Deep learning for visual understanding

Guo, Y.; Guo Y.

Chapter 2

A Comprehensive Review of Deep Learning Methods and Applications

Deep learning algorithms are a subset of the machine learning algorithms, which aim at discovering multiple levels of distributed representations. Recently, numerous deep learning algorithms have been proposed to solve traditional artificial intelligence problems. This chapter aims to review the state-of-the-art in deep learning algorithms in computer vision by highlighting the contributions and challenges from about 200 recent research papers. It first gives an overview of various deep learning approaches and their recent developments, and then briefly describes their applications in diverse vision tasks, such as image classification, object detection, image retrieval, semantic segmentation and human pose estimation. Finally, the chapter summarizes the future trends and challenges in designing and training deep neural networks.

2.1 Introduction

Deep learning is a subfield of machine learning which attempts to learn high-level abstractions in data by utilizing hierarchical architectures. It is an emerging approach and has been widely applied in traditional artificial intelligence domains, such as semantic parsing [11], natural language processing [12], computer vision [13, 14] and many more. There are mainly three important reasons for the booming of deep learning today: the dramatically increased chip processing abilities (e.g. GPU units), the significantly lowered cost of computing hardware, and the considerable advances in the machine learning algorithms [15].

Deep learning approaches have been extensively reviewed and discussed in recent years [15–19]. Among those Schmidhuber et al. [17] emphasized the important inspirations and technical contributions in a historical timeline format, while Bengio [18] examined the challenges of deep learning research and proposed a few forward-looking research directions. Deep networks have shown to be successful for computer vision tasks because they can extract appropriate features while jointly performing discrimination [15, 20]. In recent ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competitions [21], deep learning methods have been widely utilized by researchers and achieved top accuracy scores.

This chapter is intended to be useful to general neural computing, computer vision and multimedia researchers who are interested in state-of-the-art deep learning studies for computer vision. It provides an overview of various deep learning algorithms and their applications, especially those that can be applied in the computer vision domain.

The remainder of this chapter is organized as follows: In Section 2.2, we divide the deep learning algorithms into four categories: Convolutional Neural Networks, Restricted Boltzmann Machines, Autoencoder and Sparse Coding. Some well-known models in these categories as well as their developments are listed. We also describe the contributions and limitations for these models in this section. In Section 2.3, we describe the achievements of deep learning schemes in various computer vision applications, e.g. image classification, object detection, image retrieval, semantic segmentation and human pose estimation. The results on

these applications are shown and compared in the pipeline of their commonly used datasets. In Section 2.4, along with the success deep learning methods have achieved, we also face several challenges when designing and training the deep networks. In this section, we summarize some major challenges for deep learning, together with the inherent trends that might be developed in the future. In Section 2.5, we conclude this chapter.

2.2 Methods and recent developments

In recent years, deep learning has been extensively studied in the field of computer vision and as a consequence, a large number of related approaches have emerged. Generally, these methods can be divided into four categories according to the basic method they are derived from: Convolutional Neural Networks (CNNs), Restricted Boltzmann Machines (RBMs), Autoencoder and Sparse Coding.

The categorization of deep learning methods along with some representative works is shown in Figure 2.1.

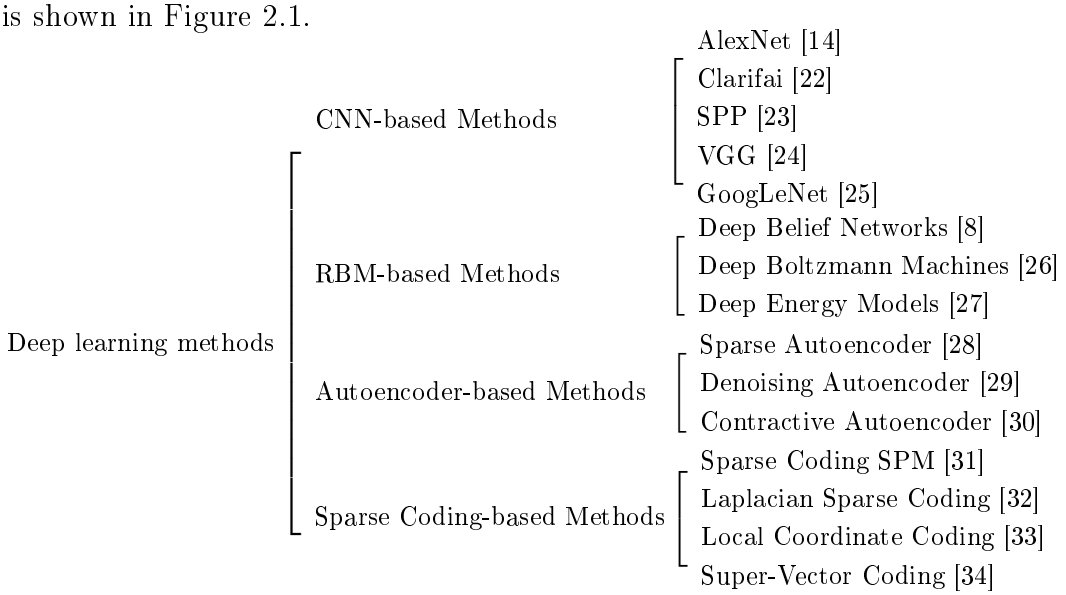


Figure 2.1: A categorization of the deep learning methods and their representative works.

In the next four parts, we will briefly review each of these deep learning methods and their most recent developments.

2.2.1 Convolutional Neural Networks (CNNs)

The Convolutional Neural Networks (CNN) is one of the most notable deep learning approaches where multiple layers are trained in an end-to-end manner [35]. It has been found highly effective and is also the most commonly used in diverse computer vision applications.

The pipeline of the general CNN architecture is shown in Figure 2.2.

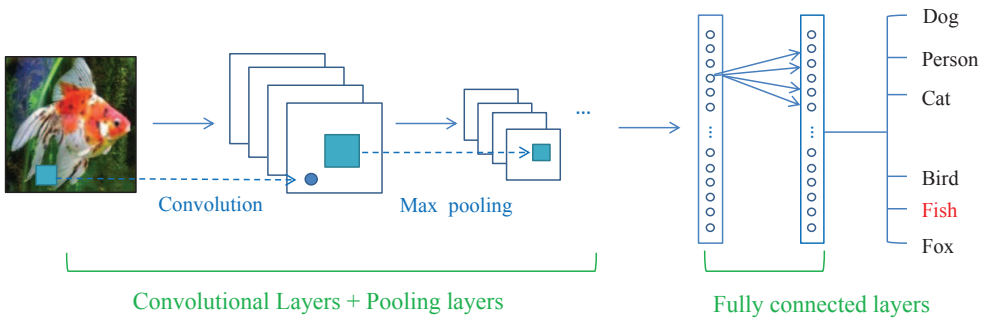


Figure 2.2: The pipeline of the general CNN architecture.

Generally, a CNN consists of three main neural layers, which are convolutional layers, pooling layers, and fully connected layers. Different kinds of layers play different roles. In Figure 2.2, a general CNN architecture for image classification [14] is shown layer-by-layer. There are two stages for training the network: a forward stage and a backward stage. First, the main goal of the forward stage is to represent the input image with the current parameters (weights and bias) in each layer. Then the prediction output is used to compute the loss cost with the ground truth labels. Second, based on the loss cost, the backward stage computes the gradients of each parameter with chain rules. All the parameters are updated based on the gradients, and are prepared for the next forward computation. After sufficient iterations of the forward and backward stages, the network learning can be stopped.

Next, we will first introduce the functions along with the recent developments of each layer, and then summarize the commonly used training strategies of the networks. Finally, we present several well-known CNN models, derived models, and conclude with the current tendency for using these models in real applications.

2.2.1.1 Types of layers

Generally, a CNN is a hierarchical neural network whose convolutional layers alternate with pooling layers, followed by some fully connected layers (see Figure 2.2). In this section, we will present the function of the three layers and briefly review the recent advances that have appeared on those layers.

- **Convolutional layers**

In the convolutional layers, a CNN utilizes various kernels to convolve the whole image as well as the intermediate feature maps, generating various feature maps, as shown in Figure 2.3.

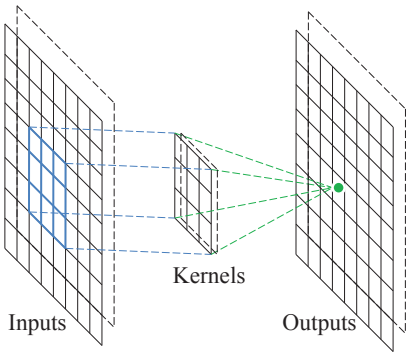


Figure 2.3: The operation of the convolutional layer.

There are three main advantages of the convolution operation [36]: 1) the weight sharing mechanism in the same feature map reduces the number of parameters; 2) local connectivity learns correlations among neighboring pixels; 3) invariance to the location of the object.

Due to the benefits introduced by the convolution operation, some well-known research papers use it as a replacement for the fully connected layers to accelerate the learning process [25, 37]. One interesting approach of handling the convolutional layers is the Network in Network (NIN) [38] method, where the main idea is to substitute the conventional convolutional layer with a small multilayer perceptron consisting of multiple fully connected layers with nonlinear activation functions, thereby replacing the linear filters with nonlinear neural networks. This method achieves good results in image classification.

- **Pooling layers**

Generally, a pooling layer follows a convolutional layer, and can be used to reduce the dimensions of feature maps and network parameters. Similar to convolutional layers, pooling layers are also translation invariant, because their computations take neighboring pixels into account. Average pooling and max pooling are the most commonly used strategies. Figure 2.4 gives an example for a max pooling process. For 8×8 feature maps, the output maps reduce to 4×4 dimensions, with a max pooling operator which has size 2×2 and stride 2.

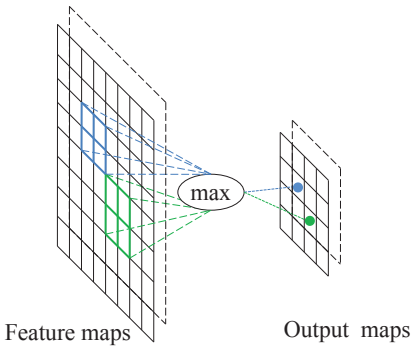


Figure 2.4: The operation of the max pooling layer.

For max pooling and average pooling, Boureau et al. [39] provided a detailed theoretical analysis of their performances. Scherer et al. [40] further conducted a comparison between the two pooling operations and found that max-pooling can lead to faster convergence, select superior invariant features and improve generalization. In recent years, various fast GPU implementations of CNN variants were presented, most of them utilize the max-pooling strategy [14, 41].

The pooling layers are the most extensively studied among the three layers. There are three well-known approaches related to the pooling layers, each having different purposes.

√ Stochastic Pooling

A drawback of max pooling is that it is sensitive to overfit the training set, making it hard to generalize well to test samples [36]. Aiming to solve this problem, Zeiler et al. [42] proposed a stochastic pooling approach which replaces the conventional deterministic pooling operations with a stochastic procedure, by randomly picking the activation within each pooling region according to a multinomial distribution. It is equivalent to standard max pooling but with many copies of the input image, each having small local deformations. This stochastic nature is helpful to prevent the overfitting problem.

√ Spatial Pyramid Pooling (SPP)

Normally, the CNN-based methods require a fixed-size input image. This restriction may reduce the recognition accuracy for images of arbitrary sizes. To eliminate this limitation, He et al. [23] utilized the general CNN architecture but replaced the last pooling layer with a spatial pyramid pooling layer. The spatial pyramid pooling can extract fixed-length representations from arbitrary images (or regions), generating a flexible solution for handling different scales, sizes, aspect ratios, and can be applied in any CNN structure to boost the performance of this structure.

√ Def-Pooling

Handling deformation is a fundamental challenge in computer vision, especially for the object recognition task. Max pooling and average pooling are useful in handling deformation but they are not able to learn the deformation constraint and geometric model of object parts. To deal with deformation more efficiently, Ouyang et al. [43] introduced a new deformation constrained pooling layer, called def-pooling layer, to enrich the deep model by learning the deformation of visual patterns. It can substitute the traditional max-pooling layer at any information abstraction level.

Because of the different purposes and procedures the pooling strategies are designed for, various pooling strategies could be combined to boost the performance of a CNN.

- **Fully-connected layers**

Following the last pooling layer in the network as seen in Figure 2.2, there are several fully-connected layers converting the 2D feature maps into a 1D feature vector, for further feature representation, as seen in Figure 2.5.

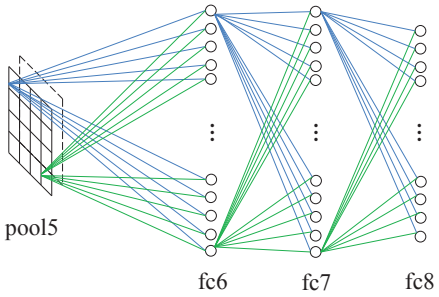


Figure 2.5: The operation of the fully-connected layer.

Fully-connected layers perform like a traditional neural network and contain about 90% of the parameters in a CNN. It enables us to feed forward the neural network into a vector with a pre-defined length. We could either feed forward the vector into certain number categories for image classification [14] or take it as a feature vector for follow-up processing [44].

Changing the structure of the fully-connected layer is uncommon, however an example came in the transferred learning approach [45], which preserved the parameters learned by ImageNet [14], but replaced the last fully-connected layer with two new fully-connected layers to adapt to the new visual recognition tasks.

The drawback of these layers is that they contain many parameters, which results in a large computational effort for training them. Therefore, a promising and commonly applied direction is to remove these layers or decrease the connections with a certain method. For example, GoogLeNet [25] designed a deep and wide network while keeping the computational budget constant, by switching from fully connected to sparsely connected architectures.

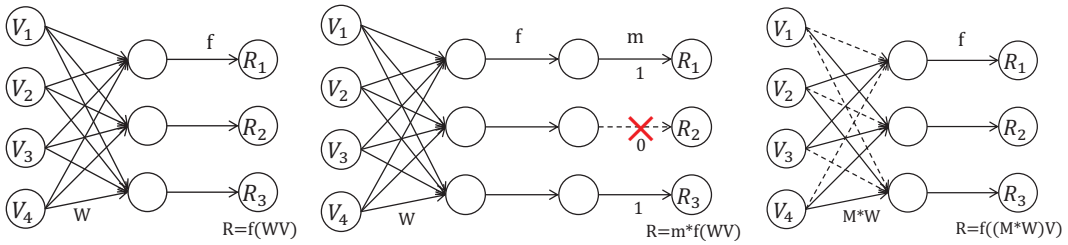
2.2.1.2 Training Strategy

Compared to shallow learning, the advantage of deep learning is that it can build deep architectures to learn more abstract information. However, the large amount of parameters introduced may also lead to another problem: overfitting. Recently, numerous regularization methods have emerged in defense of overfitting, including the stochastic pooling mentioned above. In this section, we will introduce several other regularization techniques that may influence the training performance.

- **Dropout and DropConnect**

Dropout was proposed by Hinton et al. [46] and explained in-depth by Baldi et al. [47]. During each training case, the algorithm will randomly omit half of the feature detectors in order to prevent complex co-adaptations on the training data and enhance the generalization ability. This method was further improved in [48–53]. Specifically, Warde-Farley et al. [53] analyzed the efficacy of dropouts and suggested that dropout is an extremely effective ensemble learning method.

One well-known generalization derived from Dropout is called DropConnect [54], which randomly drops weights rather than the activations. Experiments showed that it can achieve competitive or even better results on a variety of standard benchmarks, although slightly slower. Figure 2.6 gives a comparison of No-Drop, Dropout and DropConnect networks [54].



(a) No-drop Network (b) Dropout Network (c) Dropconnect Network

Figure 2.6: A comparison of No-Drop, Dropout and DropConnect networks [54].

- **Data Augmentation**

When a CNN is applied to visual object recognition, data augmentation is often utilized to generate additional data without introducing extra labeling costs. The well-known AlexNet [14] employed two distinct forms of data augmentation: the first form of data augmentation consists of generating image translations and horizontal reflections, and the second form consists of altering the intensities of the RGB channels in training images. Howard et al. [55] took AlexNet as the base model and added additional transformations that improved the translation invariance and color invariance by extending image crops with extra pixels and adding additional color manipulations. This data augmentation method was widely utilized by some of the more recent studies [23, 25]. Dosovitskiy et al. [56] proposed an unsupervised feature learning approach based on data augmentation: it first randomly sampled a set of image patches and declares each of them as a surrogate class, then extended these classes by applying transformations corresponding to translation, scale, color and contrast. Finally, it trained a CNN to discriminate between these surrogate classes. The features learnt by the network showed good results on a variety of classification tasks. Aside from the classic methods such as scaling, rotating and cropping, Wu et al. [57] further adopted color casting, vignetting and lens distortion techniques, which produced more training examples with broad coverage.

- **Pre-training and fine-tuning**

Pre-training means to initialize the networks with pre-trained parameters rather than randomly set parameters. It is quite popular in models based on CNNs, due to the advantages that it can accelerate the learning process as well as improve the generalization ability. Erhan et al. [58] has conducted extensive simulations on the existing algorithms to find why pre-trained networks work better than networks trained in a traditional way. As AlexNet [14] achieved excellent performance and is released to the public, numerous approaches choose AlexNet trained on ImageNet2012 as their baseline deep model [23, 44, 45], and use fine-tuning of the parameters according to their specific tasks. Nevertheless, there are approaches [43, 59, 60] that deliver better performance by training on other models, e.g. Clarifai [22], GoogLeNet [25], and VGG [24].

Fine-tuning is a crucial stage for refining models to adapt to specific tasks and datasets. In general, fine-tuning requires class labels for the new training dataset, which are used for computing the loss functions. In this case, all layers of the new model will be initialized based on the pre-trained model, such as AlexNet [14], except for the last output layer that depends on the number of class labels of the new dataset and will therefore be randomly initialized. However, in some occasions, it is very difficult to obtain the class labels for any new dataset. To address this problem, a similarity learning objective function was proposed to be used as the loss functions without class labels [61], so the back-propagation can work normally and allow the model to be refined layer by layer. There are also many research results describing how to transfer the pre-trained model efficiently. A new way is defined to quantify the degree to which a particular layer is general or specific [62], namely how well features at that layer transfers from one task to another. They concluded that initializing a network with transferred features from almost any number of layers can give a boost to generalization performance after fine-tuning to a new dataset.

In addition to the regularization methods described above, there are also other common methods such as weight decay, weight tying and many more [17]. Weight decay works by adding an extra term to the cost function to penalize the parameters, preventing them from exactly modeling the training data and therefore helping to generalize to new examples [14]. Weight tying allows models to learn good representations of the input data by reducing the number of parameters in Convolutional Neural Networks [63].

One noteworthy thing is that these regularization techniques for training are not mutually exclusive and they can be combined to boost the performance.

2.2.1.3 CNN architecture

With the recent developments of CNN schemes in the computer vision domain, some well-known CNN models have emerged. In this section, we first describe the commonly used CNN models, and then summarize their characteristics in their

applications. The configurations and the primary contributions of several typical CNN models are listed in Table 2.1.

Table 2.1: CNN models and their achievements in ILSVRC classification competitions.

Method	Year	Place	Configuration	Contribution
AlexNet [14]	2012	1^{st}	Five convolutional layers + three fully connected layers	an important CNN architecture which set the tone for many computer vision researches
Clarifai [22]	2013	1^{st}	Five convolutional layers + three fully connected layers	insight into the function of intermediate feature layers
SPP [23]	2014	3^{rd}	Five convolutional layers + three fully connected layers	proposed the 'spatial pyramid pooling' to remove the requirement of image resolution
VGG [24]	2014	2^{nd}	Thirteen/Fifteen convolutional layers + three fully connected layers	a thorough evaluation of networks of increasing depth
GoogLeNet [25]	2014	1^{st}	Twenty-one convolutional layers + one fully connected layer	increased the depth and width without raising the computational requirements

AlexNet [14] is a significant CNN architecture, which consists of five convolutional layers and three fully connected layers. After inputting one fixed-size (224×224) image, the network would repeatedly convolve and pool the activations, then forward the results into the fully-connected layers. The network was trained on ImageNet and integrated various regularization techniques, such as data augmentation, dropout, etc. AlexNet won the ILSVRC2012 competition [21], and set the tone for the surge of interest in deep convolutional neural networks. Nevertheless, there are two major drawbacks of this model: 1) it requires a fixed resolution of the image; 2) there is no clear understanding of why it performs so well.

In 2013, Zeiler et al. [22] introduced a novel visualization technique to give insight into the inner workings of the intermediate feature layers. These visualizations enabled them to find architectures that outperform AlexNet [14] on the ImageNet classification benchmark, and the resulting model, Clarifai, received top performance in the ILSVRC2013 competition.

As for the requirement of a fixed resolution, He et al. [23] proposed a new pooling strategy, i.e. spatial pyramid pooling, to eliminate the restriction of the image size. The resulting SPP-net could boost the accuracy of a variety of published CNN architectures despite their different designs.

In addition to the commonly used configuration of the CNN structure (five convolutional layers plus three fully connected layers), there are also approaches trying to explore deeper networks. In contrast to AlexNet, VGG [24] increased the depth of the network by adding more convolutional layers and taking advantage of very small convolutional filters in all layers. Similarly, Szegedy et al. [25] proposed a model, GoogLeNet, which also has quite a deep structure (22 layers) and has achieved leading performance in the ILSVRC2014 competition [21].

Despite top-tier classification performances have been achieved by various models, CNN-related models and applications are not limited to only image classification. Based on these models, new frameworks have been derived to address other challenging tasks, such as object detection, semantic segmentation, etc.

There are two well-known derived frameworks: RCNN (Regions with CNN features) [44] and FCN (fully convolutional network) [60], mainly designed for object detection and semantic segmentation respectively, as shown in Figure 2.7.

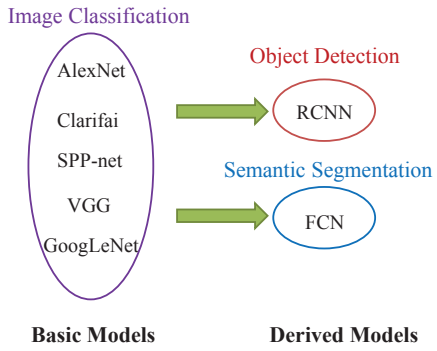


Figure 2.7: CNN basic models and derived models.

The core idea of RCNN is to generate multiple object proposals, extract features from each proposal using a CNN, and then classify each candidate window with a category-specific linear SVM. The "recognition using regions" paradigm received

encouraging performance in object detection and has gradually become the general pipeline for recent promising object detection algorithms [64–67]. However, the performance of RCNN relies too much on the precision of the object location, which may limit its robustness. Besides, the generation and processing of large number of proposals would also decrease its efficiency. Recent developments [64–66, 68, 69] are mainly focused on these two aspects.

RCNN takes the CNN models as feature extractor and does not make any change to the networks. In contrast, FCN proposes to recast the CNN models as fully convolutional nets, which removes the restriction of image resolution and could produce correspondingly-sized output efficiently. Although FCN is proposed mainly for semantic segmentation, the technique could also be utilized in other applications, e.g. image classification [70], edge detection [71] etc.

Aside from creating various models, the usage of these models also demonstrates several characteristics:

- **Large Networks**

One intuitive idea is to improve the performance of CNNs by increasing their sizes, which includes increasing the depth (the number of levels) and the width (the number of units at each level) [25]. Both aforementioned GoogLeNet [25] and VGG [24] utilized quite large networks, 22 layers and 19 layers respectively, demonstrating that increasing the size is beneficial for image recognition.

Jointly training multiple networks could lead to better performance than a single one. There are also many researchers [43, 72, 73] who designed large networks by combining different deep structures in cascade mode, where the output of the former networks is utilized by the latter ones, as shown in Figure 2.8.

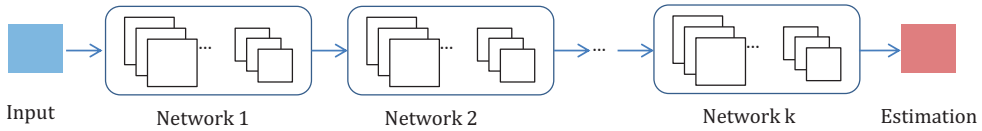


Figure 2.8: Combining deep structures in cascade mode.

The cascade architecture can be utilized to handle different tasks, and the function of the prior networks (i.e. the output) may vary with the tasks. For example, Wang et al. [73] connected two networks for object extraction, and the first network is used for object localization. Therefore, the output is the corresponding coordinates of the object. Sun et al. [72] proposed three-level carefully designed convolutional networks to detect facial keypoints. The first level provides highly robust initial estimations, while the following two levels fine-tune the initial prediction. Similarly, Ouyang et al. [43] adopted a multi-stage training scheme proposed by Zeng et al. [74], i.e. classifiers at the previous stages jointly work with the classifiers at the current stage to deal with misclassified samples.

- **Multiple Networks**

Another tendency in current applications is to combine the results of multiple networks, where each of the networks can execute the task independently, instead of designing a single architecture and jointly training the networks inside to execute the task, as seen in Figure 2.9.

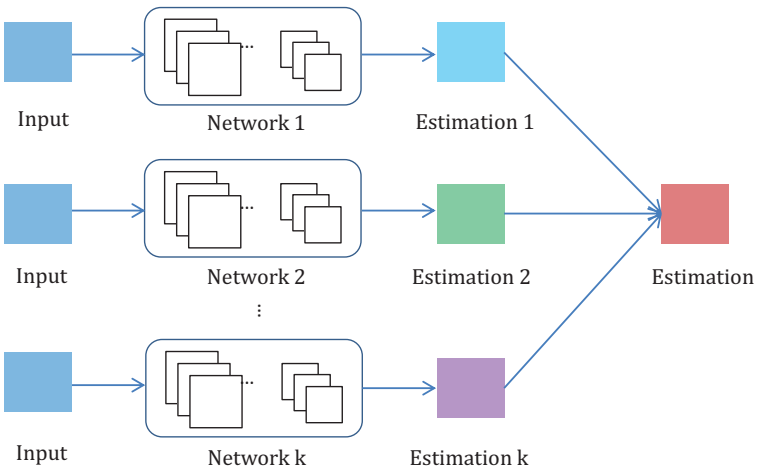


Figure 2.9: Combining the results of multiple networks.

Miclut et al. [75] gave some insight into how we should generate the final results when we have received a set of scores. Prior to the AlexNet [14], Ciresan et al. [13] proposed a method called Multi-Column DNN (MCDNN) which combines several DNN columns and averages their predictions. This model achieved

human-competitive results on tasks such as the recognition of handwritten digits or traffic signs. Recently, Ouyang et al. [43] also conducted an experiment to evaluate the performance of model combination strategies. It learnt 10 models with different settings and combined them in an averaging scheme. Results show that models generated in this way have high diversity and are complementary to each other in improving the detection results.

- **Diverse Networks**

Aside from altering the CNN structure, some researchers also attempt to introduce information from other sources, e.g. combining them with shallow structures, integrating contextual information, as illustrated in Figure 2.10.

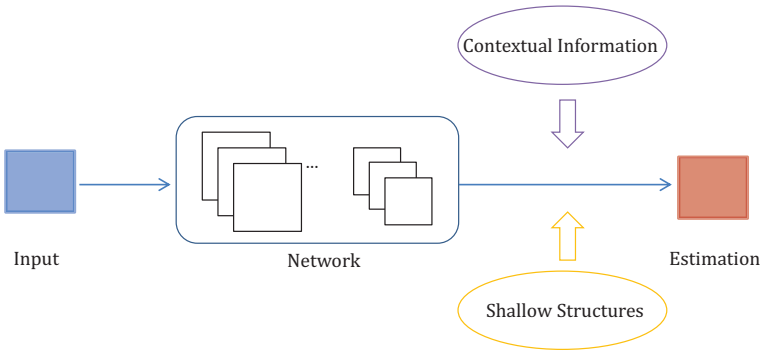


Figure 2.10: Combining a deep network with information from other sources.

Shallow methods can give additional insight into the problem. In the literature, examples can be found about combining shallow methods and deep learning frameworks [76], i.e. take a deep learning method to extract features and input these features to the shallow learning method, e.g. an SVM. One of the most representative and successful algorithms is the RCNN method [44], which feeds the highly distinctive CNN features into a SVM for the final object detection task. Besides, deep CNNs and Fisher Vectors (FV) are complementary [77] and can also be combined to significantly improve the accuracy of image classification.

Contextual information is sometimes available for an object detection task, and it is possible to integrate global context information with the information from the bounding box. In the ImageNet Large Scale Visual Recognition Challenge 2014

(ILSVRC 2014), the winning team NUS concatenated all the raw detection scores and combined them with the outputs from a traditional classification framework by context refinement [78]. Similarly, Ouyang et al. [43] also took the 1000-class image classification scores as the contextual features for object detection.

2.2.2 Restricted Boltzmann Machines (RBMs)

The Restricted Boltzmann Machine (RBM) is a generative stochastic neural network, and was proposed by Hinton et al. in 1986 [79]. It is a variant of the Boltzmann Machine, with the restriction that the visible units and hidden units must form a bipartite graph. This restriction allows for efficient training algorithms, in particular the gradient-based contrastive divergence algorithm [80].

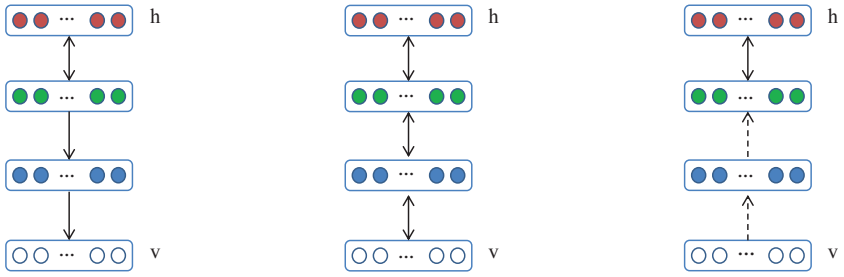
Since the model is a bipartite graph, the hidden units Hand the visible units V_1 are conditionally independent. Therefore,

$$P(H|V_1) = P(H_1|V_1)P(H_2|V_1)\cdots P(H_n|V_1) \tag{2.1}$$

Hinton [81] gave a detailed explanation and provided a practical way to train RBMs. Further work in [82] discussed the main difficulties of training RBMs, their underlying reasons and proposed a new algorithm, which consists of an adaptive learning rate and an enhanced gradient, to address those difficulties.

A well-known development of RBM can be found in [83]: the model approximates the binary units with noisy rectified linear units to preserve information about relative intensities as information travels through multiple layers of feature detectors. The refinement not only functions well in this model, but is also widely employed in various CNN-based approaches [14, 42].

Utilizing RBMs as learning modules, we can compose the following deep models: Deep Belief Networks (DBNs), Deep Boltzmann Machines (DBMs) and Deep Energy Models (DEMs). The comparison between the three models is shown in Figure 2.11.



(a) DBN1 (b) DBM1 (c) DEM1

Figure 2.11: The comparison of DBN, DBM and DEM [27].

DBNs have undirected connections at the top two layers which form an RBM and directed connections to the lower layers. DBMs have undirected connections between all layers of the network. DEMs have deterministic hidden units for the lower layers and stochastic hidden units at the top hidden layer [27].

A summary of these three deep models is provided in Table 2.2.

Table 2.2: An overview of representative RBM-based methods.

Method	characteristics	advantages	drawbacks
DBN [8]	undirected connections at the top two layers and directed connections at the lower layers	1.Properly initializes the network, which prevents poor local optima to some extent; 2.Training is unsupervised, which removes the necessity of labeled data for training	Due to the initialization process, it is computationally expensive to create a DBN model
DBM [26]	undirected connections between all layers of the network	Deals more robustly with ambiguous inputs by incorporating top-down feedback	The joint optimization is time-consuming
DEM [27]	deterministic hidden units for the lower layers and stochastic hidden units at the top hidden layer	Produces better generative models by allowing the lower layers to adapt to the training of higher layers	The learnt initial weight may not have good convergence

In the next sections, we will explain these models and describe their applications to computer vision tasks respectively.

2.2.2.1 Deep Belief Networks (DBNs)

The Deep Belief Network (DBN), proposed by Hinton [8], was a significant advance in deep learning. It is a probabilistic generative model which provides a joint probability distribution over observable data and labels. A DBN first takes advantage of an efficient layer-by-layer greedy learning strategy to initialize the deep network, and then fine-tunes all of the weights jointly with the desired outputs. The greedy learning procedure has two main advantages [84]: (1) it generates a proper initialization of the network, addressing the difficulty in parameter selection which may result in poor local optima to some extent; (2) the learning procedure is unsupervised and requires no class labels, so it removes the necessity of labeled data for training. However, creating a DBN model is a computationally expensive task that involves training several RBMs, and it is not clear how to approximate maximum-likelihood training to further optimize the model [19].

DBNs successfully focused researchers' attention on deep learning and as a consequence, many variants were created [85–88]. Nair et al. [88] developed a modified DBN where the top-layer model utilizes a third-order Boltzmann machine for object recognition. The model in [85] learned a two-layer model of natural images using sparse RBMs, in which the first layer learns local, oriented, edge filters, and the second layer captures a variety of contour features as well as corners and junctions. To improve the robustness against occlusion and random noise, Lee et al. [89] applied two strategies: one is to take advantage of sparse connections in the first layer of the DBN to regularize the model, and the other is to develop a probabilistic de-noising algorithm. When applied to computer vision tasks, a drawback of DBNs is that they do not consider the 2D structure of an input image. To address this problem, the Convolutional Deep Belief Network (CDBN) was introduced [86]. CDBN utilized the spatial information of neighboring pixels by introducing convolutional RBMs, generating a translation invariant generative model that scales well with high dimensional images. The algorithm was further extended in [90] and achieved excellent performance in face verification.

2.2.2.2 Deep Boltzmann Machines (DBMs)

The Deep Boltzmann Machine (DBM), proposed by Salakhutdinov et al. [26], is another deep learning algorithm where the units are again arranged in layers. Compared to DBNs, whose top two layers form an undirected graphical model and whose lower layers form a directed generative model, the DBM has undirected connections across its structure.

Like the RBM, the DBM is also a subset of the Boltzmann family. The difference is that the DBM possesses multiple layers of hidden units, with units in odd-numbered layers being conditionally independent of even-numbered layers, and vice versa. Given the visible units, calculating the posterior distribution over the hidden units is no longer tractable, resulting from the interactions between the hidden units. When training the network, a DBM would jointly train all layers of a specific unsupervised model, and instead of maximizing the likelihood directly, the DBM uses a stochastic maximum likelihood (SML) [91] based algorithm to maximize the lower bound on the likelihood, i.e. performing only one or a few updates using a Markov chain Monte Carlo (MCMC) method between each parameter update. To avoid ending up in poor local minima which leave many hidden units effectively dead, a greedy layer-wise training strategy is also added into the layers when pre-training the DBM network, much in the same way as the DBN [19].

This joint learning has brought promising improvements, both in terms of likelihood and the classification performance of the deep feature learner. However, a crucial disadvantage of DBMs is the time complexity of approximate inference is considerably higher than DBNs, which makes the joint optimization of DBM parameters impractical for large datasets. To increase the efficiency of DBMs, some researchers introduced an approximate inference algorithm [92, 93], which utilizes a separate 'recognition' model to initialize the values of the latent variables in all layers, thus effectively accelerating the inference.

There are also many other approaches that aim to improve the effectiveness of DBMs. The improvements can either take place at the pre-training stage [94, 95] or at the training stage [96, 97]. For example, Montavon et al. [96] introduced

the centering trick to improve the stability of a DBM and made it to be more discriminative and generative. The multi-prediction training scheme [98] was utilized to jointly train the DBM which outperforms the previous methods in image classification proposed in [97].

2.2.2.3 Deep Energy Models (DEMs)

The Deep Energy Model (DEM), introduced by Ngiam et al. [27], is a more recent approach to train deep architectures. Unlike DBNs and DBMs which share the property of having multiple stochastic hidden layers, the DEM just has a single layer of stochastic hidden units for efficient training and inference.

The model utilizes deep feed forward neural networks to model the energy landscape and is able to train all layers simultaneously. By evaluating the performance on natural images, it demonstrated the joint training of multiple layers yields qualitative and quantitative improvements over greedy layer-wise training. Ngiam et al. [27] used Hybrid Monte Carlo (HMC) to train the model. There are also other options including contrastive divergence, score matching, and others. A similar work can be found in [99].

Although RBMs are not as suitable as CNNs for vision applications, there are also some good examples adopting RBMs to vision tasks. The Shape Boltzmann Machine was proposed by Eslami et al. [100] to handle the task of modeling binary shape images, which learns high quality probability distributions over object shapes, for both realism of samples from the distribution and generalization to new examples of the same shape class. Kae et al. [101] combined the CRF and the RBM to model both local and global structure in face segmentation, which has consistently reduced the error in face labeling. A new deep architecture has been presented for phone recognition [102] that combines a Mean-Covariance RBM feature extraction module with a standard DBN. This approach attacks both the representational inefficiency issues of GMMs and an important limitation of previous work applying DBNs to phone recognition.

2.2.3 Autoencoder

The autoencoder is a special type of artificial neural network used for learning efficient encodings [103]. Instead of training the network to predict some target value Y given inputs X, an autoencoder is trained to reconstruct its own inputs X, therefore, the output vectors have the same dimensionality as the input vector. The general process of an autoencoder is shown in Figure 2.12.

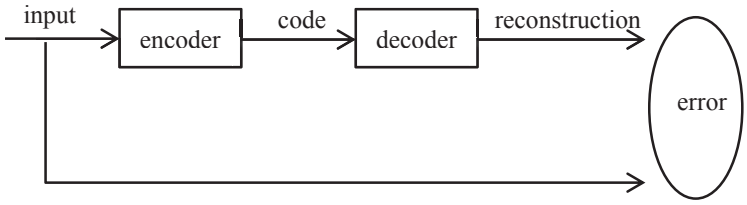


Figure 2.12: The pipeline of an autoencoder.

During the process, the autoencoder is optimized by minimizing the reconstruction error, and the corresponding code is the learned feature.

Generally, a single layer is not able to get the discriminative and representative features of raw data. Researchers now utilize the deep autoencoder, which forwards the code learnt from the previous autoencoder to the next, to accomplish their task.

The deep autoencoder was first proposed by Hinton et al. [104], and is still extensively studied in recent papers [105, 106]. A deep autoencoder is often trained with a variant of back-propagation, e.g. the conjugate gradient method. Though often reasonably effective, this model could become quite ineffective if errors are present in the first few layers. This may cause the network to learn to reconstruct the average of the training data. A proper approach to remove this problem is to pre-train the network with initial weights that approximate the final solution [104]. There are also variants of autoencoder proposed to make the representation as 'constant' as possible with respect to the changes in input.

In Table 2.3, we list some well-known variants of the autoencoder, and briefly summarize their characteristics and advantages. In the next sections, we de-

scribe three important variants: sparse autoencoder, denoising autoencoder and contractive autoencoder.

Table 2.3: Variants of the autoencoder.

Method	characteristics	advantages
Sparse Autoencoder [28, 106]	Adds a sparsity penalty to force the representation to be sparse	1. Make the categories to be more separable; 2. Make the complex data more meaningful; 3. In line with biological vision system
Denoising Autoencoder [29, 107]	Recovers the correct input from a corrupted version	More robust to noise
Contractive Autoencoder [30]	Adds an analytic contractive penalty to the reconstruction error function	Better captures the local directions of variation dictated by the data
Saturating Autoencoder [108]	Raises reconstruction error for inputs not near the data manifold	Limits the ability to reconstruct inputs which are not near the data manifold
Convolutional Autoencoder [109–111]	Shares weights among all locations in the input, preserving spatial locality	Utilizes the 2D image structure
Zero-bias Autoencoder [112]	Utilizes proper shrinkage function to train autoencoders without additional regularization	More powerful in learning representations on data with very high intrinsic dimensionality

2.2.3.1 Sparse Autoencoder

A sparse autoencoder aims to extract sparse features from raw data. The sparsity of the representation can either be achieved by penalizing the hidden unit biases [28, 85, 113] or by directly penalizing the output of hidden unit activations [114, 115].

Sparse representations have several potential advantages [28]: 1) using high-dimensional representations increases the likelihood that different categories will be easily separable, just as in the theory of SVMs; 2) sparse representations provide us with a simple interpretation of the complex input data in terms of a

number of 'parts'; 3) biological vision uses sparse representations in early visual areas [116].

A quite well-known variant of the sparse autoencoder is a nine-layer locally connected sparse autoencoder with pooling and local contrast normalization [117]. This model allows the system to train a face detector without having to label images as containing a face or not. The resulting feature detector is robust to translation, scaling and out-of-plane rotation.

2.2.3.2 Denoising Autoencoder

In order to increase the robustness of the model, Vincent et al. [29, 107] proposed a model called denoising autoencoder (DAE), which can recover the correct input from a corrupted version, thus forcing the model to capture the structure of the input distribution. The process of a DAE is shown in Figure 2.13.

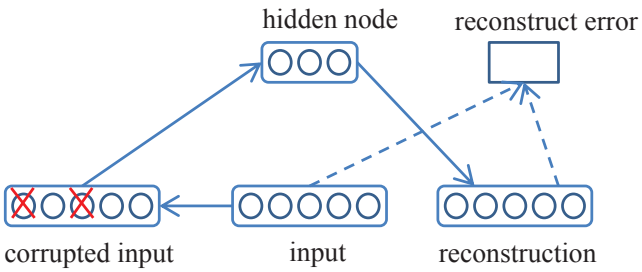


Figure 2.13: Denoising Autoencoder [29].

2.2.3.3 Contractive Autoencoder

Contractive Autoencoder (CAE), proposed by Rifai et al. [30], followed after the DAE and shared a similar motivation of learning robust representations [19]. While a DAE makes the whole mapping robust by injecting noise in the training set, a CAE achieves robustness by adding an analytic contractive penalty to the reconstruction error function.

Although the differences between DAE and CAE are stated by Bengio et al. [19], Alain et al. [118] still suggested that DAE and a form of CAE are closely related

to each other: a DAE with small corruption noise can be valued as a type of CAE where the contractive penalty is on the whole reconstruction function rather than just on the encoder. Both DAE and CAE have been successfully used in the Unsupervised and Transfer Learning Challenge [119].

2.2.4 Sparse Coding

Sparse coding intends to learn an over-complete set of basic functions to describe the input data [120], and it has numerous advantages [31, 33, 121, 122]: (1) It can reconstruct the descriptor better by using multiple bases and capturing the correlations between similar descriptors which share bases; (2) the sparsity allows the representation to capture salient properties of images; (3) it is in line with the biological visual system, which argues that sparse features of signals are useful for learning; (4) image statistics study shows that image patches are sparse signals; (5) patterns with sparse features are more linearly separable.

2.2.4.1 Solving the sparse coding equation

In this subsection, we will briefly describe how to solve the sparse coding problem. The general objective function of sparse coding is as below.

$$\min_{D} \frac{1}{T} \sum_{t=1}^{T} \min_{h^{(t)}} (\frac{1}{2} ||x^{(t)} - Dh^{(t)}||_2^2 + \lambda ||h^{(t)}||_1) \tag{2.2}$$

The first term of the function is the reconstruction error, while the second L1 regularization term is the sparsity penalty. The L1 norm regularization has been verified to lead to sparse representations [123]. Eq 2.2 can be solved with a regression method called LASSO (Least Absolute Shrinkage and Selection Operator). It cannot get the analytic solution of the sparse representation. Therefore, solving of the problem normally results in an intractable computation.

To optimize the sparse coding model, there is an alternating procedure between updating the weights and inferring the feature activations of the input given the current setting of the weights.

1. Weight update

One commonly used algorithm for updating the weights is called projected gradient algorithm [124], which renormalizes each column of the weight matrix right after each update of the traditional Gradient descent algorithm [125]. The normalization is necessary for the sparsity penalty to have any effect. However, gradient descent using iterative projections often shows slow convergence. In 2007, Lee et al. [126] derived a Lagrange dual method, which is much more efficient than gradient-based methods. Given a dictionary, the paper further proposed a feature-sign search algorithm to learn the sparse representation. The combination of these two algorithms enabled the performance to be significantly better than the previous ones. However, it cannot efficiently handle very large training sets, or dynamic training data that is changing over time. Thus it inherently accesses the whole training set at each iteration. To address this issue, an online approach [127, 128] was proposed for learning dictionaries that process one element (or a small subset) of the training set at a time. The algorithm then updates the dictionary using block-coordinate descent [129] with warm restarts, which does not require any learning rate tuning.

Gregor et al. [130] tried to accelerate the dictionary learning in another way: it imports the idea of Coordinate Descent algorithm (CoD) which only updates the "most promising" hidden units and therefore leads to dramatic reduction in the number of iterations to reach a given code prediction error.

2. Activation inference

Given a set of the weights, we need to infer the feature activations. A popular algorithm for sparse coding inference is the Iterative Shrinkage-Thresholding Algorithm (ISTA) [131], which takes a gradient step to optimize the reconstruction term, followed by a sparsity term which has a closed form shrinkage operation. Although simple and effective, the algorithm suffers from a severe problem that it converges quite slowly. The problem is partly solved by the Fast Iterative shrinkage-Thresholding Algorithm (FISTA) approach [132], which preserves the computational simplicity of ISTA, but converges more quickly due to the introduction of a 'momentum' term in the dynamics (the convergence complexity

changed from to). Both the ISTA and FISTA inference involve some sort of iterative optimization (i.e. LASSO), which is of high computational complexity. In contrast, Kavukcuoglu et al. [133] utilized a feed-forward network to approximate the sparse codes, which dramatically accelerated the inference process. Furthermore, the LASSO optimization stage was replaced by marginal regression in [134], effectively scaling up the sparse coding framework to large dictionaries.

2.2.4.2 Developments

As we have briefly stated how to generate the sparse representation given the objective function, in this subsection, we will introduce some well-known algorithms related to sparse coding, in particular those that are used in computer vision tasks. The well-known sparse coding algorithms and relations, along with their contributions and drawbacks are shown in Figure 2.14.

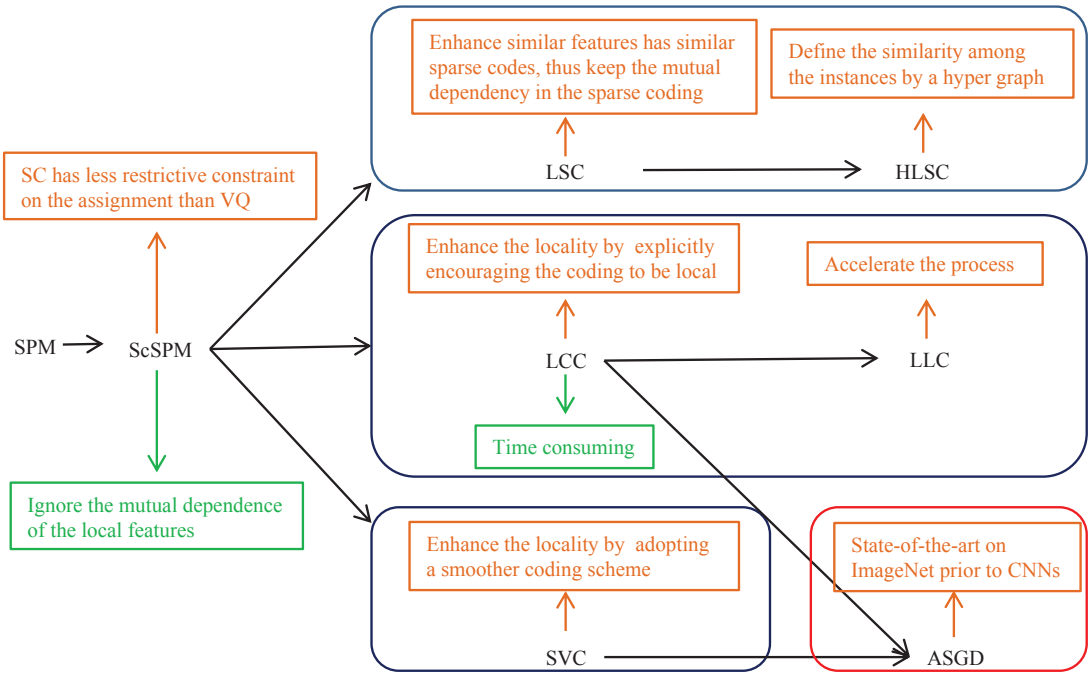
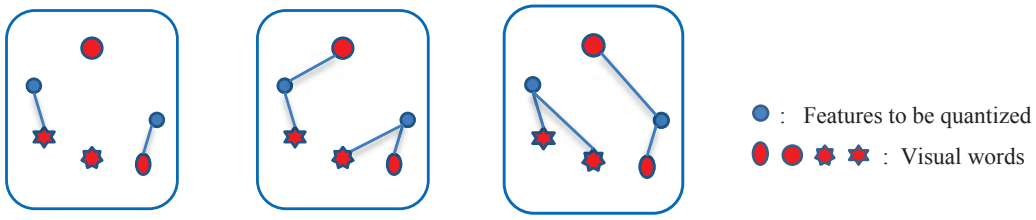


Figure 2.14: The well-known sparse coding algorithms, relations, contributions and drawbacks

One representative algorithm for sparse coding is Sparse coding SPM (ScSPM) [31], which is an extension of the Spatial Pyramid Matching (SPM) method [135]. Unlike the SPM, which uses vector quantization (VQ) for the image representation, ScSPM utilizes sparse coding (SC) followed by multi-scale spatial max pooling. The codebook of SC is an over-complete basis and each feature can activate a small number of them. Compared to VQ, SC receives a much lower reconstruction error due to the less restrictive constraint on the assignment. Coates et al. [136] further investigated the reasons for the success of SC over VQ in detail. A drawback of ScSPM is that it deals with local features separately, thus ignores the mutual dependence among them, which makes it too sensitive to feature variance, i.e. the sparse codes may vary a lot, even for similar features.

To address this problem, Gao et al. [32] proposed a Laplacian Sparse Coding (LSC) approach, in which similar features are not only assigned to optimally-selected cluster centers, but that also guarantees the selected cluster centers to be similar. The difference between K-means, Sparse Coding and Laplacian Sparse Coding is shown in Figure 2.15.



(a) K-means (b) Sparse Coding (c) Laplacian SC

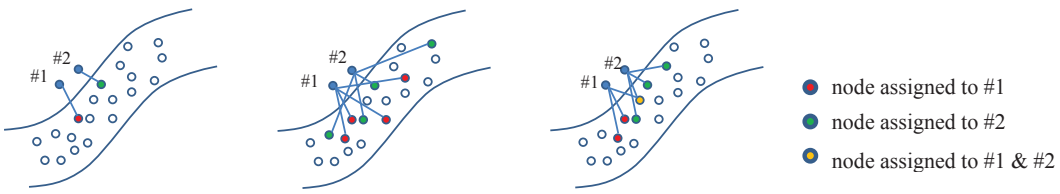
Figure 2.15: The difference between K-means, Sparse Coding and Laplacian Sparse Coding [32].

By adding the locality preserving constraint to the objective of sparse coding, the LSC can keep the mutual dependency in the sparse coding procedure. Gao et al. [137] further raised a Hyper-graph Laplacian Sparse Coding (HLSC) method, which extends the LSC to the case where the similarity among the instances is defined by a hyper graph. Both LSC and HLSC enhance the robustness of sparse coding.

Another way to address the sensitivity problem is the hierarchical sparse coding method proposed by Yu et al. [138]. It introduced a two-layer sparse coding model: the first layer encodes individual patches, and the second layer jointly encodes the set of patches that belong to the same group. Therefore, the model leverages the spatial neighborhood structure by modeling the higher-order dependency of patches in the same local region of an image. Besides that, it is a fully automatic method to learn features from the pixel level, rather than for example the hand-designed SIFT feature. The hierarchical sparse coding is utilized in another research [139] to learn features for images in an unsupervised fashion. The model is further improved by Zeiler et al. [140].

In addition to the sensitivity, another method exists for improving the ScSPM algorithm, by considering the locality. Yu et al. [33] observed that the ScSPM results tend to be local, i.e. nonzero coefficients are often assigned to bases nearby. As a result of these observations, they suggested a modification to ScSPM, called Local Coordinate Coding (LCC), which explicitly encourages the coding to be local. They also theoretically showed that locality is more important than sparsity. Experiments have shown that locality can enhance sparsity and that sparse coding is helpful for learning only when the codes are local, so it is preferred to let similar data have similar non-zero dimensions in their codes. Although LCC has a computational advantage over classic sparse coding, it still needs to solve the L1-norm optimization problem, which is time-consuming. To accelerate the learning process, a practical coding method called Locality-Constrained Linear Coding (LLC) was introduced [122], which can be seen as a fast implementation of LCC that replaces the L1-norm regularization with L2-norm regularization.

A comparison between VQ, ScSPM and LLC [122] are shown in Figure 2.16.



(a) VQ (b) ScSPM (c) LLC

Figure 2.16: A comparison between VQ, ScSPM, LLC [122].

Besides LLC, there is another model, called super-vector coding (SVC) [34], which can also guarantee local sparse coding. Given x, SVC will activate those coordinates associated to the neighborhood of x to achieve the sparse representation. SVC is a simple extension of VQ by expanding VQ in local tangent directions, and is thus a smoother coding scheme.

A remarkable result is shown in [141], in which the proposed averaging stochastic gradient descent (ASGD) scheme combined LCC and SVC algorithm to scale the image classification to large-scale dataset, and produced state-of-the-art results on ImageNet object recognition tasks prior to the rise of CNN architectures.

Another well-known smooth coding method is presented in [134], called Smooth Sparse Coding (SSC). The method incorporates the neighborhood similarity and temporal information into sparse coding, leading to codes that represent a neighborhood rather than an individual sample and that have lower mean square reconstruction error.

More recently, He et al. [142] proposed a new unsupervised feature learning framework, called Deep Sparse Coding (DeepSC), which extends sparse coding to a multi-layer architecture and has the best performance among the sparse coding schemes described above.

2.2.5 Discussion

In order to compare and understand the above four categories of deep learning, we summarize their advantages and disadvantages with respect to diverse properties, as listed in Table 2.4. There are nine properties in total. In details, 'Generalization' refers to whether the approach has been shown to be effective in diverse media (e.g. text, images, audio) and applications, including speech recognition, visual recognition and so on. 'Unsupervised learning' refers to the ability to learn a deep model without supervisory annotation. 'Feature learning' is the ability to automatically learn features based on a data set. 'Real-time training' and 'Real-time prediction' refer to the efficiency of the learning and inferring processes, respectively. 'Biological understanding' and 'Theoretical justification' represent

whether the approach has significant biological underpinnings or theoretical foundations, respectively. 'Invariance' refers to whether the approach has been shown to be robust to transformations such as rotation, scale and translation. 'Small training set' refers to the ability to learn a deep model using a small number of examples. It is important to note that the table only represents the general current findings and not future possibilities nor specialized niche cases.

Table 2.4: Comparisons among four categories of deep learning (Note: 'Yes' indicates that the category does well in the property; otherwise, they will be marked by 'No'. The 'Yes*' refers to a preliminary or weak ability)

Properties	CNNs	RBMs	AutoEncoder	Sparse Coding
Generalization	Yes	Yes	Yes	Yes
Unsupervised learning	No	Yes	Yes	Yes
Feature learning	Yes	Yes*	Yes*	No
Real-time training	No	No	Yes	Yes
Real-time prediction	Yes	Yes	Yes	Yes
Biological understanding	No	No	No	Yes
Theoretical justification	Yes*	Yes	Yes	Yes
Invariance	Yes*	No	No	Yes
Small training set	Yes*	Yes*	Yes	Yes

2.3 Applications and Results

Deep learning has been widely adopted in various directions of computer vision, such as image classification, object detection, image retrieval, semantic segmentation, and human pose estimation, which are key tasks for image understanding. In this part, we will briefly summarize the developments of deep learning (all of the results are referred from the original papers), especially the CNN based algorithms, in these five areas.

2.3.1 Image Classification

The image classification task consists of labeling input images with a probability of the presence of a particular visual object class [143], as is shown in Figure 2.17.

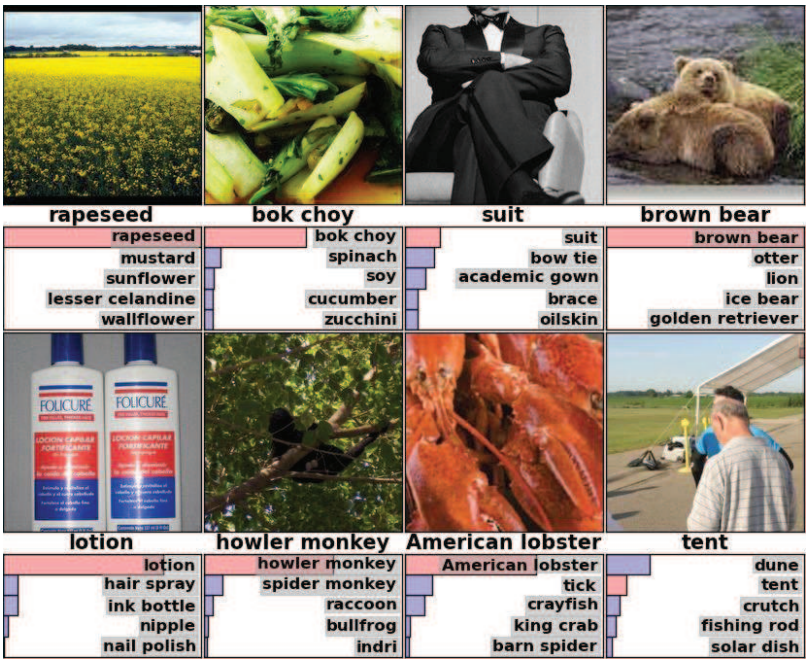


Figure 2.17: Image classification examples from AlexNet [14]. Each image has one ground truth label, followed by the top 5 guesses with probabilities.

Prior to deep learning, perhaps the most commonly used methods in image classification were methods based on bags of visual words (BoW) [144], which first describes the image as a histogram of quantized visual words, and then feeds the histogram into a classifier (typically an SVM [145]). This pipeline was based on the orderless statistics, to incorporate spatial geometry into the BoW descriptors. Lazebnik et al. [135] integrated a spatial pyramid approach into the pipeline, which counts the number of visual words inside a set of image sub-regions instead of the whole region. Thereafter, this pipeline was further improved by importing sparse coding optimization to the building of codebooks [141], which receives the

best performance on the ImageNet 1000-class classification in 2010. Sparse coding is one of the basic algorithms in deep learning, and it is more discriminative than the original hand-designed ones, e.g. HOG [141] and LBP [146].

The approaches based on BoW just concern the zero order statistics (i.e. counts of visual words), discarding a lot of valuable information of the image [143]. The method introduced by Perronnin et al. [147] overcame this issue and extracted higher order statistics by employing the Fisher Kernel [148], achieving the state-of-the-art image classification result in 2011.

Krizhevsky et al. [14] represented a turning point for large-scale object recognition when a large CNN was trained on the ImageNet database [149], thus proving that CNN could, in addition to handwritten digit recognition [35], perform well on natural image classification. The proposed AlexNet won the ILSVRC 2012, with a top-5 error rate of 15.3%, which sparked significant additional activity in CNN research. In Figure 2.18, we present the state-of-the-art results on the ImageNet test dataset since 2012, along with the pipeline of ILSVRC.

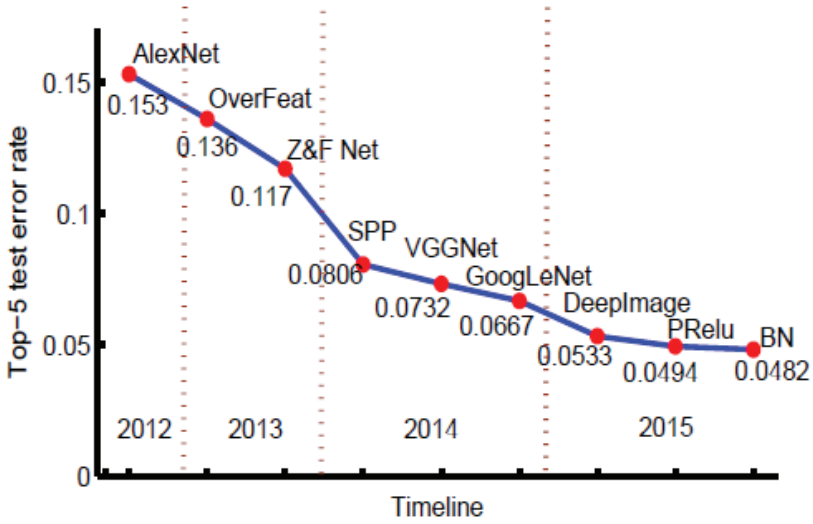


Figure 2.18: ImageNet classification results on test dataset.

OverFeat [150] proposed a multiscale and sliding window approach, which could find the optimal scale of the image and fulfill different tasks simultaneously, i.e.

classification, localization and detection. Specifically, the algorithm decreased the top-5 test error to 13.6%. Zeiler et al. [22] introduced a novel visualization technique to give insight into the function of intermediate feature layers and further adjusted a new model, which outperformed AlexNet, reaching 11.7% top-5 error rate, and had top performance at ILSVRC 2013.

ILSVRC 2014 witnessed the steep growth of deep learning, as most participants utilized CNNs as the basis for their models. Again significant progress had been made in image classification, as the error was almost halved since ILSVRC2013. The SPP-net [23] model eliminated the restriction of the fixed input image size and could boost the accuracy of a variety of published CNN architectures despite their different designs. Multiple SPP-nets further reduced the top-5 error rate to 8.06% and ranked third in the image classification challenge of ILSVRC 2014. Along with the improvements of the classic CNN model, another characteristic shared by the top-performing models is that the architectures became deeper, as shown by GoogLeNet [25] (rank 1 in ILSVRC 2014) and VGG [24] (rank 2 in ILSVRC 2014), which achieved 6.67% and 7.32% respectively.

Despite the potential capacity possessed by larger models, they also suffered from overfitting and underfitting problems when there is little training data or little training time. To avoid this shortcoming, Wu et al. [57] developed new strategies, i.e. DeepImage, for data augmentation and usage of multi-scale images. They also built a large supercomputer for deep neural networks and developed a highly optimized parallel algorithm, and the classification result achieved a relative 20% improvement over the previous one with a top-5 error rate of 5.33%. More Recently, He et al. [9] proposed the Parametric Rectified Linear Unit to generate the traditional rectified activation units and derived a robust initialization method. This scheme led to 4.94% top-5 test error and surpassed human-level performance (5.1%) for the first time. Similar results were achieved by Ioffe et al. [151], whose method reached a 4.8% test error by utilizing an ensemble of batch-normalized networks.

2.3.2 Object Detection

Object detection is different from but closely related to an image classification task. For image classification, the whole image is utilized as the input and the class label of objects within the image are estimated. For object detection, besides outputting the information of the presence of a given class, we also need to estimate the position of the instance (or instances), as shown in Figure 2.19. A detection window is regarded as correct if the outputted bounding box has sufficiently large overlap with the ground truth object (usually more than 50%).

Figure 2.19: Object detection examples from RCNN [44]. The red box extracts the salient objects contains, the green box contains the prediction score.

The challenging PASCAL VOC datasets are the most widely employed for the evaluation of object detection. There are twenty classes in this database. During the test phase, an algorithm should predict the bounding boxes of the objects belong to each class in a test image. In this section, we will describe the recent developments of deep learning schemes for object detection, according to their achievements in VOC 2007 and VOC 2012. The related advances are shown in Table 2.5.

Before the surge of deep learning, the Deformable Part Model (DPM) [152] was the most effective method for object detection. It takes advantage of deformable part models and detects objects across all scales and locations on the image in an exhaustive manner. After integrating with some post-processing techniques, i.e. bounding box prediction and context rescoring, the model achieved 29.09% average precision for VOC 2007 test set.

Table 2.5: Object detection results of the VOC 2007 and VOC 2012 challenges

Methods	Training data	VOC2007 (mAP)		VOC2012(mAP)	
		A/C-Net	VGG-Net	Alex-Net	VGG-Net
DPM [152]	07	29.09%	-	-	-
DetectorNet [153]	12	30.41%	-	-	-
DeepMultiBox [154]	12	29.22%	-	-	-
RCNN [44]	07	54.2%	62.2%	-	-
RCNN [44] + BB	07	58.5%	66%	-	-
RCNN [44]	12	-	-	49.6%	59.2%
RCNN [44] + BB	12	-	-	53.3%	62.4%
SPP-Net [23]	07	55.2%	60.4%	-	-
SPP-Net [23]	07+12	-	64.6%	-	-
SPP-Net [23] + BB	07	59.2%	63.1%	-	-
FRCN [64]	07	-	66.9%	-	65.7%
FRCN [64]	07++12	-	70.0%	-	68.4%
RPN [65]	07	59.9%	69.9%	-	-
RPN [65]	12	-	-	-	67%
RPN [65]	07+12	-	73.2%	-	-
RPN [65]	07++12	-	-	-	70.4%
MR_CNN [67]	07	-	74.9%	-	69.1%
MR_CNN [67]	12	-	-	-	70.7%
FGS [69]	07	-	66.5%	-	-
FGS [69] + BB	07	-	68.5%	-	66.4%
NoC [155]	07+12	62.9%	71.8%	-	67.6%
NoC [155] + BB	07+12	-	73.3%	-	68.8%

Note: Training data: "07": VOC07 trainval; "12": VOC2 trainval; "07+12": VOC07 trainval union with VOC12 trainval; "07++12": VOC07 trainval and test union with VOC12 trainval; BB: bounding box regression; A/C-Net: approaches based on AlexNet[6] or Clarifai [52]; VGG-Net: approaches based on VGG-Net[31]

As deep learning methods (especially the CNN-based methods) had achieved top tier performance on image classification tasks, researchers started to transfer it to the object detection problem. An early deep learning approach for object detection was introduced by Szegedy et al. [153]. The paper proposed an algorithm,

called DetectorNet, which replaced the last layer of AlexNet [14] with a regression layer. The algorithm captured object location well and achieved competitive results on the VOC2007 test set with the most advanced algorithms at that time. To handle multiple instances of the same object in the image, DeepMultiBox [154] also showed a saliency-inspired neural network model.

A general pattern for current successful object detection systems is to generate a large pool of candidate boxes and classify those using CNN features. The most representative approach is the RCNN scheme proposed by Girshick et al. [44]. It utilizes selective search [156] to generate object proposals, and extracts the CNN features for each proposal. The features are then fed into an SVM classifier to decide whether the related candidate windows contain the object or not. RCNNs improved the benchmark by a large margin, and became the base model for many other promising algorithms [64–66, 68, 69].

The algorithms derived from RCNNs are mainly divided into two categories: the first category aims to accelerate the training and testing process. Although an RCNN has excellent object detection accuracy, it is computationally intensive because it first warps and then processes each object proposal independently. Consequently, some well-known algorithms which aim to improve its efficiency appeared, such as SPP-net [23], FRCN [64], RPN [65], YOLO [157], etc. These algorithms detect objects faster, while achieving comparable mAP with state-of-the-art benchmarks.

The second category is mainly intended to improve the accuracy of RCNNs. The performance of the 'recognition using regions' paradigm is highly dependent on the quality of object hypotheses. Currently, there are many object proposal algorithms, such as objectness [158], selective search [156], category-independent object proposals [159], BING [160], and edge boxes [161]. These schemes are exhaustively evaluated in [162]. Although those schemes are good at finding rough object positions, they normally could not accurately localize the whole object via a tight bounding box, which forms the largest source of detection error [163, 164]. Therefore, many approaches have emerged that try to correct the poor localizations.

One important direction of these methods is to combine them with semantic segmentation techniques [66, 68, 165]. For example, the SDS scheme proposed by Hariharan et al. [68] utilizes segmentation to mask-out the background inside the detection, resulting in improved performance for object detection (from 49.6% to 50.7%, both without bounding box regression). On the other hand, the UDS method [165] unified the object detection and semantic segmentation process in one framework, by enforcing their consistency and integrating context information, the model demonstrated encouraging performance on both tasks. Similar works come with segDeepM proposed by Zhu et al. [66] and MR_CNN in [67], which also incorporate the segmentation along with additional evidence to boost the accuracy of object detection.

There are also approaches which attempt to precisely locate the object in other ways. For instance, FGS [69] addresses the localization problem via two methods: 1) develop a fine-grained search algorithm to iteratively optimize the location; 2) train a CNN classifier with a structured SVM objective to balance between classification and localization. The combination of these methods demonstrates promising performance on two challenging datasets.

Aside from the efforts in object localization, the NoC framework in [155] tries to evolve efforts in the object classification step. In place of the commonly used multi-layer perceptron (MLP), it explored different NoC structures to implement the object classifiers.

It is much cheaper and easier to collect a large amount of image-level labels than it is to collect detection data and label it with precise bounding boxes. Therefore, a major challenge in scaling the object detection is the difficulty of obtaining labeled images for large numbers of categories [166, 167]. Hoffman et al. [166] proposed a Deep Detection Adaption (DDA) algorithm to learn the difference between image classification and object detection, transferring classifiers for categories into detectors, without bounding box annotated data. The method has the potential to enable the detection for thousands of categories which lack bounding box annotations.

Two other promising, scalable approaches are ConceptLeaner [168] and BabyLearning [169].Both of them can learn accurate concept detectors but without the massive annotation of visual concepts. As collecting weakly labeled images is cheap, ConceptLeaner [168] develops a max-margin hard instance learning algorithm to automatically discover visual concepts from noisy labeled image collections. As a result, it has the potential to learn concepts directly from the web. On the other hand, the BabyLearning [169] approach simulates a baby's interaction with the physical world, and can achieve comparable results with state-of-the-art full-training based approaches with only few samples for each object category, along with large amounts of online unlabeled videos.

From Table 2.5, we can also observe several factors that could improve the performance, in addition to the algorithm itself: 1) larger training set; 2) deeper base model; 3) Bounding Box regression.

2.3.3 Image Retrieval

Image retrieval aims to find images containing a similar object or scene as in the query image, as illustrated in Figure 2.20.

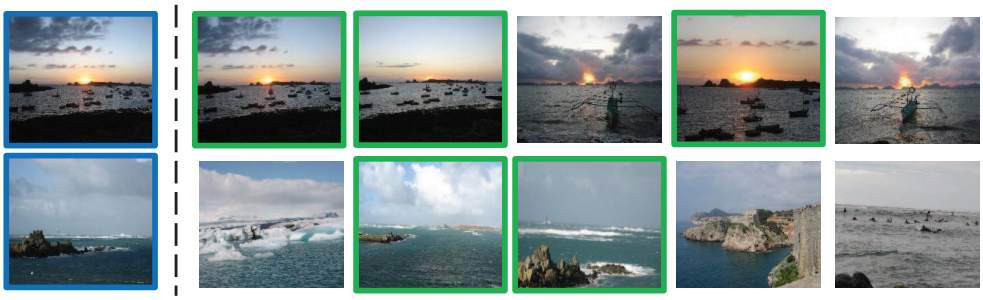


Figure 2.20: Image retrieval examples using CNN features. The left images are querying ones, and the images with green frames in the right represent the positive retrieval candidates.

The success of AlexNet [14] suggests that the features emerging in the upper layers of the CNN learned to classify images can serve as good descriptors for

image classification. Motivated by this, many recent studies use CNN models for image retrieval tasks [61, 170–173]. These studies achieved competitive results compared with the traditional methods, such as VLAD and Fisher Vector. In the following paragraphs, we will introduce the main ideas of these CNN based methods.

Inspired by Spatial Pyramid Matching, Gong et al. [170] proposed a kind of 'reverse SPM' idea that extracts patches at multiple scales, starting with the whole image, and then pool each scale without regard to spatial information. Then it aggregates local patch responses at the finer scales via VLAD encoding. The orderless nature of VLAD helps to build a more invariant representation. Finally, the original global deep activations are concatenated with the VLAD features for the finer scales to form the new image representation.

Razavian et al. [171] used features extracted from the OverFeat network as a generic image representation to tackle the diverse range of vision tasks, including recognition and retrieval. First, it augments the training set by adding cropped and rotated samples. Then for each image, it extracts multiple sub-patches of different sizes at different locations. Each sub-patch is computed for its CNN presentation. The distance between the reference and the query image is set to the average distance of each query sub-patch to the reference image.

Given the recent successes that deep learning techniques have achieved, the research presented in [61] attempts to evaluate if deep learning can bridge the semantic gap in content-based image retrieval (CBIR). Their encouraging results reveal that deep CNN models pre-trained on large datasets can be directly used for feature extraction in new CBIR tasks. When being applied for feature representation in a new domain, it was found that similarity learning can further boost the retrieval performance. Further, by retraining the deep models with a classification or similarity learning objective on the new domain, the accuracy can be improved significantly.

A different approach shown in [172] is to first extract object-like image patches with a general object detector. Then, one CNN feature is extracted in each object patch with the pre-trained AlexNet model. With many results from their

experiments, it is concluded that their method can achieve a significant accuracy improvement with the same space consumption, and with the same time cost it still obtains a higher accuracy.

Finally, without sliding windows or multiple-scale patches, Babenko et al. [173] focus on holistic descriptors where the whole image is mapped to a single vector with a CNN model. It found that the best performance is observed not at the very top of the network, but rather at the layer that is two levels below the outputs. An important result is that PCA affects the performance of the CNN much less than the performance of VLADs or Fisher Vectors. Therefore PCA compression works better for CNN features. In Table 2.6, we show the retrieval results in several public datasets.

Table 2.6: Image retrieval results on several datasets

Methods	Holidays	Paris6K	Oxford5K	UKB
Babenko et al. [173]	74.7	-	55.7	3.43
SUN et al. [172]	79.0	-	-	3.61
Gong et al. [170]	80.2	-	-	-
Razavian et al. [171]	84.3	79.50	68.0	-(91.1)
Wan et al. [61]	-	94.7	78.3	-

There is one more interesting problem in CNN features: which layer has the highest impact on the final performance? Some methods extract features in the second fully connected layer [170, 172]. In contrast to them, other methods use the first fully connected layer in their CNN model for image representation [171, 173]. Moreover, these choices may change for different datasets [61]. Thus, we think investigating the characteristics of each layer is still an open problem.

2.3.4 Semantic Segmentation

In recent years, a large number of studies focus on the semantic segmentation task, and yield promising progress. The main reason of their success comes from CNN models, which are capable of tackling the pixel-level predictions with the

pre-trained networks on large-scale datasets. Different from image-level classification and object-level detection, semantic segmentation requires output masks that have a 2D spatial distribution. As for semantic segmentation, recent and advanced CNN based methods can be summarized as follows:

(1) Detection-based segmentation. The approach segments images based on the candidate windows outputted from object detection [44, 174–176]. RCNN [44] and SDS [68] first generated region proposals for object detection, and then utilized traditional approaches to segment the region and to assign the pixels with the class label from detection. Based on SDS [68], Hariharan et al. [176] proposed the hyper-column at each pixel as the vector of activations, and gained large improvement. One disadvantage of detection-based segmentation is the largely additional expense for object detection. Without extracting regions from raw images, Dai et al. [175] designed a convolutional feature masking (CFM) method to extract proposals directly from the feature maps, which is efficient as the convolutional feature maps only need to be computed once. Even though, the errors caused by proposals and object detection tend to be propagated to the segmentation stage.

(2) FCN-CRFs based segmentation. In the second one, fully convolutional networks(FCN), replacing the fully connected layers with more convolutional layers, has been a popular strategy and baseline for semantic segmentation [60, 174]. Long et al. [60] defined a novel architecture that combined semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. DeepLab [174] proposed a similar FCN model, but also integrated the strength of conditional random fields (CRFs) into FCN for detailed boundary recovery. Instead of using CRFs as a post-processing step, Lin et al. [177] jointly trains the FCN and CRFs by efficient piecewise training. Likewise, the work in [178] converted the CRFs as a recurrent neural network (RNN), which can be plugged in as a part of FCN model.

(3) Weakly supervised annotations. Apart from the advancements in segmentation models, some works are focused on weakly supervised segmentation. Papandreou et al. [179] studied the more challenging segmentation with weakly annotated training data such as bounding boxes or image-level labels. Likewise,

the BoxSup method in [180] made use of bounding box annotations to estimate segmentation masks, which are used to update network iteratively. These works both showed excellent performance when combining a small number of pixel-level annotated images with a large number of bounding box annotated images.

We compare their results on PASCAL VOC 2012 val and test set in Table 2.7.

Table 2.7: Semantic segmentation results on PASCAL VOC 2012 val and test set

Methods	Train	Val 2012	Test 2012
SDS [68]	VOC extra	53.9	51.6
CFM [175]	VOC extra	60.9	61.8
FCN-8s [60]	VOC extra	-	62.2
Hypercolumn [176]	VOC extra	59	62.6
DeepLab [174]	VOC extra	63.7	66.4
DeepLab-MSc-LargeFOV [174]	VOC extra	68.7	71.6
Piecewise-DCRFs [177]	VOC extra	70.3	70.7
CRF-RNN [178]	VOC extra	69.6	72.0
BoxSup [180]	VOC extra+COCO	68.2	71.0
Cross-Joint [179]	VOC extra+COCO	71.7	73.9

2.3.5 Human Pose Estimation

Human pose estimation aims to estimate the localization of human joints from still images or image sequences, as shown in Figure 2.21.. It is very important for a wide range of potential applications, such as video surveillance, human behavior analysis, human-computer interaction (HCI), and is being extensively studied recently [181–191]. However, this task is also very challenging because of the great variation of human appearances, complicated backgrounds, as well as many other nuisance factors, such as illumination, viewpoint, scale, etc. In this part, we mainly summarize deep learning schemes to estimate the human articulation from still images, although these schemes could be incorporated with motion features to further boost their performance in videos [181–183].

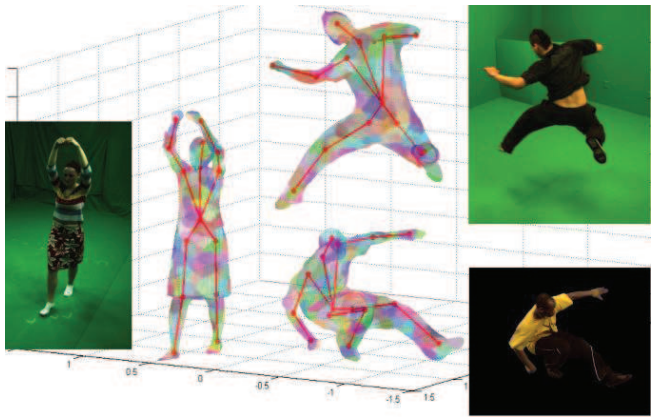


Figure 2.21: Human pose estimation [192].

Normally, human pose estimation involves multiple problems such as recognizing people in images, detecting and describing human body parts, and modeling their spatial configuration. Prior to deep learning, the best performing human pose estimation methods were based on body part detectors, i.e. detect and describe the human body part first, and then impose the contextual relations between local parts. One typical part-based approach is pictorial structures [193], which takes advantage of a tree model to capture the geometric relations between adjacent parts and has been developed by various well-known part-based methods [194–197].

As deep learning algorithms can learn high-level features which are more tolerant to the variations of nuisance factors, and have achieved success in various computer vision tasks, they have recently received significant attention from the research community.

We have summarized the performance of related deep learning algorithms on two commonly used datasets: Frames Labeled In Cinema (FLIC) [198] and Leeds Sports Pose (LSP) [199]. FLIC consists of 3987 training images and 1016 test images obtained from popular Hollywood movies, containing people in diverse poses, annotated with upper-body joint labels. LSP and its extension contains 11000 training and 1000 testing images of sports people gathered from Flickr with 14 full body joints annotated. There are two widely accepted evaluation metrics

for the evaluation: Percentage of Correct Parts (PCP) [200], which measures the rate of correct limb detection, and Percent of Detected Joints (PDJ) [198], which measures the rate of correct limb detection.

In the following, Table 2.8 illustrates the PDJ comparison of various deep learning methods on FLIC dataset, with a normalized distance of 0.05, and Table 2.9 lists out the PCP comparison on LSP dataset.

Table 2.8: The PDJ comparison on FLIC dataset

PDJ(PCK)	Head	Shoulder	Elbow	Wrist
Jain et al. [186]	-	42.6	24.1	22.3
DeepPose [201]	-	-	25.2	26.4
Chen et al. [185]	-	-	36.5	41.2
DS-CNN [190]	-	-	30.5	36.5
Tompson et al. [187]	90.7	70.4	50.2	55.4
Tompson et al. [188]	92.6	73	57.1	60.4

Table 2.9: The PCP comparison on LSP dataset

	Torso	Head	U.arms	L.arms	U.legs	L.legs	Mean
Ouyang et al. [189]	85.8	83.1	63.3	46.6	76.5	72.2	68.6
DeepPose [201]	-	-	56	38	77	71	-
Chen et al. [185]	92.7	87.8	69.2	55.4	82.9	77	75
DS-CNN [190]	98	85	80	63	90	88	84

In general, deep learning schemes in human pose estimation can be categorized according to the handling manner of input images: holistic processing or part-based processing.

The holistic processing methods tend to accomplish their task in a global manner, and do not explicitly define a model for each individual part and their spatial relationships. One typical model is called DeepPose proposed by Toshev et al. [201]. This model formulates the human pose estimation method as a joint regression problem and does not explicitly define the graphical model or part detectors for the human pose estimation. More specifically, it utilizes a two-layer architecture:

the first layer addresses the ambiguity between body parts in a holistic way and generates the initial pose estimation. The second layer refines the joint locations for the estimation. This model achieved advances on several challenging datasets. However, the holistic-based method suffers from inaccuracy in the high-precision region, since it is difficult to learn direct regression of complex pose vectors from images.

The part-based processing methods propose to detect the human body parts individually, followed with a graphic model to incorporate the spatial information. Instead of training the network using the whole image, Chen et al. [185] utilized the local part patches and background patches to train a DCNN, in order to learn conditional probabilities of the part presence and spatial relationships. By incorporating with graphic models, the algorithm gained promising performance. Moreover, Jain et al. [186] trained multiple smaller convnets to perform independent binary body-part classification, followed with a higher-level weak spatial model to remove strong outliers and to enforce global pose consistency. Similarly, Tompson et al. [187] designed multi-resolution ConvNet architectures to perform heat-map likelihood regression for each body part, followed with an implicit graphic model to further promote joint consistency. The model was further extended in [188], which argues that the pooling layers in the CNNs would limit spatial localization accuracy and try to recover the precision loss of the pooling process. They especially improve the method from [187] by adding a carefully designed Spatial Dropout layer, and present a novel network which reuses hidden-layer convolutional features to improve the precision of the spatial locality.

There are also approaches which suggesting combining both the local part appearance and the holistic view of the parts for more accurate human pose estimation. For example, Ouyang et al. [189] derived a multi-source deep model from a Deep Belief Net (DBN), which attempts to take advantage of three information sources of human articulation, i.e. mixture type, appearance score and deformation, and combine their high-level representations to learn holistic, high-order human body articulation patterns. On the other hand, Fan et al. [190] proposed a dual-source convolutional neutral network (DS-CNN) to integrate the holistic and partial view in the CNN framework. It takes part patches and body patches

as inputs to combine both local and contextual information for more accurate pose estimation.

As most of the schemes tend to design new feed-forward architectures, Carreira et al. [191] introduced a self-correcting model, called Iterative Error Feedback (IEF). This model can encompass rich structure in both input and output spaces by incorporating top-down feedback, and shows promising results.

2.4 Trends and Challenges

Along with the promising performance deep learning has achieved, the research literature has indicated several important challenges as well as the inherent trends, which are described next.

2.4.1 Theoretical Understanding

Although promising results in addressing computer vision tasks have been achieved by deep learning methods, the underlying theory is not well understood, and there is no clear understanding of which architectures should perform better than others. It is difficult to determine which structure, how many layers, or how many nodes in each layer are proper for a certain task, and it also need specific knowledge to choose sensible values such as the learning rate, the strength of the regularizer, etc. The design of the architecture has historically been determined on an ad-hoc basis. Chu et al. [202] proposed a theoretical method for determining the optimal number of feature maps. However, this theoretical method only worked for extremely small receptive fields. To better understand the behavior of the well-known CNN architectures, Zeiler et al. [22] developed a visualization technique that gave insight into the function of intermediate feature layers. By revealing the features in interpretable patterns, it brought further possibilities for better architecture designs. A similar visualization was also studied by Yu et al. [203].

Apart from visualizing the features, RCNN [44] attempted to discover the learning pattern of CNN. It tested the performance in a layer-by-layer pattern during the training process, and found that the convolutional layers can learn more general features and convey most of the CNN representational capacity, while the top fully-connected layers are domain-specific. In addition to analyzing the CNN features, Agrawal et al. [204] further investigated the effects of some commonly used strategies on CNN performance, such as fine-tuning and pre-training, and provided evidence-backed intuitions to apply CNN models to computer vision problems.

Despite the progress achieved in the theory of deep learning, there is significant room for better understanding in evolving and optimizing the CNN architectures toward improving desirable properties such as invariance and class discrimination.

2.4.2 Human-level Vision

Human vision has a remarkable proficiency in computer vision tasks, even in simple visual representations or under changes to geometric transformations, background variation, and occlusion. Human-level vision can refer to either bridging the semantic gap in terms of accuracy or in bringing new insights from studies of the human brain to be integrated into machine learning architectures. Compared with the traditional low-level features, a CNN mimics human brain structure and builds multi-layers activations for mid-level or high-level features. The study in [61] aimed to evaluate how much retrieval improvement can be achieved by developing deep learning techniques, and whether deep features are a desirable key to bridge the semantic gap in the long term. As seen in Figure 2.18, the image classification error on the ImageNet test set decreases 10%, from 15.3% [14] in 2012 to 4.82% [151] in 2015. This promising improvement verifies the efficiency of CNNs. In particular, the result in [151] has exceeded the accuracy of human raters. However, we cannot conclude that the representational performance of a CNN rivals that of the brain [205]. For example, it is easy to produce images

that are completely unrecognizable to humans, but one state-of-the-art CNN believes it to contain recognizable objects with 99.99% confidence [206]. This result highlights the difference between human vision and current CNN models, and raises questions about the generality of CNNs in computer vision. The study in [205] found that, like the IT cortex, recent CNNs could generate similar feature spaces for the same category, and distinct ones for images with different categories. This result indicates that CNNs may provide insight into understanding primate visual processing. In another study [207], the authors considered a novel approach for brain decoding for fMRI data by leveraging unlabeled data and multi-layer temporal CNNs, which learned multiple layers of temporal filters and trained powerful brain decoding models. Whether CNN models that rely on computational mechanisms are similar to the primate visual system is yet to be determined, but it has the potential for further improvements by mimicking and incorporating the primate visual system.

2.4.3 Training with limited data

Larger models demonstrate more potential capacity and have become the tendency of recent developments. However, the shortage of training data may limit the size and learning ability of such models, especially when it is expensive to obtain fully labeled data. How to overcome the need for enormous amounts of training data and how to train large networks effectively remains to be addressed.

Currently, there are two commonly used solutions to obtain more training data. The first solution is to generalize more training data from existing data based on various data augmentation schemes, such as scaling, rotating and cropping. On top of these, Wu et al. [57] further adopted color casting, vignetting and lens distortion techniques, which could produce much more converted training examples with broad coverage. The second solution is to collect more training data with weak learning algorithms. Recently, there has been a range of articles on learning visual concepts from image search engines [208, 209]. In order to scale up computer vision recognition systems, Zhou et al. [168] proposed the ConceptLearner

approach, which could automatically learn thousands of visual concept detectors from weakly labeled image collections. Besides that, to reduce laborious bounding box annotation costs for object detection, many weakly-supervised approaches have emerged with image-level object-presence labeling [210]. Nevertheless, it is promising to further develop techniques for generating or collecting more comprehensive training data, which could make the networks learn better features that are robust under various changes, such as geometric transformations, and occlusion.

2.4.4 Time complexity

The early CNNs were seen as a method that required a lot of computational resources and were not candidates for real-time applications. One of the trends is towards developing new architectures which allow running a CNN in real-time. The study in [59] conducted a series of experiments under constrained time cost, and proposed models that are fast for real-world applications, yet are competitive with existing CNN models. In addition, fixing the time complexity also helps to understand the impacts of factors such as depth, numbers of filters, filter sizes, etc. Another study [211] eliminated all the redundant computations in the forward and backward propagation in CNNs, which resulted in a speedup of over 1500 times. It has robust flexibility for various CNN models with different designs and structures, and reaches high efficiency because of its GPU implementation. Ren et al. [212] converted the key operators in deep CNNs to vectorized forms, so that high parallelism can be achieved given basic parallelized matrix-vector operators. They further provided a unified framework for both high-level and low-level vision applications.

2.4.5 More Powerful Models

As deep learning related algorithms have moved forward the-state-of-the-art results of various computer vision tasks by a large margin, it becomes more chal-

lenging to make progress on top of that. There might be several directions for more powerful models:

The first direction is to increase the generalization ability by increasing the size of the networks [24, 25]. Larger networks could normally bring higher quality performance, but care should be taken to address the issues this may cause, such as overfitting and the need for a lot of computational resources.

A second direction is to combine the information from multiple sources. Feature fusion has long been popular and appealing, and this fusion can be categorized in two types. 1) Combine the features of each layer in the network. Different layers may learn different features [44]. It is promising if we could develop an algorithm to make the features from each layer to be complementary. For example, DeepIndex [213] proposed to integrate multiple CNN features by multiple inverted indices, including different layers in one model or several layers from distinct models. 2) Combine the features of different types. We can obtain more comprehensive models by integrating with other type of features, such as SIFT. To improve the image retrieval performance, DeepEmbedding [214] used the SIFT features to build an inverted index structure, and extracted the CNN features from the local patches to enhance the matching strength.

A third direction towards more powerful models is to design more specific deep networks. Currently, almost all of the CNN-based schemes adopt a shared network for their predictions, which may not be distinctive enough. A promising direction is to train a more specific deep network, i.e. we should focus more on type of object we are interested in. The study in [43] has verified that object-level annotation is more useful than image-level annotation for object detection. This can be viewed as a kind of specific deep network which just focuses on the object rather than the whole image. Another possible solution is to train different networks for different categories. For instance, [215] built on the intuition that not all classes are equally difficult to distinguish from a true class label, and designed an initial coarse classifier CNN as well as several fine CNNs. By adopting a coarse-to-fine classification strategy, it achieves state-of-the-art performance on CIFAR100.

2.5 Conclusion

This chapter presents a comprehensive review of deep learning and develops a categorization scheme to analyze the existing deep learning literature. It divides the deep learning algorithms into four categories according to the basic model they derived from: Convolutional Neural Networks, Restricted Boltzmann Machines, Autoencoder and Sparse Coding. The state-of-the-art approaches of the four classes are discussed and analyzed in detail. For the applications in the computer vision domain, the chapter mainly reports the advancements of CNN based schemes, as it is the most extensively utilized and most suitable for images. Most notably, some recent articles have reported inspiring advances showing that some CNN-based algorithms have already exceeded the accuracy of human raters.

Despite the promising results reported so far, there is significant room for further advances. For example, the underlying theoretical foundation does not yet explain under what conditions they will perform well or outperform other approaches, and how to determine the optimal structure for a certain task. This chapter describes these challenges and summarizes the new trends in designing and training deep neural networks, along with several directions that may be further explored in the future.