CHAPTER

An Introduction to Deep Convolutional Neural Nets for Computer Vision

2

Suraj Srinivas, Ravi K. Sarvadevabhatla, Konda R. Mopuri, Nikita Prabhu, Srinivas S.S. Kruthiventi, R. Venkatesh Babu

Indian Institute of Science, Bangalore, India

CHAPTER OUTLINE

2.1 Introduction 26
2.2 Convolutional Neural Networks 27
2.2.1 Building Blocks of CNNs 27
2.2.1.1 Why Convolutions? 27
2.2.1.2 Max-Pooling 29
2.2.1.3 Nonlinearity 29
2.2.1.4 Fully-Connected Layers 29
2.2.2 Depth 29
2.2.3 Learning Algorithm 30
2.2.3.1 Gradient-Based Optimization 30
2.2.3.2 Dropout 31
2.2.3.3 Batch Normalization 31
2.2.4 Tricks to Increase Performance 31
2.2.5 Putting It All Together: AlexNet 32
2.2.6 Using Pre-Trained CNNs 32
2.2.6.1 Fine-Tuning 32
2.2.6.2 CNN Activations as Features 33
2.2.7 Improving AlexNet 33
2.3 CNN Flavors 34
2.3.1 Region-Based CNNs 34
2.3.2 Fully Convolutional Networks 35
2.3.3 Multi-Modal Networks 38
2.3.4 CNNs with RNNs 40
2.3.4.1 Action Recognition 41
2.3.4.2 Image and Video Captioning 42
2.3.4.3 Visual Question Answering 42
2.3.5 Hybrid Learning Methods 43

Deep Learning for Medical Image Analysis. DOI: 10.1016/B978-0-12-810408-8.00003-1

2.3.5.1 Multi-Task Learning 43
2.3.5.2 Similarity Learning 44
2.4 Software for Deep Learning 45
References 46

2.1 INTRODUCTION

Computer vision problems like image classification and object detection have traditionally been approached using hand-engineered features like SIFT [63] and HoG [19]. Representations based on the bag-of-visual-words descriptor [110], in particular, enjoyed success in image classification. These were usually followed by learning algorithms like Support Vector Machines (SVMs). As a result, the performance of these algorithms crucially relied on the features used. This meant that progress in computer vision was based on hand-engineering better sets of features. With time, these features started becoming more and more complex, resulting in a difficulty of coming up with better, more complex features. From the perspective of the computer vision practitioner, there were two steps to be followed: feature design and learning algorithm design, both of which were largely independent.

Meanwhile, some researchers in the machine learning community had been working on learning models which incorporated learning of features from raw images. These models typically consisted of multiple layers of nonlinearity. This property was considered to be very important, and this lead to the development of the first deep learning models. Early examples like Restricted Boltzmann Machines [40], Deep Belief Networks [41], and Stacked Autoencoders [97] showed promise on small datasets. The primary idea behind these works was to leverage the vast amount of unlabeled data to train models. This was called the 'unsupervised pre-training' stage. It was believed that these 'pre-trained' models would serve as a good initialization for further supervised tasks such as image classification. Efforts to scale these algorithms on larger datasets culminated in 2012 during the ILSVRC competition [79], which involved, among other things, the task of classifying an image into one of thousand categories. For the first time, a Convolutional Neural Network (CNN) based deep learned model [56] brought down the error rate on that task by half, beating traditional hand-engineered approaches. Surprisingly, this could be achieved by performing end-to-end supervised training, without the need for unsupervised pre-training. Over the next couple of years, 'ImageNet classification using deep neural networks' [56] became one of the most influential papers in computer vision. Convolutional Neural Networks, a particular form of deep learning models, have since been widely adopted by the vision community. In particular, the network trained by Alex Krizhevsky, popularly called "AlexNet" has been used and modified for various vision problems. Hence, in this chapter, we primarily discuss CNNs, as they are more relevant to the vision community. This can also serve as a guide for beginning practitioners in deep learning/computer vision.

The chapter is organized as follows. We first develop the general principles behind CNNs (Section 2.2), and then discuss various modifications to suit different problems

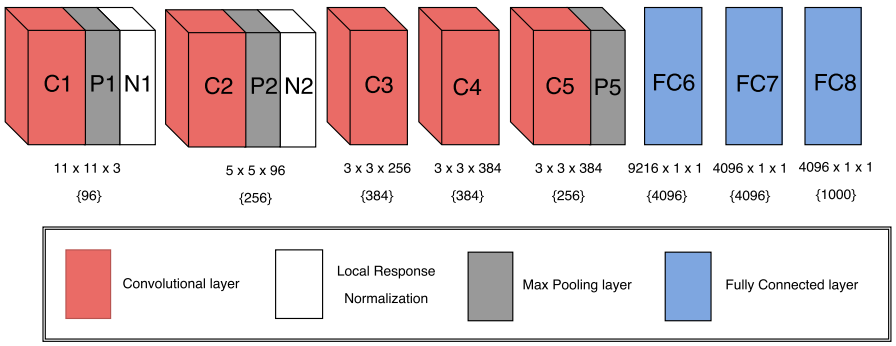

FIGURE 2.1

An illustration of the weights in the AlexNet model. Note that after every layer, there is an implicit ReLU nonlinearity. The number inside curly braces represents the number of filters with dimensions mentioned above it.

(Section 2.3). Finally, we discuss briefly about some of the existing software tools available for implementing these algorithms.

2.2 CONVOLUTIONAL NEURAL NETWORKS

The idea of a Convolutional Neural Network (CNN) is not new. This model had been shown to work well for hand-written digit recognition [59] as early as 1998. However, due to the inability of these networks to scale to much larger images, they slowly fell out of favor. This was largely due to memory and hardware constraints, and the unavailability of large amounts of training data. With increase in computational power thanks to wide availability of GPUs, and the introduction of large scale datasets like the ImageNet [79], it was possible to train larger, more complex models. This was first shown by the popular *AlexNet* model which was discussed earlier. This largely kick-started the usage of deep networks in computer vision.

2.2.1 BUILDING BLOCKS OF CNNS

In this section, we shall look at the basic building blocks of CNNs in general. This assumes that the reader is familiar with traditional neural networks, which we shall call "fully connected layers". Fig. 2.1 shows a representation of the weights in the *AlexNet* model. While the first five layers are convolutional, the last three are fully connected layers.

2.2.1.1 Why Convolutions?

Using traditional neural networks for real-world image classification is impractical for the following reason: Consider a 2D image of size 200×200 for which we would

have 40,000 input nodes. If the hidden layer has 20,000 nodes, the size of the matrix of input weights would be $40{,}000 \times 20{,}000 = 800$ million. This is just for the first layer – as we increase the number of layers, this number increases even more rapidly. Besides, vectorizing an image completely ignores the complex 2D spatial structure of the image. How do we build a system that overcomes both these disadvantages?

One way is to use 2D convolutions instead of matrix multiplications. Learning a set of convolutional filters (each of 11×11, say) is much more tractable than learning a large matrix ($40{,}000 \times 20{,}000$). 2D convolutions also naturally take the 2D structure of images into account. Alternately, convolutions can also be thought of as regular neural networks with two constraints [11]:

- *Local connectivity*. This comes from the fact that we use a convolutional filter with dimensions much smaller than the image it operates on. This contrasts with the *global* connectivity paradigm typically relevant to vectorized images.
- *Weight sharing*. This comes from the fact that we perform convolutions, i.e., we apply the same filter across the image. This means that we use the same *local* filters on many locations in the image. In other words, the weights between all these filters are shared.

There is also evidence from visual neuroscience for similar computations within the human brain. Hubel and Wiesel [47] found two types of cells in the primary visual cortex, the simple cells and the complex cells. The simple cell responded primarily to oriented edges and gratings, which are reminiscent of Gabor filters, a special class of convolutional filters. The complex cells were also sensitive to these edges and grating. However, they exhibited spatial invariance as well. This motivated the Neocognitron model [25], which proposed the learning of convolutional filters in an artificial neural network. This model is said to have inspired convolutional networks, which are analogous to the simple cells mentioned above.

In practical CNNs, however, the convolution operations are not applied in the traditional sense wherein the filter shifts one position to the right after each multiplication. Instead, it is common to use larger shifts (commonly referred to as stride). This is equivalent to performing image down-sampling after regular convolution.

If we wish to train these networks on RGB images, one would need to learn multiple *multi-channel* filters. In the representation in Fig. 2.1, the numbers $11 \times 11 \times 3$, along with {96} below **C1**, indicate that there are 96 filters in the first layers, each of spatial dimension of 11×11, with one for each of the 3 RGB channels. As a result, the resulting feature activation after convolution has 96 channels, and we can now apply another set of multi-channel features for these as well.

We note that this paradigm of convolution-like operations (location independent feature-detectors) is not entirely suitable for registered images. As an example, images of faces require different feature-detectors at different spatial locations. To account for this, some models [90] consider only locally-connected networks with no weight-sharing. Thus, the choice of layer connectivity depends on the underlying type of problem.

2.2.1.2 Max-Pooling

The Neocognitron model inspired the modeling of simple cells as convolutions. Continuing in the same vein, the complex cells can be modeled as a max-pooling operation. This operation can be thought of as a *max filter*, where each $n \times n$ region is replaced with it's max value. This operation serves two purposes:

1. It picks out the highest activation in a local region, thereby providing a small degree of spatial invariance. This is analogous to the operation of complex cells.
2. It reduces the size of the activation for the next layer by a factor of n^2. With a smaller activation size, a smaller number of parameters need to be learned in the later layers.

Other types of pooling include average-pooling, winner-takes-all pooling [87] and stochastic pooling [117]. However, these are not as commonly used max-pooling.

2.2.1.3 Nonlinearity

Deep networks usually consist of convolutions followed by a nonlinear operation after each layer. This is necessary because cascading linear systems (like convolutions) is another linear system. Nonlinearities between layers ensure that the model is more expressive than a linear model.

In theory, no nonlinearity has more expressive power than any other, as long as they are continuous, bounded and monotonically increasing [44]. Traditional feed-forward neural networks used the sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$) or the tanh ($\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) nonlinearities. However, modern convolutional networks use the Rectified Linear Unit (ReLU), which is defined as $\text{ReLU}(x) = \max(0, x)$. CNNs with this nonlinearity have been found to train faster [70].

Recently, Maas et al. [64] introduced a new kind of nonlinearity, called the leaky-ReLU. It was defined as $\text{Leaky-ReLU}(x) = \max(0, x) + \alpha \min(0, x)$, where α is a pre-determined parameter. He et al. [39] improved on this by suggesting that the α parameter also be learned, leading to a much richer model.

2.2.1.4 Fully-Connected Layers

Traditionally, neural networks were composed of matrix multiplications alternated with nonlinearities like sigmoid. These matrix multiplication layers are now called as *fully-connected layer*, owing to the fact that each unit in the previous layer is connected to every unit in the next layer. This is in contrast with convolutional layers, where there is only a local spatial connection. In modern networks, using many fully connected layers are generally avoided as it uses an extremely large number of parameters [38].

2.2.2 DEPTH

The Universal Approximation theorem [44] states that a neural network with a single hidden layer is sufficient to model any continuous function. However, it has been

shown [10] that such networks need an exponentially large number of neurons when compared to a neural network with many hidden layers.

Although the motivation for creating deeper networks was clear, for a long time researchers did not have an algorithm that could efficiently train neural networks with more than 3 layers. With the introduction of greedy layer-wise pre-training [41], researchers were able to train much deeper networks. This played a major role in bringing the so-called *Deep Learning* systems into mainstream machine learning. Modern deep networks such as AlexNet have 7 layers. Recent models like VGGnet [83] and GoogleNet [89] have 19 and 22 layers, respectively, and were shown to perform much better than AlexNet. One disadvantage of these very deep networks is that they become very difficult to train. Srivastava et al. [86] introduced Highway networks, using which it was possible to train very deep neural networks, up to a 100 layers deep. He et al. [38] used a modification of this technique with 152 layers to obtain state-of-the-art results on the ILSVRC 2015 dataset.

2.2.3 LEARNING ALGORITHM

A powerful, expressive model is of little use without an algorithm to learn the model's parameters efficiently. Greedy layer-wise pre-training approaches in the pre-AlexNet era attempted to create such an efficient algorithm. For computer vision tasks, however, it turns out that a simpler supervised training procedure is enough to learn a powerful model.

Learning is generally performed by minimizing a loss function which is dependent on the underlying task. Tasks based on classification use the softmax loss function or the sigmoid cross-entropy function, while those involving regression use the Euclidean error function. In the example of Fig. 2.1, the output of the `FC8` layer is trained to represent one of thousand classes of the dataset.

2.2.3.1 *Gradient-Based Optimization*

Neural networks are generally trained using the backpropagation algorithm [78], which uses the chain rule to speed up computation of the gradient for the gradient descent (GD) algorithm. However, for datasets with thousands (or more) of data points, using GD is impractical. In such cases, an approximation called the Stochastic Gradient Descent (SGD) is often used, where one computes gradients with respect to individual data points rather than the entire dataset. It has been found that training with SGD generalizes better than with GD. However, one disadvantage of SGD is that it is relatively slow to converge [12]. To counteract this, SGD is typically used with a mini-batch, where the mini-batch typically contains a small number of data-points ($\sim$100) rather than a single data-point.

Momentum [74] belongs to a family of methods that aim to speed the convergence of SGD. This is largely used in practice to train deep networks, and is often considered as an essential component. Extensions like Adagrad [23], Nesterov's accelerated GD [72], Adadelta [116] and Adam [55] are known to work equally well,

if not better than vanilla momentum in certain cases. For detailed discussion on how these methods work, the reader is encouraged to read the following survey paper [88].

2.2.3.2 Dropout

When training a network with a large number of parameters, an effective regularization mechanism is essential to combat overfitting. Classical regularizers such as ℓ_1 or ℓ_2 regularization on the weights of the neural net have been found to be insufficient in this aspect. Dropout is a powerful regularization method [42] which has been shown to improve generalization for large neural nets. In dropout, we *randomly* drop neurons with a probability p during training. As a result, only a random subset of neurons are trained in a single iteration of SGD. At test time, we use all neurons, however, we simply multiply the activation of each neuron with p to account for the scaling. Hinton et al. [42] showed that this procedure was equivalent to training a large ensemble of neural nets with shared parameters, and then using their geometric mean to obtain a single prediction.

Many extensions to dropout like DropConnect [99] and Fast Dropout [103] have been shown to work better in certain cases. Maxout [32] is a nonlinearity that improves performance of a network which uses dropout.

2.2.3.3 Batch Normalization

Batch Normalization (BN) [48] is another useful regularizer that improves generalization as well as drastically speeds up convergence. This technique tackles the problem of *internal covariate shift*, where the distribution of each layer's inputs change continuously during the training process. This is a result of each layer's predecessor's changing weights, which result in a changing distribution of output activations. This phenomenon usually slows down training and requires careful initialization.

To counteract this problem, BN normalizes the output activations of a layer to ensure that it's range is within a small interval. Specifically, BN performs normalization with the running average of the mean–variance statistics of each mini-batch. Additional shift-and-scale parameters are also learned to counteract the normalization effect if needed. In recent times, BN has been found to be an essential component in training very deep networks.

2.2.4 TRICKS TO INCREASE PERFORMANCE

While the techniques and components described above are well-grounded, some additional *tricks* are crucial to obtaining *state-of-the-art* performance.

It is well known that machine learning models perform better in the presence of more data. Data augmentation is a process by which some geometric transforms are applied to training data to increase their number. Some examples of commonly used geometric transforms include random cropping, RGB jittering, image flipping and small rotations. It has been found that using augmented data typically boosts performance by about 3% [14].

Also well-known is the fact that an ensemble of models perform better than one. Hence, it is the commonplace to train several CNNs and average their predictions at test time. Using ensembles has been found to typically boost accuracy by 1–2% [83,89].

2.2.5 PUTTING IT ALL TOGETHER: ALEXNET

The building blocks discussed above largely describe AlexNet as a whole. As shown in Fig. 2.1, only layers 1, 2, and 5 contain max-pooling, while dropout is only applied to the last two fully connected layers as they contain the most number of parameters. Layers 1 and 2 also contain Local Response Normalization, which has not been discussed as it has been shown [14] that its absence does not impact performance.

This network was trained on the ILSVRC 2012 training data, which contained 1.2 million training images belonging to 1000 classes. This was trained on 2 GPUs over the course of one month. The same network can be trained today in little under a week using more powerful GPUs [14]. The hyper-parameters of the learning algorithms like learning rate, momentum, dropout and weight decay were hand tuned. It is also interesting to note the trends in the nature of features learned at different layers. The earlier layers tend to learn Gabor-like oriented edges and blob-like features, followed by layers that seem to learn more higher order features like shapes. The very last layers seem to learn semantic attributes such as eyes or wheels, which are crucial parts in several categories. A method to visualize these was provided by Zeiler and Fergus [115].

2.2.6 USING PRE-TRAINED CNNS

One of the main reasons for the success of the AlexNet model was that it was possible to directly use the pre-trained model to do various other tasks which it was not originally intended for. It became remarkably easy to download a learned model, and then tweak it slightly to suit the application at hand. We describe two such ways to use models in this manner.

2.2.6.1 Fine-Tuning

Given a model trained for image classification, how does one modify it to perform a different (but related) task? The answer is to just use the trained weights as an initialization and run SGD again for this new task. Typically, one uses a learning rate much lower than what was used for learning the original net. If the new task is very similar to the task of image classification (with similar categories), then one need not re-learn a lot of layers. The earlier layers can be fixed and only the later, more semantic layers need to be re-learned. However, if the new task is very different, one ought to either re-learn all layers, or learn everything from scratch. The number of layers to re-learn also depends on the number of data points available for training the new task. The more the data, the higher is the number of layers that can be re-learned. The reader is urged to refer to Yosinski et al. [112] for more thorough guidelines.

2.2.6.2 CNN Activations as Features

As remarked earlier, the later layers in AlexNet seem to learn visually semantic attributes. These intermediate representations are crucial in performing 1000-way classification. Since these represent a wide variety of classes, one can use the FC7 activation of an image as a generic feature descriptor. These features have been found to be better than hand-crafted features like SIFT or HoG for various computer vision tasks.

Donahue et al. [22] first introduced the idea of using CNN activations as features and performed tests to determine their suitability for various tasks. Activations of fully-connected layers are also used for image retrieval [7]. In fact, these serve as generic features and can be used in several tasks [75] as a strong baseline. Hypercolumns [36] are activations *across* layers as a feature. Specifically, they look at the activations produced by a single image pixels across the network and pool them together. They were found to be useful for fine-grained tasks such as keypoint localization.

2.2.7 IMPROVING ALEXNET

The performance of AlexNet motivated a number of CNN-based approaches, all aimed at a performance improvement over and above that of AlexNet's. Just as AlexNet was the winner for ILSVRC challenge in 2012, a CNN-based net `Overfeat` [80] was the top-performer at ILSVRC-2013. Their key insight was that training a convolutional network to simultaneously classify, locate, and detect objects in images can boost the classification accuracy and the detection and localization accuracy of all tasks. Given its multi-task learning paradigm, we discuss `Overfeat` when we discuss hybrid CNNs and multi-task learning in Section 2.3.5.

GoogleNet [89], the top-performer at ILSVRC-2014, established that very deep networks can translate to significant gains in classification performance. Since naively increasing the number of layers results in a large number of parameters, the authors employ a number of "design tricks". One such trick is to have a trivial 1×1 convolutional layer after a regular convolutional layer. This has the net effect of not only reducing the number of parameters, but also results in CNNs with more expressive power. In fact, it is argued that having one or more 1×1 convolutional layers is akin to having a multi-layer Perceptron network processing the outputs of the convolutional layer that precedes it. Another trick that the authors utilize is to involve inner layers of the network in the computation of the objective function instead of the typical final softmax layer (as in AlexNet). The authors attribute scale invariance as the reason behind this design decision.

VGG-19 [83] is another example of a high-performing CNN where the deeper-is-better philosophy is applied in the net design. An interesting feature of VGG design is that it forgoes larger sized convolutional filters for stacks of smaller sized filters. These smaller sized filters tend to be chosen so that they contain approximately the same number of parameters as the larger filters they supposedly replace. The net ef-

fect of this design decision is efficiency and regularization-like effect on parameters due to the smaller size of the filters involved.

Deep Residual Networks [38] takes depth to the extreme by training layers that are as deep as 152–200 layers. This is achieved by adding so-called *residual* connections to layers of the neural network. The motivation behind this is to train layers to fit the residual map, rather than directly fit the underlying map. In other words, instead of learning a mapping $h(x)$, we learn $f(x) = h(x) - x$, and then later add $f(x) + x$. This simple technique was found to be useful to learn models with hundreds of layers, yielding state-of-the-art accuracy in many tasks.

2.3 CNN FLAVORS

2.3.1 REGION-BASED CNNS

Most CNNs that are trained for image classification are trained using a dataset of images containing a single object. At test time, even in case of multiple objects, the CNN may still predict a single class. This inherent problem with the design of the CNNs is not restricted to image classification alone. For example, the problem of object detection and localization requires not only classifying the image but also estimating the class and precise location of the object(s) present in the image. Object detection is challenging since we potentially want to detect multiple objects with varying sizes within a single image. It generally requires processing the image patch-wise, looking for the presence of objects. Neural nets have been employed in this way for detecting specific objects like faces [93,77] and for pedestrians [81].

Meanwhile, detecting a set of object-like regions in a given image – also called region proposals or object proposals – has gained a lot of attention [45]. These region proposals are class agnostic and reduce the overhead incurred by the traditional exhaustive sliding window approach. These region proposal algorithms operate at low level and output hundreds of object like image patches at multiple scales. In order to employ a classification net toward the task of object localization, image regions of different scales have to be searched one at a time.

Girshick et al. [28] attempt to solve the object localization problem using a set of region proposals. During test time, the method generates around 2000 category independent region proposals using selective search approach [92] from the test image. They employ a simple affine image warping to feed each of these proposals to a CNN trained for classification. The CNN then describes each of these regions with a fixed size high level semantic feature. Finally, a set of category specific linear SVMs classify each region, as shown in Fig. 2.2. This method achieved the best detection results on the PASCAL VOC 2012 dataset. As this method uses image regions followed by a CNN, it is dubbed R-CNN (Region-based CNN). Generating region proposals is a computational bottleneck. Improvements [27,37] are proposed to make this approach computationally efficient and also to increase the detection accuracy. Fully convolutional nets called Region Proposal Nets (RPNs) [76,109] are presented to compute

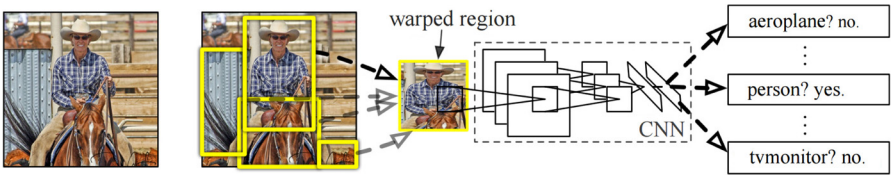

FIGURE 2.2

Object detection system of Girshick et al. [28] using deep features extracted from image regions.

Source: Girshick et al. [28]

region proposals at no extra cost while detecting the objects. These networks detect the objects with a very less number of proposals (as low as 20) compared to the earlier region proposal works.

A series of works adapted the R-CNN approach to extract richer set of features at patch or region level to solve a wide range of target applications in vision. However, CNN representations lack robustness to geometric transformations restricting their usage. Gkioxari and Malik [31] show empirical evidence that the global CNN features are sensitive to general transformations such as translation, rotation and scaling. In their experiments, they report that this inability of global CNN features translates directly into a loss in the classification accuracy. They proposed a simple technique to pool the CNN activations extracted from the local image patches. The method extracts image patches in an exhaustive sliding-window manner at different scales and describes each of them using a CNN. The resulting dense CNN features are pooled using VLAD [50] in order to result in a representation which incorporates spatial as well as semantic information.

Instead of considering the image patches at exhaustive scales and image locations, it is possible to utilize the objectness prior to automatically extract the image patches at different scales [69]. This is shown to be a more robust image representation by aggregating the individual CNN features from the patches for image retrieval.

A related technique can be used to perform multi-label classification as well [105]. This can be done by considering an arbitrary number of region proposals in an image and sharing a common CNN across all of them in order to obtain individual predictions. Finally, we employ a simple pooling technique to produce the final multi-label prediction.

2.3.2 FULLY CONVOLUTIONAL NETWORKS

The success of Convolutional Neural Networks in the tasks of image classification [56,89] and object detection [28] has inspired researchers to use deep networks for more challenging recognition problems like semantic object segmentation and scene parsing. Unlike image classification, semantic segmentation and scene parsing are problems of structured prediction where every pixel in the image grid needs to be

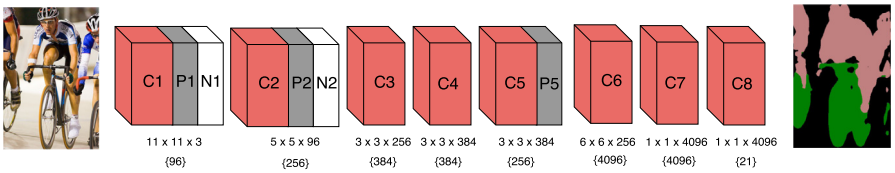

FIGURE 2.3

Fully Convolutional Net – AlexNet modified to be fully convolutional for performing semantic object segmentation on PASCAL VOC 2012 dataset with 21 classes.

assigned a label of the class to which it belongs (e.g., road, sofa, table, etc.). This problem of per-pixel classification has been traditionally approached by generating region-level (e.g., superpixel) hand crafted features and classifying them using a Support Vector Machine (SVM) into one of the possible classes.

Doing away with these engineered features, Farabet et al. [24] used hierarchical learned features from a convolutional neural net for scene parsing. Their approach comprised of densely computing multi-scale CNN features for each pixel and aggregating them over image regions upon which they are classified. However, their method still required the post-processing step of generating over-segmented regions, like superpixels, for obtaining the final segmentation result. Additionally, the CNNs used for multi-scale feature learning were not very deep with only three convolution layers.

Later, Long et al. [61] proposed a fully convolutional network architecture for learning per-pixel tasks, like semantic segmentation, in an end-to-end manner. This is shown in Fig. 2.3. Each layer in the fully convolutional net (FullConvNet) performs a location invariant operation, i.e., a spatial shift of values in the input to the layer will only result in an equivalent scaled spatial shift in its output while keeping the values nearly intact. This property of translational invariance holds true for the convolutional and maxpool layers which form the major building blocks of a FullConvNet. Further, these layers have an output-centered, fixed-size receptive field on its input blob. These properties of the layers of FullConvNet allow it to retain the spatial structure present in the input image in all of its intermediate and final outputs.

Unlike CNNs used for image classification, a FullConvNet does not contain any densely connected/inner product layers as they are not translation invariant. The restriction on the size of input image to a classification CNN (e.g., 227×227 for AlexNet [56], 224×224 for VGG [83]) is imposed due to the constraint on the input size to its inner product layers. Since a FullConvNet does not have any of these inner product layers, it can essentially operate on input images of any arbitrary size.

During the design of CNN architectures, one has to make a trade-off between the number of channels and the spatial dimensions for the data as it passes through each layer. Generally, the number of channels in the data are made to increase progressively while bringing down its spatial resolution, by introducing stride in the convolution and max-pool layers of the net. This is found to be an effective strategy

for generating richer semantic representations in a hierarchical manner. While this method enables the net to recognize complex patterns in the data, it also diminishes the spatial resolution of the data blob progressively after each layer. While this is not a major concern for classification nets which require only a single label for the entire image, this results in per-pixel prediction only at a sub-sampled resolution in case of FullConvNets. For tackling this problem, Long et al. [61] have proposed a deconvolution layer which brings back the spatial resolution from the sub-sampled output through a learned upsampling operation. This upsampling operation is performed at intermediate layers of various spatial dimensions and are concatenated to obtain pixel-level features at the original resolution.

On the other hand, a more simplistic approach [15] for maintaining resolution can be done by removing the stride in the layers of FullConvNet, wherever possible. Following this, the FullConvNet predicted output is modeled as a unary term for Conditional Random Field (CRF) constructed over the image grid at its original resolution. With labeling smoothness constraint enforced through pair-wise terms, the per-pixel classification task is modeled as a CRF inference problem. While this post-processing of FullConvNet's coarse labeling using CRF has been shown to be effective for pixel-accurate segmentation, a better approach [119] would be where the CRF constructed on image is modeled as a Recurrent Neural Network (RNN). By modeling the CRF as an RNN, it can be integrated as a part of any Deep Convolutional Net making the system efficient at both semantic feature extraction and fine-grained structure prediction. This enables the end-to-end training of the entire FullConvNet + RNN system using the stochastic gradient descent (SGD) algorithm to obtain fine pixel-level segmentation.

Noh et al. [73] proposed a semantic object segmentation architecture where a full-fledged deconvolution network follows the convolution network in the segmentation pipeline. Here, the architecture of the deconvolution network mirrors that of the convolution network with the convolution and pooling layers replaced by deconvolution and unpooling layers, respectively. In this architecture, the authors observe that the convolution network plays the role of a multi-dimensional semantic feature extractor whereas the deconvolution network plays the role of an object shape generator using the extracted features. This pipeline of convolution–deconvolution network when used together with a FullConvNet architecture as an ensemble is observed to give superior segmentation performance.

Visual saliency prediction is another important per-pixel prediction problem in computer vision. This task involves predicting the per-pixel saliency map for an image as given by human eye fixations. Huang et al. [46] modified deep networks trained for image classification into fully convolutional networks and fine-tuned them for eye fixation prediction with saliency based objectives. Their model handles multi-scale aspects of saliency by considering a two-stream architecture with each stream specializing for a scale. Kruthiventi and Babu [57] proposed another fully convolutional architecture which characterizes the multi-scale aspects using inception blocks and captures the global context using convolutional layers with large receptive fields. Another work [108], proposed a multi-task FullConvNet architecture – DeepSaliency

– for joint saliency detection and semantic object segmentation. Their work showed that learning features collaboratively for two related prediction tasks can boost overall performance.

2.3.3 MULTI-MODAL NETWORKS

The success of CNNs on standard RGB vision tasks is naturally extended to works on other perception modalities like RGB-D and motion information in the videos. There has been an increasing evidence for the successful adaptation of the CNNs to learn efficient representations from the depth images. Socher et al. [84] exploited the information from color and depth modalities for addressing the problem of classification. In their approach, a single layer of CNN extracts low level features from both RGB and depth images separately. These low level features from each modality are given to a set of RNNs for embedding into a lower dimension. Concatenation of the resulting features forms the input to the final soft-max layer. The work by Couprie et al. [18] extended the CNN method of [24] to label the indoor scenes by treating depth information as an additional channel to the existing RGB data. Similarly, Wang et al. [100] adapt an unsupervised feature learning approach to scene labeling using RGB-D input with four channels. Gupta et al. [35] proposed an encoding for the depth images that allows CNNs to learn stronger features than from the depth image alone. They encode depth image into three channels at each pixel: horizontal disparity, height above ground, and the angle the pixel's local surface normal makes with the inferred gravity direction. Their approach for object detection and segmentation processes RGB and the encoded depth channels separately. The learned features are fused by concatenating and further classified using an SVM classifier.

Similarly, one can think of extending these works for video representation and understanding. When compared to still images, videos provide important additional information in the form of motion. However, a majority of the early works that attempted to extend CNNs for video, fed the networks with raw frames. This makes for a much difficult learning problem. Jhuang et al. [51], proposed a biologically inspired model for action recognition in videos with a predefined set of spatio-temporal filters in the initial layer. Combined with a similar but spatial HMAX (Hierarchical model and X) model, Kuehne et al. [58] proposed spatial and temporal recognition streams. Ji et al. [52] addressed an end-to-end learning of the CNNs for videos for the first time using 3-D convolutions over a bunch of consecutive video frames. A more recent work by Karpathy et al. [54] proposes a set of techniques to fuse the appearance information present from a stack of consecutive frames in a video. However, the authors report that the net that processes individual frames performs on par with the net that operates on a stack of frames. This might suggest that the learned spatio-temporal filters are not suitable to capture the motion patterns efficiently.

A more suitable CNN model to represent videos is proposed in a contemporaneous work by Simonyan and Zisserman [82], which is called a two-stream network approach. Though the model in Kuehne et al. [58] is also a two stream model, the

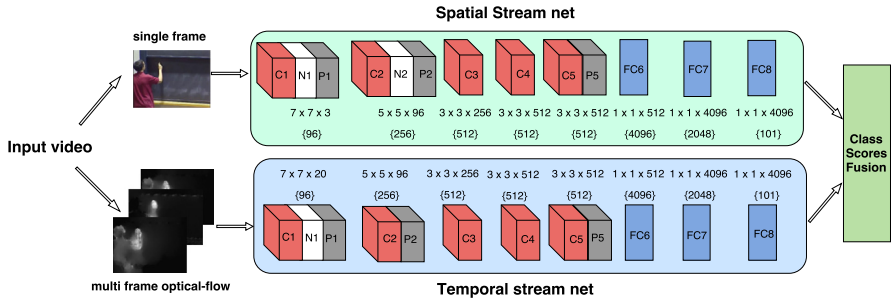

FIGURE 2.4

Two stream architecture for video classification from Simonyan and Zisserman [82].

main difference is that the streams are shallow and implemented with hand-crafted models. The reason for the success of this approach is the natural ability of the videos to be separated into spatial and temporal components. The spatial component in the form of frames captures the appearance information like the objects present in the video. The temporal component in the form of motion (optical flow) across the frames captures the movement of the objects. These optical flow estimates can be obtained either from classical approaches [9] or deep-learned approaches [106].

This approach models the recognition system dividing into two parallel streams as depicted in Fig. 2.4. Each is implemented by a dedicated deep CNN, whose predictions are later fused. The net for the spatial stream is similar to the image recognition CNN and processes one frame at a time. However, the temporal stream takes the stacked optical flow of a bunch of consecutive frames as input and predicts the action. Both the nets are trained separately with the corresponding input. An alternative motion representation using the trajectory information similar to Wang et al. [101] is also observed to perform similar to optical flow.

The most recent methods that followed [82] have similar two-stream architecture. However, their contribution is to find the most active spatio-temporal volume for the efficient video representation. Inspired from the recent progress in the object detection in images, Georgia et al. [30] built action models from shape and motion cues. They start from the image proposals and select the motion salient subset of them and extract spatio-temporal features to represent the video using the CNNs.

Wang et al. [102] employ deep CNNs to learn discriminative feature maps and conduct trajectory constrained pooling to summarize into an effective video descriptor. The two streams operate in parallel extracting local deep features for the volumes centered around the trajectories.

In general, these multi-modal CNNs can be modified and extended to suit any other kind of modality like audio, text to complement the image data leading to a better representation of image content.

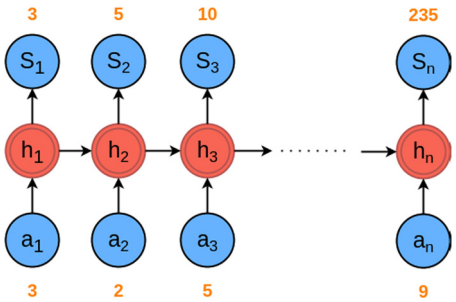

FIGURE 2.5

Toy RNN example – problem of sequence addition. The inputs and outputs are shown in blue. The red cells correspond to the hidden units. An unrolled version of the RNN is shown.

2.3.4 CNNS WITH RNNS

While CNNs have made remarkable progress in various tasks, they are not very suitable for learning sequences. Learning such patterns requires memory of previous states and feedback mechanisms which are not present in CNNs. RNNs are neural nets with at least one feedback connection. This looping structure enables the RNN to have an internal memory and to learn temporal patterns in data.

Fig. 2.5 shows the unrolled version of a simple RNN applied to a toy example of sequence addition. The problem is defined as follows: Let a_t be a positive number, corresponding to the input at time t. The output at time t is given by

$$S_t = \sum_{i=1}^{t} a_i.$$

We consider a very simple RNN with just one hidden layer. The RNN can be described by equations below:

$$h_{t+1} = f_h(W_{ih} \times a_t + W_{hh} \times h_t),$$
$$S_{t+1} = f_o(W_{ho} \times h_{t+1})$$

where W_{ih}, W_{hh}, and W_{ho} are learned weights, and f_h and f_o are nonlinearities. For the toy problem considered above, the weights learned would result in $W_{ih} = W_{hh} = W_{ho} = 1$. Let us consider the nonlinearity to be ReLu. The equations would then become

$$h_{t+1} = ReLu(a_t + h_t),$$
$$S_{t+1} = ReLu(h_{t+1}).$$

Thus, as shown in Fig. 2.5, the RNN stores previous inputs in memory and learns to predict the sum of the sequence up to the current time step t.

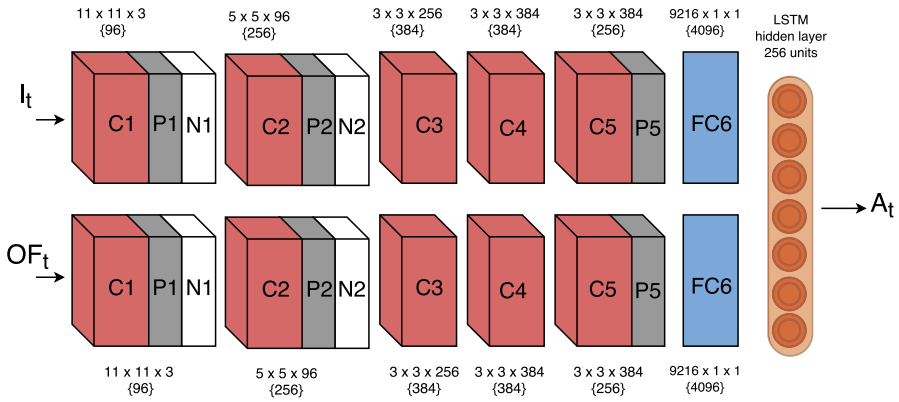

FIGURE 2.6

LRCN. A snapshot of the model at time t. I_t refers to the frame and OF_t the optical flow at t. The video is classified by averaging the output A_t over all t.

As with CNNs, recurrent neural networks have been trained with various back propagation techniques. These conventional methods, however, resulted in the *vanishing gradient problem*, i.e., the errors sent backward over the network, either grew very large or vanished leading to problems in convergence. In 1997, Hochreiter and Schmidhuber [43] introduced LSTM (Long Short Term Memory), which succeeded in overcoming the vanishing gradient problem, by introducing a novel architecture consisting of units called Constant Error Carousels. LSTMs were thus able to learn very deep RNNs and successfully remembered important events over long (thousands of steps) durations of time.

Over the next decade, LSTMs became the network of choice for several sequence learning problems, especially in the fields of speech and handwriting recognition [33,34]. In the sections that follow, we shall discuss applications of RNNs in various computer vision problems.

2.3.4.1 Action Recognition

Recognizing human actions from videos has long been a pivotal problem in the tasks of video understanding and surveillance. Actions, being events which take place over a finite length of time, are excellent candidates for a joint CNN–RNN model.

In particular, we discuss the model proposed by Donahue et al. [21]. They use raw *RGB* values as well as *optical flow* features to jointly train a variant of AlexNet combined with a single layer of LSTM (256 hidden units). Frames of the video are sampled, passed through the trained network, and classified individually. The final prediction is obtained by averaging across all the frames. A snapshot of this model at time t is shown in Fig. 2.6.

Yue-Hei Ng et al. [113] instead utilize a deep LSTM having five stacked layers of 512 units each following the CNN. Each video frame is fed to the CNN and then

processed by the multi-layer LSTM. A softmax classifier forms the final layer and performs action classification at each time step.

2.3.4.2 Image and Video Captioning

Another important aspect of scene understanding is the textual description of images and videos. Relevant textual description also helps complement image information, as well as form useful queries for retrieval.

RNNs have long been used for machine translation [8,16], where the task is to convert a sentence from one language to another. In these works, the common approach has been to use encoder RNNs to generate a latent vector representation of the sentence in the source language. A decoder RNN then transforms this latent vector into the sentence representation for the target language. This general framework can be used for image description as well, where the idea is to perform image-to-sentence 'translation' instead of between sentences of two languages. Vinyals et al. [98] developed such an end-to-end system called *Show and Tell*, by first encoding an image using a CNN and then using the encoded image as an input to an RNN, which generated sentences. An alternative way to approach the problem is using the multi-modal RNN (m-RNN) [67], which generates sentences by predicting the next word from the present word. To make this decision, image embeddings are used by RNN, making it multi-modal. This was further extended [66] by incorporating a special form of weight-sharing, enabling the network to learn novel visual concepts from images. One problem with the methods discussed above is that it is difficult to understand *why* the network produced the caption that it did. To this end, Xu et al. developed *Show, Attend and Tell*, which incorporates attention into the image description framework. Attention mechanisms act as gates to the input image. Given an image, an attention mechanism learns to select only a small spatial region in the image. Using this, the authors are able to localize nouns in a text description to objects in the image.

Similar to images, natural language description can also be obtained for videos. Considering a video as a collection of image frames, a simple technique would involve simply collecting image representations and use the set of representations for video description. Venugopalan et al. [96] use a mean-pooled CNN representation of image frames for a video. They train an LSTM to use this input to generate a description for the video. They further improve upon this task by developing S2VT [95], a stacked LSTM model which accounts for both the RGB as well as flow information available in videos, rather than just the RGB information used previously.

2.3.4.3 Visual Question Answering

A more comprehensive understanding of images should enable a system not only to make a statement about it, but also to answer questions related to it. Therefore answering questions based on visual concepts in an image is the next natural step for machine understanding algorithms. This requires the system to model both the textual question and the image representation, before generating an answer.

A critical question when designing such a system is that of combining visual and text (question) information in an efficient manner. Malinowski et al. [65] train

an LSTM which models the concatenation of a question and an answer. After the question is specified, the model is expected to predict the answer word-by-word. At each time step, visual information in the form of CNN features is also available to the LSTM. This system can, in principle, use information from both modalities at each time step. The disadvantage of this model is again that it is not possible to know *why* the network gave a particular answer for a given question. To overcome this, attention mechanism is used, as mentioned earlier for the image captioning example.

Yang et al. [111] develop a stack attention framework, where an image is queried multiple times and the answer is determined progressively. Different parts of the image are described by CNN features, and the question is described by an LSTM. The stacked attention layers use these descriptions to select regions of the image which may be relevant to the answer. These regions are progressively refined until the final attention layer, where the answer is predicted.

Wu et al. [107] use knowledge generated from attributes and image captions along with that mined from knowledge bases to perform visual question answering. They first train a CNN to generate attributes for a given image. These attributes are used to query the knowledge base. The retrieved information is collated and converted to a vector representation. The attributes are also used to generate image captions. The top five captions are also summarized to produce a combined representation. The attribute vector in combination with the caption vector and the knowledge base vector are used to initialize the LSTM. The question is then fed to the LSTM, which proceeds to generate the answer.

2.3.5 HYBRID LEARNING METHODS

2.3.5.1 Multi-Task Learning

Multi-task learning is essentially a machine learning paradigm wherein the objective is to train the learning system to perform well on multiple tasks. Multi-task learning frameworks tend to exploit shared representations that exist among the tasks to obtain a better generalization performance than counterparts developed for a single task alone.

In CNNs, multi-task learning is realized using different approaches. One class of approaches utilize a multi-task loss function with hyper-parameters typically regulating the task losses. For example, Girshik et al. [29] employ a multi-task loss to train their network jointly for classification and bounding-box regression tasks thereby improving performance for object detection. Zhang et al. [118] propose a facial landmark detection network which adaptively weights auxiliary tasks (e.g., head pose estimation, gender classification, age estimation) to ensure that a task that is deemed not beneficial to accurate landmark detection is prevented from contributing to the network learning. Devries et al. [20] demonstrate improved performance for facial expression recognition task by designing a CNN for simultaneous landmark localization and facial expression recognition. A hallmark of these approaches is the division of tasks into primary task and auxiliary task(s) wherein the purpose of the latter is typically to improve the performance of the former (see Fig. 2.7). Note that in some

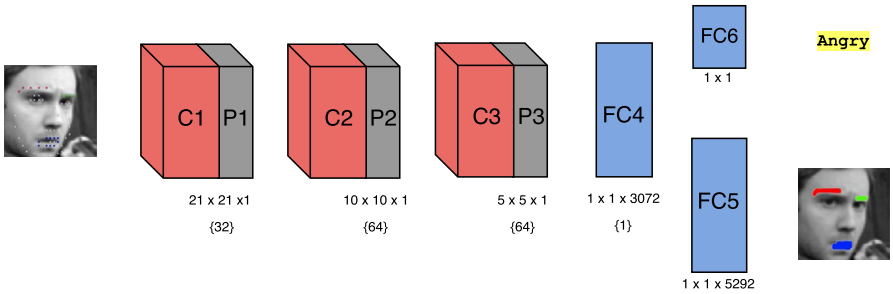

FIGURE 2.7

The facial expression recognition system of Devries et al. [20] which utilizes facial landmark (shown overlaid on the face toward the right of the image) recognition as an auxiliary task which helps improve performance on the main task of expression recognition.

cases, the auxiliary task may play an adversarial role in improving the performance. For instance, Ganin et al. [26] design a network for visual domain-adaptation wherein the performance of the auxiliary task (domain classification) is sought to be degraded to assist the main task of domain-adaptation.

Some approaches tend to have significant portions of the original network modified for multiple tasks. For instance, Sermanet et al. [80] replace pre-trained layers of a net originally designed to provide spatial (per-pixel) classification maps with a regression network and fine-tune the resulting net to achieve simultaneous classification, localization, and detection of scene objects.

Another class of multi-task approaches tend to have task-specific sub-networks as a characteristic feature of CNN design. Li et al. [60] utilize separate sub-networks for the joint point regression and body part detection tasks. Kruthiventi et al. [85] proposed a deep fully convolutional network with a branched architecture for simultaneously predicting eye fixations and segmenting salient objects in an image. Wang et al. [104] adopt a serially stacked design wherein a localization sub-CNN and the original object image are fed into a segmentation sub-CNN to generate its object bounding box and extract its segmentation mask. To solve an unconstrained word image recognition task, Jaderberg et al. [49] propose an architecture consisting of a character sequence CNN and an N-gram encoding CNN which act on an input image in parallel and whose outputs are utilized along with a CRF model to recognize the text content present within the image. Typically, the task-specific sub-networks do not communicate directly. However, there are approaches where cross-network connections exist. These connections are designed to achieve an optimal trade off between shared and task-specific aspects of the overall learning framework [68,62].

2.3.5.2 *Similarity Learning*

Apart from classification, CNNs can also be used for tasks like metric learning. Rather than asking the CNN to identify objects, we can instead ask it to verify

Table 2.1 A curated list of software for Convolutional Neural Networks

Software	Interfaces provided	AutoDiff
Caffe [53]	Python, MATLAB, C++, command-line	No
Theano [91]	Python	Yes
Tensorflow [5]	C++, Python	Yes
Torch [4]	Torch, C	Yes (non-native)
CNTK [6]	C++, command-line	No
MatConvNet [94]	MATLAB	No
Chainer [1]	Python	No

whether two images contain the same object or not. In other words, we ask the CNN to learn which images are *similar*, and which are not. Image retrieval is one application where such questions are routinely asked.

Structurally, Siamese networks resemble two-stream networks discussed previously. However, the difference here is that both 'streams' have identical weights. Siamese networks consist of two separate (but identical) networks, where two images are fed in as input. Their activations are combined at a later layer, and the output of the network consists of a single number, or a *metric*, which is a notion of distance between the images. Training is done so that images which are considered to be similar have a lower output score than images which are considered different. Bromley et al. [13] introduced the idea of Siamese networks and used it for signature verification. Later on, Chopra et al. [17] extended it for face verification. Zagoruyko et al. [114] further extended and generalized this to learning similarity between image patches.

2.4 SOFTWARE FOR DEEP LEARNING

We would like to end this chapter with a short discussion on the various software packages available for deep learning. Table 2.1 provides a non-exhaustive list.

For computer vision applications, Caffe seems to be the most widely used software framework. It is written in C++ and is often used via the command line/Python interfaces. The design of this framework makes it easy to use pre-trained neural networks out of the box.

In the general deep learning community, Theano, Torch, and Tensorflow seem to be more popular. These frameworks are designed to make it easy to modify low level details of neural network models. While Theano and Tensorflow are used in Python, Torch is used in Lua. There also exist a few wrapper libraries for Theano, such as Lasagne [3] and Keras [2], which provide commonly used functionalities as well as some support for pre-trained models.

In this context, Automatic Differentiation (AutoDiff) [71] is an important feature to have in such framework. When specifying new layers/regularizers for neural networks, one also needs to specify how the gradient through that layer would look like.

AutoDiff automatically computes the derivative of complicated functions in closed form using the definitions of derivatives of its primitives.

REFERENCES

1. Chainer, http://chainer.org/, 2016, accessed: 17 May 2016.
2. Keras, http://keras.io/, 2016, accessed: 17 May 2016.
3. Lasagne, http://lasagne.readthedocs.io/, 2016, accessed: 17 May 2016.
4. Torch, http://torch.ch/, 2016, accessed: 17 May 2016.
5. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467, 2016.
6. A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, R. Hoens, et al., An Introduction to Computational Networks and the Computational Network Toolkit, Technical report, 2014.
7. A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, in: Lect. Notes Comput. Sci., vol. 8689, Springer International Publishing, 2014, pp. 584–599.
8. D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv:1409.0473, 2014.
9. S. Baker, I. Matthews, Lucas–Kanade 20 years on: a unifying framework, Int. J. Comput. Vis. 56 (3) (2004) 221–255.
10. Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
11. C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
12. L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
13. J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network, Int. J. Pattern Recognit. Artif. Intell. 7 (04) (1993) 669–688.
14. K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving deep into convolutional nets, in: Proceedings of the British Machine Vision Conference, Nottingham, UK, BMVA Press, 2014.
15. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Semantic image segmentation with deep convolutional nets and fully connected CRFS, arXiv:1412.7062, 2014.
16. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, arXiv:1406.1078, 2014.
17. S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, CVPR 2005, San Diego, CA, USA, IEEE, 2005, pp. 539–546.
18. C. Couprie, C. Farabet, L. Najman, Y. LeCun, Indoor semantic segmentation using depth information, arXiv:1301.3572, 2013.

19. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, CVPR 2005, San Diego, CA, USA, IEEE, 2005, pp. 886–893.
20. T. Devries, K. Biswaranjan, G.W. Taylor, Multi-task learning of facial landmarks and expression, in: 2014 Canadian Conference on Computer and Robot Vision (CRV), IEEE, 2014, pp. 98–103.
21. J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2625–2634.
22. J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: a deep convolutional activation feature for generic visual recognition, arXiv:1310.1531, 2013.
23. J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
24. C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.
25. K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
26. Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1180–1189.
27. R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
28. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, IEEE, 2014, pp. 580–587.
29. R.B. Girshick, Fast R-CNN, arXiv:1504.08083, 2015.
30. G. Gkioxari, J. Malik, Finding action tubes, in: CVPR, 2015.
31. Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, in: Lect. Notes Comput. Sci., vol. 8695, Springer International Publishing, 2014, pp. 392–407.
32. I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, arXiv:1302.4389, 2013.
33. A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868.
34. A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, IEEE, 2013, pp. 6645–6649.
35. S. Gupta, R. Girshick, P. Arbelaez, J. Malik, Learning rich features from RGB-D images for object detection and segmentation, in: ECCV, 2014.
36. B. Hariharan, P. Arbeláez, R. Girshick, J. Malik, Hypercolumns for object segmentation and fine-grained localization, arXiv:1411.5752, 2014.
37. K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, arXiv:1406.4729, 2014.
38. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, arXiv:1512.03385, 2015.

39. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, arXiv:1502.01852, 2015.
40. G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.
41. G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
42. G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv:1207.0580, 2012.
43. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
44. K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
45. J. Hosang, R. Benenson, P. Dollár, B. Schiele, What makes for effective detection proposals?, IEEE Trans. Pattern Anal. Mach. Intell. 38 (4) (2016) 814–830.
46. X. Huang, C. Shen, X. Boix, Q. Zhao, Salicon: reducing the semantic gap in saliency prediction by adapting deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 262–270.
47. D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J. Physiol. 160 (1) (1962) 106.
48. S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167, 2015.
49. M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman, Deep structured output learning for unconstrained text recognition, arXiv:1412.5903, 2014.
50. H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, C. Schmid, Aggregating local image descriptors into compact codes, IEEE Trans. Pattern Anal. Mach. Intell. 34 (9) (2012) 1704–1716.
51. H. Jhuang, T. Serre, L. Wolf, T. Poggio, A biologically inspired system for action recognition, in: IEEE 11th International Conference on Computer Vision, ICCV 2007, 2007, pp. 1–8.
52. S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 221–231.
53. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, arXiv:1408.5093, 2014.
54. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, IEEE, 2014, pp. 1725–1732.
55. D. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv:1412.6980, 2014.
56. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.
57. S.S. Kruthiventi, K. Ayush, R.V. Babu, DeepFix: a fully convolutional neural network for predicting human eye fixations, arXiv:1510.02927, 2015.
58. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, HMDB: a large video database for human motion recognition, in: 2011 IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, IEEE, 2011, pp. 2556–2563.

59. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
60. S. Li, Z.-Q. Liu, A. Chan, Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network, Int. J. Comput. Vis. 113 (1) (2015) 19–36.
61. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.
62. M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks, in: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, 2015, pp. 97–105.
63. D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
64. A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. ICML, vol. 30, 2013.
65. M. Malinowski, M. Rohrbach, M. Fritz, Ask your neurons: a neural-based approach to answering questions about images, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1–9.
66. J. Mao, X. Wei, Y. Yang, J. Wang, Z. Huang, A.L. Yuille, Learning like a child: fast novel visual concept learning from sentence descriptions of images, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2533–2541.
67. J. Mao, W. Xu, Y. Yang, J. Wang, A. Yuille, Deep captioning with multimodal recurrent neural networks (M-RNN), arXiv:1412.6632, 2014.
68. I. Misra, A. Shrivastava, A. Gupta, M. Hebert, Cross-stitch networks for multi-task learning, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.
69. K. Mopuri, R. Babu, Object level deep feature pooling for compact image representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 62–70.
70. V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.
71. R.D. Neidinger, Introduction to automatic differentiation and Matlab object-oriented programming, SIAM Rev. 52 (3) (2010) 545–563.
72. Y. Nesterov, A method of solving a convex programming problem with convergence rate $O(1/k2)$, Sov. Math. Dokl. 27 (1983) 372–376.
73. H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520–1528.
74. B.T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17.
75. A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Columbus, OH, USA, IEEE, 2014, pp. 512–519.
76. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.

77. H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1) (1998) 23–38.
78. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cogn. Model. 5 (1988) 3.
79. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
80. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: integrated recognition, localization and detection using convolutional networks, arXiv:1312.6229, 2013.
81. P. Sermanet, K. Kavukcuoglu, S. Chintala, Y. LeCun, Pedestrian detection with unsupervised multi-stage feature learning, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, USA, IEEE, 2013, pp. 3626–3633.
82. K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Adv. Neural Inf. Process. Syst., vol. 27, 2014, pp. 568–576.
83. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556, 2014.
84. R. Socher, B. Huval, B. Bath, C.D. Manning, A.Y. Ng, Convolutional-recursive deep learning for 3D object classification, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Adv. Neural Inf. Process. Syst., vol. 25, 2012, pp. 656–664.
85. Srinivas S.S. Kruthiventi, J.H.D. Vennela Gudisa, R.V. Babu, Saliency unified: a deep architecture for simultaneous eye fixation prediction and salient object segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
86. R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2368–2376.
87. R.K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, J. Schmidhuber, Compete to compute, in: Advances in Neural Information Processing Systems, 2013, pp. 2310–2318.
88. I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1139–1147.
89. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, arXiv:1409.4842, 2014.
90. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: closing the gap to human-level performance in face verification, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, IEEE, 2014, pp. 1701–1708.
91. Theano Development Team, Theano: a Python framework for fast computation of mathematical expressions, arXiv:1605.02688, 2016.
92. J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, A.W.M. Smeulders, Selective search for object recognition, Int. J. Comput. Vis. 104 (2) (2013) 154–171.
93. R. Vaillant, C. Monrocq, Y. Le Cun, Original approach for the localisation of objects in images, IEE Proc., Vis. Image Signal Process. 141 (4) (1994) 245–250.
94. A. Vedaldi, K. Lenc, MatConvNet – convolutional neural networks for MATLAB, 2015.
95. S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko, Sequence to sequence – video to text, in: The IEEE International Conference on Computer Vision (ICCV), 2015.
96. S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, K. Saenko, Translating videos to natural language using deep recurrent neural networks, arXiv:1412.4729, 2014.

97. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.
98. O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3156–3164.
99. L. Wan, M. Zeiler, S. Zhang, Y.L. Cun, R. Fergus, Regularization of neural networks using DropConnect, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.
100. A. Wang, J. Lu, G. Wang, J. Cai, T.-J. Cham, Multi-modal unsupervised feature learning for RGB-D scene labeling, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Zurich, Switzerland, in: Lect. Notes Comput. Sci., vol. 8693, Springer, 2014, pp. 453–467.
101. H. Wang, C. Schmid, Action recognition with improved trajectories, in: IEEE International Conference on Computer Vision, Sydney, Australia, 2013.
102. L. Wang, Y. Qiao, X. Tang, Action recognition with trajectory-pooled deep-convolutional descriptors, in: CVPR, 2015, pp. 4305–4314.
103. S. Wang, C. Manning, Fast dropout training, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 118–126.
104. X. Wang, L. Zhang, L. Lin, Z. Liang, W. Zuo, Deep joint task learning for generic object extraction, arXiv:1502.00743, 2015.
105. Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, S. Yan, CNN: single-label to multi-label, arXiv:1406.5726, 2014.
106. P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, DeepFlow: large displacement optical flow with deep matching, in: 2013 IEEE International Conference on Computer Vision (ICCV), Portland, USA, IEEE, 2013, pp. 1385–1392.
107. Q. Wu, C. Shen, A.v.d. Hengel, P. Wang, A. Dick, Image captioning and visual question answering based on attributes and their related external knowledge, arXiv:1603.02814, 2016.
108. L. Xi, L. Zhao, L. Wei, M. Yang, F. Wu, Y. Zhuang, H. Ling, J. Wang, DeepSaliency: multi-task deep neural network model for salient object detection, arXiv:1510.05484, 2015.
109. B. Yang, J. Yan, Z. Lei, S.Z. Li, Craft objects from images, arXiv:1604.03239, 2016.
110. J. Yang, Y.-G. Jiang, A.G. Hauptmann, C.-W. Ngo, Evaluating bag-of-visual-words representations in scene classification, in: Proceedings of the International Workshop on Workshop on Multimedia Information Retrieval, ACM, New York, NY, USA, 2007, pp. 197–206.
111. Z. Yang, X. He, J. Gao, L. Deng, A. Smola, Stacked attention networks for image question answering, arXiv:1511.02274, 2015.
112. J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014, pp. 3320–3328.
113. J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici, Beyond short snippets: deep networks for video classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4694–4702.
114. S. Zagoruyko, N. Komodakis, Learning to compare image patches via convolutional neural networks, arXiv:1504.03641, 2015.

115. M. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, in: Lect. Notes Comput. Sci., vol. 8689, Springer International Publishing, 2014, pp. 818–833.
116. M.D. Zeiler, ADADELTA: an adaptive learning rate method, arXiv:1212.5701, 2012.
117. M.D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, arXiv:1301.3557, 2013.
118. Z. Zhang, P. Luo, C.C. Loy, X. Tang, Learning and transferring multi-task deep representation for face alignment, arXiv:1408.3967, 2014.
119. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P. Torr, Conditional random fields as recurrent neural networks, arXiv:1502.03240, 2015.