we follow the simplest solution and try several small log-spaced values $(10^{-1}, 10^{-2}, \ldots)$ [31]. Then



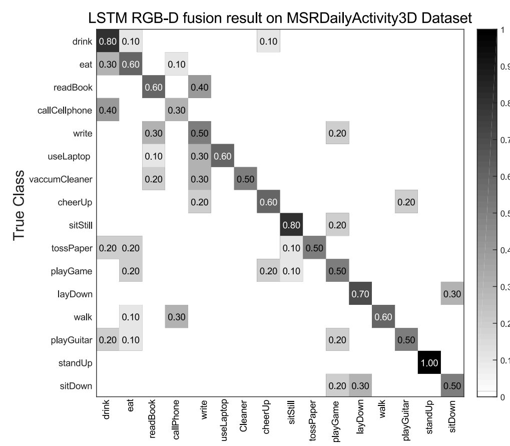
(a) DBN



(b) 3D-CNN



(d) SDAE



(e) LSTM

Figure 16: 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-CNN, (c) SDAE, (d) LSTM.

Table 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**

Table 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

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, while natural image datasets which are with hundreds of pixels in each dimension are better to use larger kernel sizes such as 10×10

Table 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 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

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 $(10, 15, 20, \ldots)$.

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

Table 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

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 1, Table 5 and Table 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 13 reveal that it will also show a great performance without lots of training samples when the experimental datasets are simple.

5. Conclusion

In this paper, we performed 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 gave the overall performance analysis about our results and introduced some tricks about adjusting hyper-parameters. We noted 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 could facilitate a better understanding of the deep learning models on RGB-D datasets. In the future, we will focus on collecting large-scale RGB-D datasets for better gauging new algorithms and finding convenient ways to adjust hyper-parameters.

[1] Peter Allen. *Robotic object recognition using vision and touch*, volume 34. 2012.

[2] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39, 2011.

[3] Jing Bai, Yan Wu, Junming Zhang, and Fuqiang Chen. Subset based deep learning for rgb-d object recognition. *Neurocomputing*, 2015.

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

[5] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*, 19:153, 2007.

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

[7] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[8] Manuel Blum, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *IEEE International Conference on Robotics and Automation*, pages 1298–1303, 2012.

[9] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402, 2013.

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

[11] Jake Bouvrie. Notes on convolutional neural networks. 2006.

[12] Björn Browatzki, Jan Fischer, Birgit Graf, HH Bulthoff, and Christian Wallraven. 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, 2011.

[13] Ziyun Cai, Jungong Han, Li Liu, and Ling Shao. Rgb-d datasets using microsoft kinect or similar sensors: a survey. *Multimedia Tools and Applications*, pages 1–43, 2016.

[14] Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[15] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.

[16] Leandro Cruz, Djalma Lucio, and Luiz Velho. Kinect and rgbd images: Challenges and applications. In *Conference on Graphics, Patterns and Images Tutorials*, pages 36–49, 2012.

[17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[18] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *IEEE International Conference on Robotics and Automation*, pages 1691–1696, 2012.

[19] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Interspeech*, pages 1964–1968, 2014.

[20] Raul Fernandez, Asaf Rendel, Bhuvana Ramabhadran, and Ron Hoory. Prosody contour prediction with long short-term memory, bidirectional, deep recurrent neural networks. In *Interspeech*, pages 2268–2272, 2014.

[21] Alexander Förster, Alex Graves, and Jürgen Schmidhuber. Rnn-based learning of compact maps for efficient robot localization. In *European Symposium on Artificial Neural Networks*, pages 537–542, 2007.

[22] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[23] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *International Conference on Machine Learning*, pages 513–520, 2011.

[24] Javier Gonzalez-Dominguez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno. Automatic language identification using long short-term memory recurrent neural networks. In *INTERSPEECH*, pages 2155–2159, 2014.

[25] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[26] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552, 2009.

[27] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740, 2015.

[28] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360, 2014.

[29] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, 2013.

[30] A Handa, V Ptrucean, V Badrinarayanan, R Cipolla, et al. Understanding realworld indoor sceneswith synthetic data. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2016, pages 4077–4085, 2016.

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

[32] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

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

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

[35] Armin Hornung, Kai M Wurm, and Maren Bennewitz. Humanoid robot localization in complex indoor environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1690–1695, 2010.

[36] Jian-Fang Hu, Wei-Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5344–5352, 2015.

[37] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, 1968.

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

[39] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[40] Lu Jin, Shenghua Gao, Zechao Li, and Jinhui Tang. Hand-crafted features or machine learnt features? together they improve rgb-d object recognition. In *International Symposium on Multimedia*, pages 311–319, 2014.

[41] Graeme A Jones, Nikos Paragios, and Carlo S Regazzoni. *Video-based surveillance systems: computer vision and distributed processing*. 2012.

[42] Yinda Zhang Mingru Bai Pushmeet Kohli, Shahram Izadi, and Jianxiong Xiao. Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding. *arXiv preprint arXiv:1603.04922*, 2016.

[43] Yu Kong and Yun Fu. Bilinear heterogeneous information machine for rgb-d action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1054–1062, 2015.

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

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[46] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[47] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *International Conference on Robotics and Automation*, pages 1817–1824, 2011.

[48] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, pages 473–480, 2007.

[49] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.

[50] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[51] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[52] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, pages 1096–1104, 2009.

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

[54] Shao-Zi Li, Bin Yu, Wei Wu, Song-Zhi Su, and Rong-Rong Ji. Feature learning based on sae–pca network for human gesture recognition in rgbd images. *Neurocomputing*, 151:565–573, 2015.

[55] Hui Liang, Junsong Yuan, and Daniel Thalmann. 3d fingertip and palm tracking in depth image sequences. In *ACM International Conference on Multimedia*, pages 785–788, 2012.

[56] Liang Lin, Keze Wang, Wangmeng Zuo, Meng Wang, Jiebo Luo, and Lei Zhang. A deep structured model with radius–margin bound for 3d human activity recognition. *International Journal of Computer Vision*, pages 1–18, 2015.

[57] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833, 2016.

[58] Li Liu and Ling Shao. Learning discriminative representations from rgb-d video data. In *International joint conference on Artificial Intelligence*, pages 1493–1500, 2013.

[59] Xinda Liu, Xueming Wang, and Shuqiang Jiang. Rgb-d scene classification via heterogeneous model fusion. In *IEEE International Conference on Image Processing*, pages 499–503, 2016.

[60] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[61] Erik Marchi, Giacomo Ferroni, Florian Eyben, Leonardo Gabrielli, Stefano Squartini, and Björn Schuller. 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, 2014.

[62] Abdel-rahman Mohamed and Geoffrey Hinton. Phone recognition using restricted boltzmann machines. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 4354–4357, 2010.

[63] Florent Monay and Daniel Gatica-Perez. On image auto-annotation with latent space models. In *International Conference on Multimedia*, pages 275–278, 2003.

[64] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144, 2006.

[65] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[66] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras. Application of deep belief networks for natural language understanding. *Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, 2014.

[67] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does human action recognition require? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[68] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[69] Amirhossein Shantia, Rik Timmers, Lambert Schomaker, and Marco Wiering. Indoor localization by denoising autoencoders and semi-supervised learning in 3d simulated environment. In *International Joint Conference on Neural Networks*, pages 1–7, 2015.

[70] Ling Shao, Xiantong Zhen, Dacheng Tao, and Xuelong Li. Spatio-temporal laplacian pyramid coding for action recognition. *IEEE Transactions on Cybernetics*, 44(6):817–827, 2014.

[71] Yuanlong Shao, Yuan Zhou, Xiaofei He, Deng Cai, and Hujun Bao. Semi-supervised topic modeling for image annotation. In *International Conference on Multimedia*, pages 521–524, 2009.

[72] Wei Shen, Ke Deng, Xiang Bai, Tommer Leyvand, Baining Guo, and Zhuowen Tu. Exemplar-based human action pose correction and tagging. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1784–1791, 2012.

[73] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *International Conference on World Wide Web*, pages 373–374, 2014.

[74] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *International Conference on Computer Vision Workshops*, pages 601–608, 2011.

[75] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[76] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012.

[77] Luciano Spinello and Kai Oliver Arras. People detection in rgb-d data. In *International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[78] Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Scene in the loop: Towards adaptation-by-tracking in rgb-d data. In *Proc. Workshop RGB-D, Adv. Reason. Depth Cameras*, 2012.

[79] Ilya Sutskever and Geoffrey E Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Artificial Intelligence and Statistics*, volume 2, pages 548–555, 2007.

[80] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[81] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *Asian Conference on Computer Vision*, pages 525–538. 2013.

[82] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, pages 1345–1352, 2006.

[83] James Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110, 2012.

[84] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine learning*, pages 1096–1103, 2008.

[85] Sarvesh Vishwakarma and Anupam Agrawal. A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29(10):983–1009, 2013.

[86] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

[87] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[88] Anran Wang, Jianfei Cai, Jiwen Lu, and Tat-Jen Cham. Mmss: multi-modal sharable and specific feature learning for rgb-d object recognition. In *IEEE International Conference on Computer Vision*, pages 1125–1133, 2015.

[89] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Relational stacked denoising autoencoder for tag recommendation. In *AAAI Conference on Artificial Intelligence*, pages 3052–3058, 2015.

[90] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297, 2012.

[91] Martin Wollmer, Christoph Blaschke, Thomas Schindl, Björn Schuller, Berthold Farber, Stefan Mayer, and Benjamin Trefflich. Online driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):574–582, 2011.

[92] Di Wu, Lionel Pigou, Pieter-Jan Kindermans, LE Nam, Ling Shao, Joni Dambre, and Jean-Marc Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE Transactions on Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[93] Di Wu and Ling Shao. Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–731, 2014.

[94] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[95] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *IEEE International Conference on Robotics and Automation workshop*, volume 2, 2010.

[96] Chen Xing, Li Ma, and Xiaoquan Yang. Stacked denoise autoencoder based feature extraction and classification for hyperspectral images. *Journal of Sensors*, 2016, 2015.

[97] Yan Xu, Tao Mo, Qiwei Feng, Peilin Zhong, Maode Lai, Eric I Chang, et al. 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, 2014.

[98] Hongyang Xue, Yao Liu, Deng Cai, and Xiaofei He. Tracking people in rgbd videos using deep learning and motion clues. *Neurocomputing*, 204:70–76, 2016.

[99] Hasan FM Zaki, Faisal Shafait, and Ajmal Mian. Convolutional hypercube pyramid for accurate rgb-d object category and instance recogni-

tion. In *IEEE International Conference on Robotics and Automation*, pages 1685–1692, 2016.

[100] Hongyuan Zhu, Jean-Baptiste Weibel, and Shijian Lu. Discriminative multi-modal feature fusion for rgbd indoor scene recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2969–2976, 2016.