Chapter 2
Basics of Supervised Deep Learning

2.1 Introduction

The use of supervised and unsupervised deep learning models has grown at a fast rate due to their success with learning of complex problems. High-performance computing resources, availability of huge amounts of data (labeled and unlabeled) and state-of-the-art open-source libraries are making deep learning more and more feasible for various applications. Since the main focus of this chapter is on supervised deep learning, Convolutional Neural Network (CNN or ConvNets) that is one of the most commonly used supervised deep learning models is discussed in this chapter.

2.2 Convolutional Neural Network (ConvNet/CNN)

Convolutional Neural Network also known as ConvNet or CNN is a deep learning technique that consists of multiple numbers of layers. ConvNets are inspired by the biological visual cortex. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. Different neurons in the brain respond to different features. For example, certain neurons fire only in the presence of lines of a certain orientation, some neurons fire when exposed to vertical edges and some when shown horizontal or diagonal edges. This idea of certain neurons having a specific task is the basis behind ConvNets.

ConvNets have shown excellent performance on several applications such as image classification, object detection, speech recognition, natural language processing, and medical image analysis. Convolutional neural networks are powering core of computer vision that has many applications which include self-driving cars, robotics, and treatments for the visually impaired. The main concept of ConvNets is to obtain local features from input (usually an image) at higher layers and combine them into more complex features at the lower layers. However, due to its multilayered architecture, it is computationally exorbitant and training such networks on a large dataset

M. A. Wani et al., *Advances in Deep Learning*, Studies in Big Data 57,
https://doi.org/10.1007/978-981-13-6794-6_2

takes several days. Therefore, such deep networks are usually trained on GPUs. Convolutional neural networks are so powerful on visual tasks that they outperform almost all the conventional methods.

2.3 Evolution of Convolutional Neural Network Models

LeNet: The first practical convolution-based architecture was LeNet which used backpropagation for training the network. LeNet was designed to classify handwritten digits (MNIST), and it was adopted to read large numbers of handwritten checks in the United States. Unfortunately, the approach did not get much success as it did not scale well to larger problems. The main reasons for this limitation were as follows:

a. Small labeled datasets.
b. Slow computers.
c. Use of wrong nonlinearity (activation) function.

The use of appropriate activation function in a neural network has huge impact on the final performance. Any deep neural network that uses a nonlinear activation function like sigmoid or tanh and is trained using backpropagation suffers from *vanishing gradient*. Vanishing gradient is a problem found in training the neural networks with gradient-based training methods. Vanishing gradient makes it hard to train and tune the parameters of the top layers in a neural network. The problem worsens as the total number of layers in the network increases.

AlexNet: The first breakthrough came in 2012 when the convolutional model which was named AlexNet significantly outperformed all other conventional methods in ImageNet Large-Scale Visual Recognition Competition (ILSVRC) 2012 that featured the ImageNet dataset. The AlexNet brought down classification error rate from 26 to 15%, a significant improvement at that time. AlexNet was simple but much more efficient than LeNet. The improvements to overcome the above mentioned problems were due to the following reasons:

a. Large labeled image database (ImageNet), which contained around 15 million labeled images from a total of over 22,000 categories, was used.
b. The model was trained on high-speed GTX 580 GPUs for 5 to 6 days.
c. ReLU (Rectified Linear Unit) $f(x) = \max(x, 0)$ activation function was used. This activation function is several times faster than the conventional activation functions like sigmoid and tanh. The ReLU activation function does not experience the vanishing gradient problem.

AlexNet consists of five convolutional layers, three pooling layers, three fully connected layers, and a 1000-way softmax classifier.

ZFNet: In 2013, an improved version of CNN architecture called ZFNet was introduced. ZFNet reduced the filter size in the first layer from 11×11 to 7×7 and used a stride of 2 instead of 4 which resulted in more distinctive features and fewer

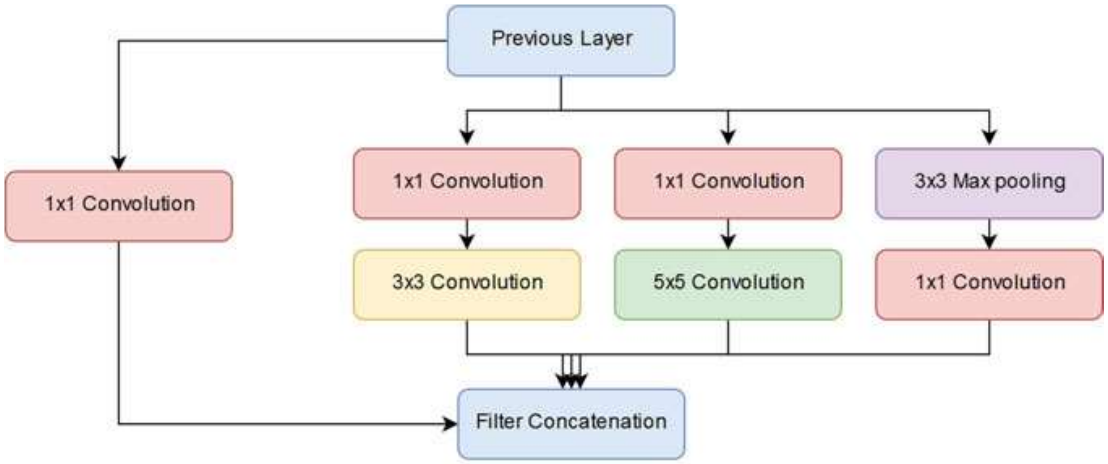

Fig. 2.1 Inception module in GoogLeNet

dead features. ZFNet turned out to be the winner of ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2013.

VGGNet: VGGNet, introduced in 2014, used increased depth of the network for improving the results. The depth of the network was made 19 layers by adding more convolutional layers with 3×3 filters, along with 2×2 max-pooling layers with stride and padding of 1 in all layers. Reducing filter size and increasing the depth of the network resulted in CNN architecture that produced more accurate results. VGGNet achieved an error rate of 7.32% in ILSVRC 2014 and was the runner-up model in ILSVRC 2014.

GoogLeNet: Google developed a ConvNet model called GoogLeNet in 2015. The model has 22 layers and was the winner of ILSVRC 2015 for having the error rate of 6.7%. The previous ConvNet models had convolution, and pooling layers stacked on top of each other but the GoogLeNet architecture is a little different. It uses an *inception* module which helps in reducing the number of parameters in the network. The *inception* module is actually a concatenated layer of convolutions (3×3 and 5×5 convolutions) and pooling sub-layers at different scales with their output filter banks concatenated into a single output vector making the input for the succeeding stage. These sub-layers are not stacked sequentially but the sub-layers are connected in parallel as shown in Fig. 2.1. In order to compensate for additional computational complexity due to extra convolutional operations, 1×1 convolution is used that results in reduced computations before expensive 3×3 and 5×5 convolutions are performed. GoogLeNet model has two convolutional layers, four max-pooling layers, nine inception layers, and a softmax layer. The use of this special inception architecture makes GoogLeNet to have 12 times lesser parameters than AlexNet.

Increasing the number of layers increases the number of features which enhances the accuracy of the network. However, there is a practical limitation to that. (1) Vanishing gradients: Some neurons in too deep networks may die during training which can cause loss of useful information and (2) Optimization difficulty: too many param-

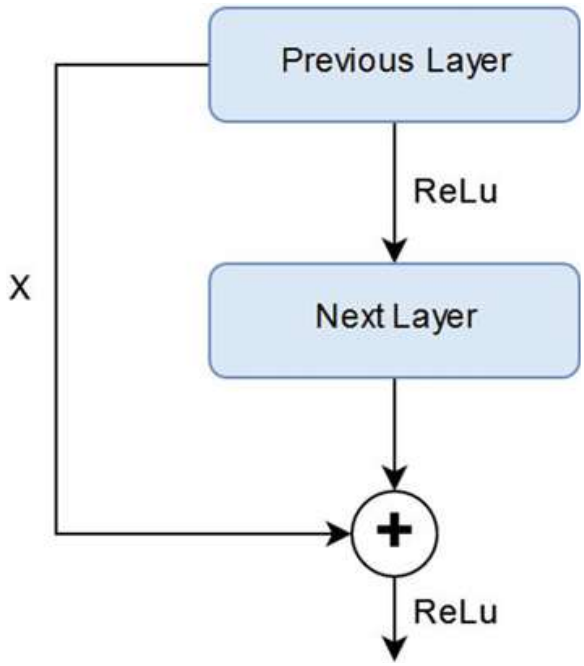

Fig. 2.2 Residual connection in ResNet

eters can make training the network a difficult task. The network depth should be increased without any negative effects. The inception model was refined as Inception V3 in 2016, and as Inception-ResNet in 2017.

ResNet: Microsoft Research Asia proposed a CNN architecture in 2015, which is, 152 layers deep and is called ResNet. ResNet introduced *residual connections* in which the output of a conv-relu-conv series is added to the original input and then passed through Rectified Linear Unit (ReLU) as shown in Fig. 2.2. In this way, the information is carried from the previous layer to the next layer and during backpropagation, the gradient flows easily because of the addition operations, which distributes the gradient. ResNet proved that a complex architecture like *Inception* is not required to achieve the best results but a simple and deep architecture can be tweaked to get better results. ResNet performed good in classification, detection, and localization and won ILSVRC 2015 with an incredible error rate of 3.6% which is better than the human error rate of 5–10%. ResNet is currently deepest network trained on ImageNet and has lesser parameters than VGGNet which is eight times lesser in depth.

Inception-ResNet: A hybrid inception model which uses residual connections, as in ResNet, was proposed in 2017. This hybrid model called Inception-ResNet dramatically improved the training speed of inception model and slightly outperformed the pure ResNet model by a thin margin.

Xception: A convolutional neural network architecture based on depthwise separable convolution layers is called Xception. The architecture is actually inspired by inception model and that is why it is called Xception (Extreme Inception). Xception architecture is a pile of depthwise separable convolution layers with residual connec-

Table 2.1 Classification accuracy of AlexNet, VGG-16, ResNet-152, Inception and Xception on ImageNet

Model	Top-1 accuracy	Top-5 accuracy
AlexNet	0.625	0.86
VGG-16	0.715	0.901
Inception	0.782	0.941
ResNet-152	0.870	0.963
Xception	0.790	0.945

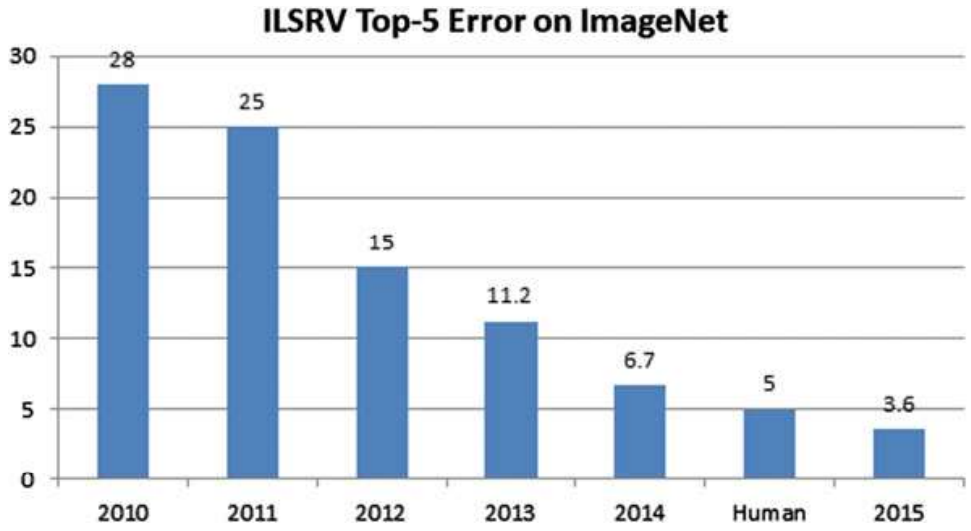

Fig. 2.3 ILSRV top-5 error on ImageNet since 2010

tions. Xception has 36 convolutional layers organized into 14 modules, all having linear residual connections around them, except for the first and last modules. The Xception has claimed to perform slightly better than Inception V3 on ImageNet. Table 2.1 and Fig. 2.3 show classification performance of VGG-16, ResNet-152, Inception V3 and Xception on ImageNet.

SqueezeNet: As the accuracy of new ConvNets models kept on improving, researchers started focusing on how to reduce the size and complexity of the existing ConvNet architectures without compromising on accuracy. The goal was to design a model that has very few parameters, while maintaining high accuracy. A pretrained model was used, and those of its parameters with values below a certain threshold were replaced with zeros to form a sparse matrix followed by few iterations of training on the sparse ConvNet.

Another version of SqueezeNet model used the following three main strategies to reduce the parameters and computational effort significantly while maintaining high accuracy. (a) Replace 3×3 filters with 1×1 filters. (b) Reduce the number of input channels to 3×3 filters. (c) Delay subsampling till late in the network so that convolution layers have large activation maps. SqueezeNet achieved AlexNet-level accuracy on ImageNet with 50 times fewer parameters.

ShuffleNet: Another ConvNet architecture called ShuffleNet was introduced in 2017 for devices with limited computational power, like mobile devices, without compromising on accuracy. ShuffleNet used two ideas, pointwise group convolution and channel shuffle, to considerably decrease the computational cost while maintaining the accuracy.

2.4 Convolution Operation

Convolution is a mathematical operation performed on two functions and is written as $(f * g)$, where f and g are two functions. The output of the convolution operation for domain n is defined as

$$(f * \boldsymbol{g})(n) = \sum_m f(m)\boldsymbol{g}(n-m)$$

For time-domain functions, n is replaced by t. The convolution operation is commutative in nature, so it can also be written as

$$(f * \boldsymbol{g})(n) = \sum_m f(n-m)\boldsymbol{g}(m)$$

Convolution operation is one of the important operations used in digital signal processing and is used in many areas which includes statistics, probability, natural language processing, computer vision, and image processing.

Convolution operation can be applied to higher dimensional functions as well. It can be applied to a two-dimensional function by sliding one function on top of another, multiplying and adding. Convolution operation can be applied to images to perform various transformations; here, images are treated as two-dimensional functions. An example of a two-dimensional filter, a two-dimensional input, and a two-dimensional feature map is shown in Fig. 2.4. Let the 2D input (i.e., 2D image) be denoted by A, the 2D filter of size $m \times n$ be denoted by K, and the 2D feature map be denoted by F. Here, the image A is convolved with the filter K and produces the feature map F. This convolution operation is denoted by $A*K$ and is mathematically given as

$$F(i,j) = (A * K)(i,j) = \sum_m \sum_n A(m,n)K(i-m, j-n) \tag{2.1}$$

The convolution operation is commutative in nature, so we can write Eq. 2.1 as

$$F(i,j) = (A * K)(i,j) = \sum_m \sum_n A(i-m, j-n)K(m,n) \tag{2.2}$$

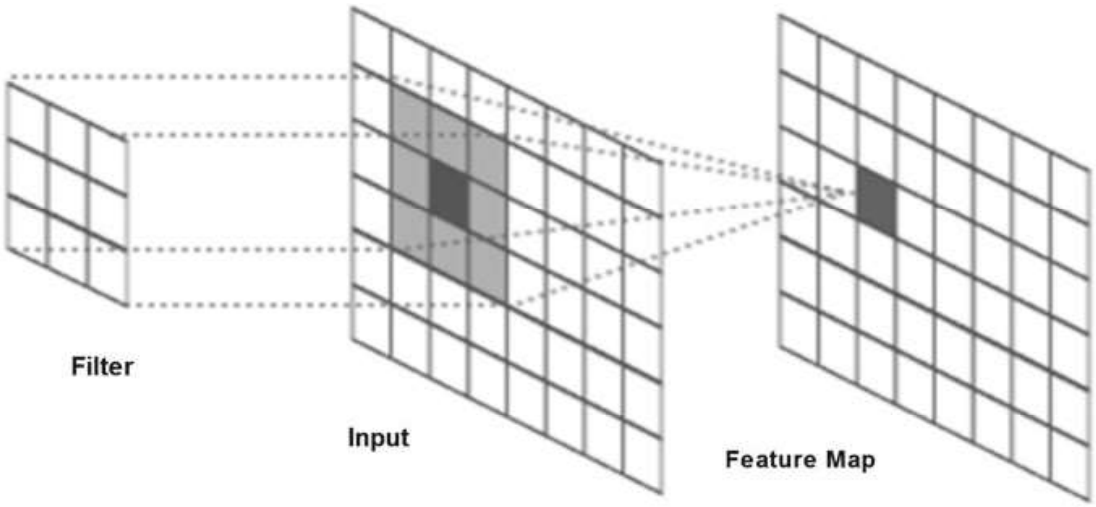

Fig. 2.4 Convolution operation

The kernel K is flipped relative to the input. If the kernel is not flipped, then convolution operation will be same as cross-correlation operation that is given below:

$$F(i, j) = (A * K)(i, j) = \sum_m \sum_n A(i + m, j + n)K(m, n) \tag{2.3}$$

Many CNN libraries use cross-correlation function as convolution function because cross-correlation is more convenient to implement than convolution operation itself. According to Eq. 2.3, the operation computes the inner product (element-wise multiplication) of the filter at every location in the image.

2.5 Architecture of CNN

In a traditional neural network, neurons are fully connected between different layers. Layers that sit between the input layer and output layer are called hidden layers. Each hidden layer is made up of a number of neurons, where each neuron is fully connected to all neurons in the preceding layer. The problem with the fully connected neural network is that its densely connected network architecture does not scale well to large images. For large images, the most preferred approach is to use convolutional neural network.

Convolutional neural network is a deep neural network architecture designed to process data that has a known, grid-like topology, for example, 1D time-series data, 2D or 3D data such as images and speech signal, and 4D data such as videos. ConvNets have three key features: local receptive field, weight sharing, and subsampling (pooling).

(i) **Local Receptive Field**

In a traditional neural network, each neuron or hidden unit is connected to every neuron in previous layer or every input unit. Convolutional neural networks, however, have local receptive field architecture, i.e., each hidden unit can only connect to a small region of the input called local receptive field. This is accomplished by making the filter/weight matrix smaller than the input. With local receptive field, neurons can extract elementary visual features like edges, corners, end points, etc.

(ii) **Weight Sharing**

Weight sharing refers to using the same filter/weights for all receptive fields in a layer. In ConvNet, since the filters are smaller than the input, each filter is applied at every position of the input, i.e., same filter is used for all local receptive fields.

ConvNet consists of a sequence of different types of layers to achieve different tasks. A typical convolutional neural network consists of the following layers:

- Convolutional layer,
- Activation function layer (ReLU),
- Pooling layer,
- Fully connected layer and
- Dropout layer.

These layers are stacked up to make a full ConvNet architecture. Convolutional and activation function layers are usually stacked together followed by an optional pooling layer. Fully connected layer makes up the last layer of the network, and the output of the last fully connected layer produces the class scores of the input image. In addition to these main layers mentioned above, ConvNet may include optional layers like batch normalization layer to improve the training time and dropout layer to address the overfitting issue.

(iii) **Subsampling (Pooling)**

Subsampling reduces the spatial size of the input, thus reducing the parameters in the network. There are few subsampling techniques available, and the most common subsampling technique is max-pooling.

2.5.1 Convolution Layer

Convolution layer is the core building block of a convolutional neural network which uses convolution operation (represented by *) in place of general matrix multiplication. Its parameters consist of a set of learnable filters also known as kernels. The main task of the convolutional layer is to detect features found within local regions of the input image that are common throughout the dataset and mapping their appearance to a feature map. A **feature map** is obtained for each filter in the layer by repeated application of the filter across subregions of the complete image, i.e., *convolving*

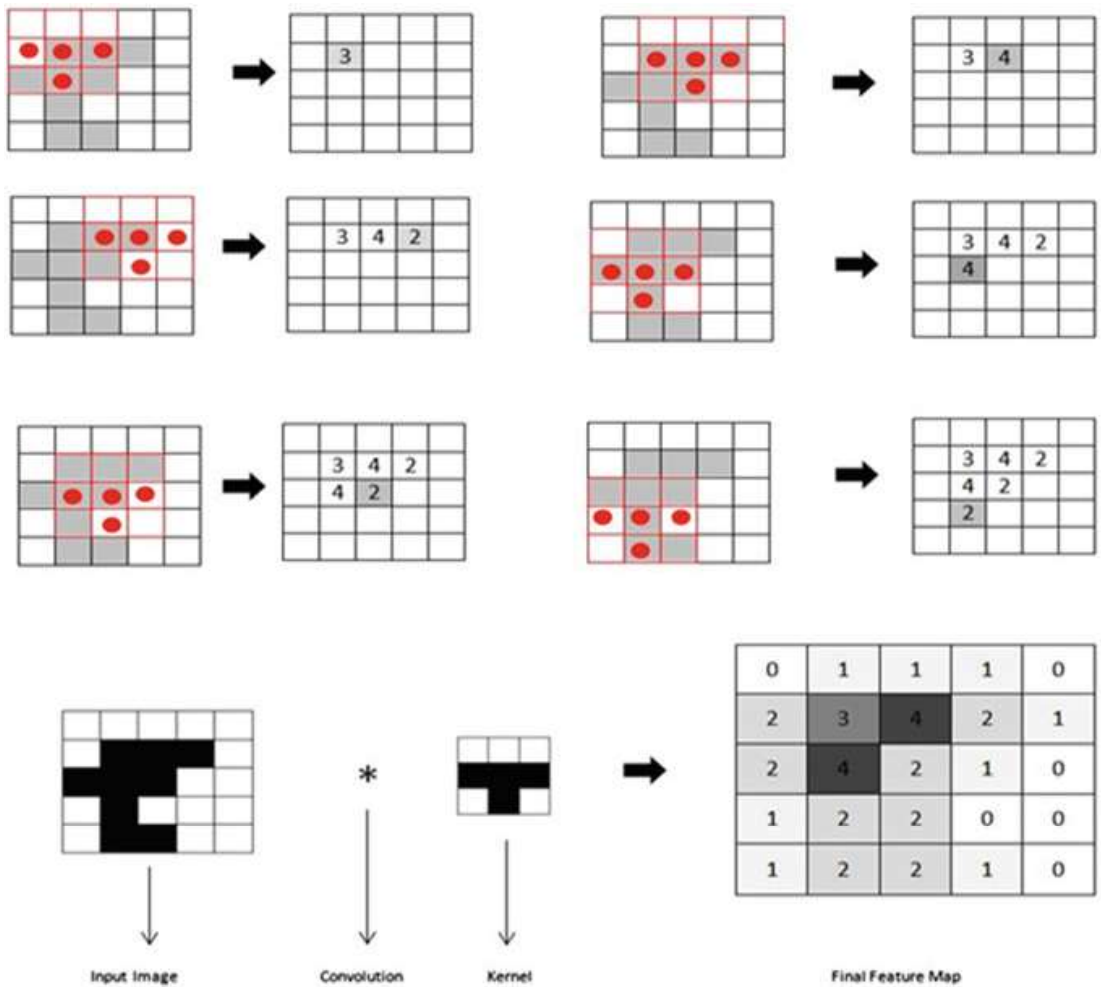

Fig. 2.5 Example of convolution operation

the filter with the input image, adding a bias term, and then applying an activation function. The input area on which a filter is applied is called **local receptive field**. The size of the receptive field is same as the size of the filter. Figure 2.5 shows how a filter (T-shaped) is convolved with the input to get the feature map.

Feature map is obtained after adding a bias term and then applying a nonlinear function to the output of the convolution operation. The purpose of nonlinearity function is to introduce nonlinearity in the ConvNet model, and there are a number of nonlinearity functions available which are briefly explained in the next section.

Filters/Kernels

The weights in each convolutional layer specify the convolution filters and there may be multiple filters in each convolutional layer. Every filter contains some feature like edge, corner, etc. and during forward pass, each filter is slid across the width and height of the input generating feature map of that filter.

Hyperparameters

Convolutional neural network architecture has many hyperparameters that are used to control the behavior of the model. Some of these hyperparameters control the size

of the output while some are used to tune the running time and memory cost of the model. The four important hyperparameters in the convolution layer of the ConvNet are given below:

a. *Filter Size*: Filters can be of any size greater than 2×2 and less than the size of the input but the conventional size varies from 11×11 to 3×3. The size of a filter is independent of the size of input.
b. *Number of Filters*: There can be any reasonable number of filters. AlexNet used 96 filters of size 11×11 in the first convolution layer. VGGNet used 96 filters of size 7×7, and another variant of VGGNet used 64 filters of size 11×11 in first convolution layer.
c. *Stride*: It is the number of pixels to move at a time to define the local receptive field for a filter. Stride of one means to move across and down a single pixel. The value of stride should not be too small or too large. Too small stride will lead to heavily overlapping receptive fields and too large value will overlap less and the resulting output volume will have smaller dimensions spatially.
d. *Zero Padding*: This hyperparameter describes the number of pixels to pad the input image with zeros. Zero padding is used to control the spatial size of the output volume.

Each filter in the convolution layer produces a feature map of size $([A - K + 2P]/S) + 1$ where A is the input volume size, K is the size of the filter, P is the number of padding applied and S is the stride. Suppose the input image has size 128×128, and 5 filters of size 5×5 are applied, with single stride and zero padding, i.e., $A = 128$, $F = 5$, $P = 0$ and $S = 1$. The number of feature maps produced will be equal to the number of filters applied, i.e., 5 and the size of each feature map will be $([128 - 5 + 0]/1) + 1 = 124$. Therefore, the output volume will be $124 \times 124 \times 5$.

2.5.2 Activation Function (ReLU)

The output of each convolutional layer is fed to an activation function layer. The activation function layer consists of an activation function that takes the feature map produced by the convolutional layer and generates the activation map as its output. The activation function is used to transform the activation level of a neuron into an output signal. It specifies the output of a neuron to a given input. An activation function usually has a squashing effect which takes an input (a number), performs some mathematical operation on it and outputs the activation level of a neuron between a given range, e.g., 0 to 1 or -1 to 1.

A typical activation function should be differentiable and continuous everywhere. Since ConvNets are trained using gradient-based methods, an activation function should be differential at any point. However, if a non-gradient-based method is used, then differentiability is not necessary.

There are many of activation functions in use with Artificial Neural Networks (ANNs) and some of the commonly used activation functions are as follows:

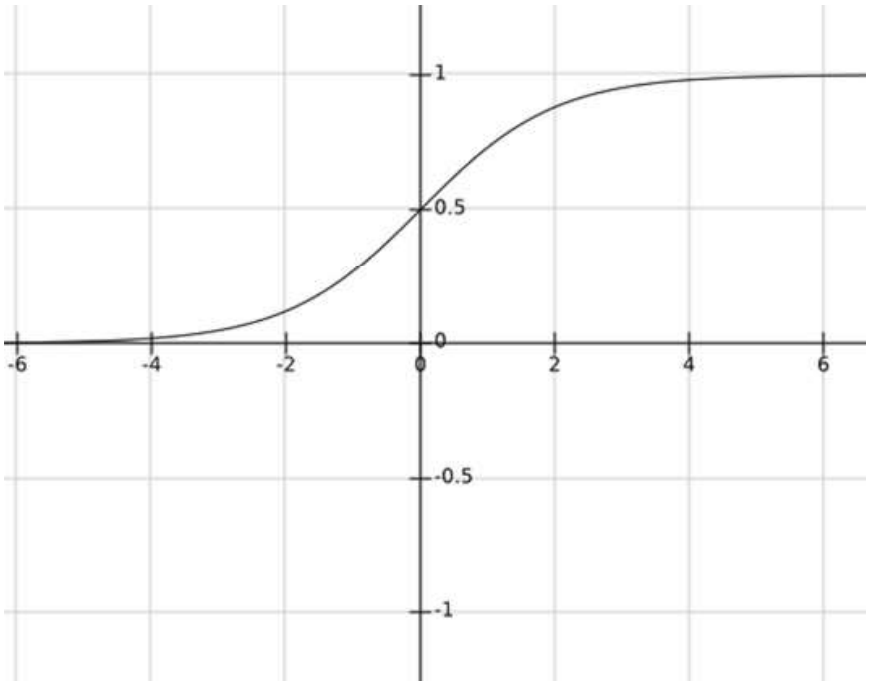

Fig. 2.6 Graph of sigmoid activation function

- **Logistic/Sigmoid Activation Function**: The sigmoid function is mathematically represented as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

It is an S-shaped curve as shown in Fig. 2.6. Sigmoid function squashes the input into the range [0, 1].

- **Tanh Activation Function**: The hyperbolic tangent function is similar to sigmoid function but its output lies in the range $[-1, 1]$. The advantage of tanh over *sigmoid* is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph as shown in Fig. 2.7.
- **Softmax Function** (**Exponential Function**): It is often used in the output layer of a neural network for classification. It is mathematically represented as

$$\sigma\left(x_j\right) = \frac{e^{x_j}}{\sum_{k=1}^{n} e^{x_k}}$$

The softmax function is a more generalized logistic activation function which is used for multiclass classification.

- **ReLU Activation Function**: Rectified Linear Unit (ReLU) has gained some importance in recent years and currently is the most popular activation function for deep neural networks. Neural networks with ReLU train much faster than other

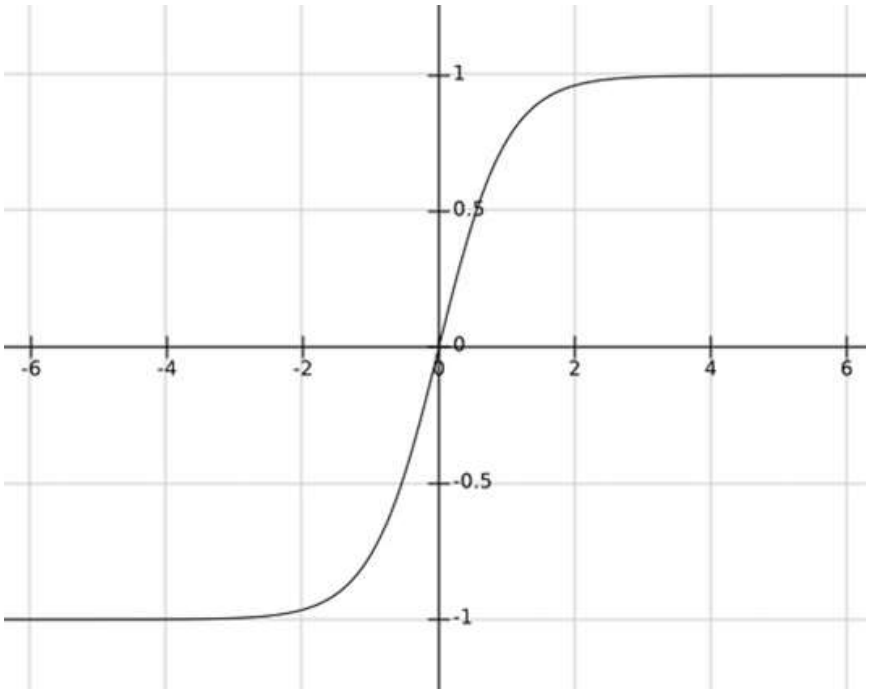

Fig. 2.7 Graph of tanh activation function

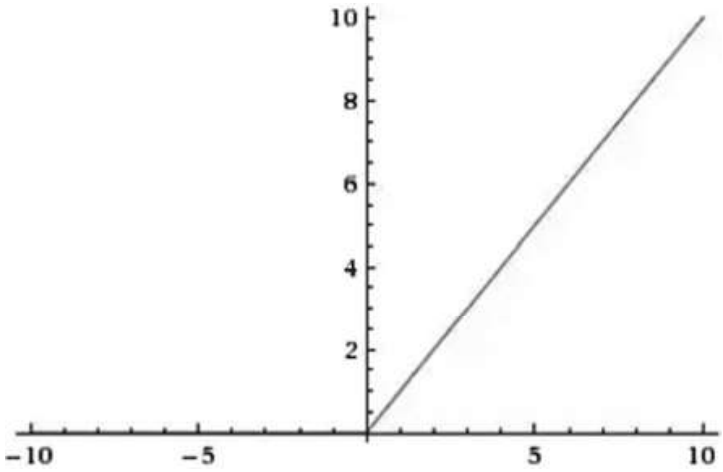

Fig. 2.8 Rectified Linear Unit (ReLU) activation function

activation functions like sigmoid and tanh. ReLU simply computes the activation by thresholding the input at zero. In other words, a rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. It is mathematically given as

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{max}(\mathbf{0}, \boldsymbol{x})$$

Rectified linear unit activation function produces a graph which is zero when $x < 0$ and linear with slope 1 when $x > 0$ as shown in Fig. 2.8.

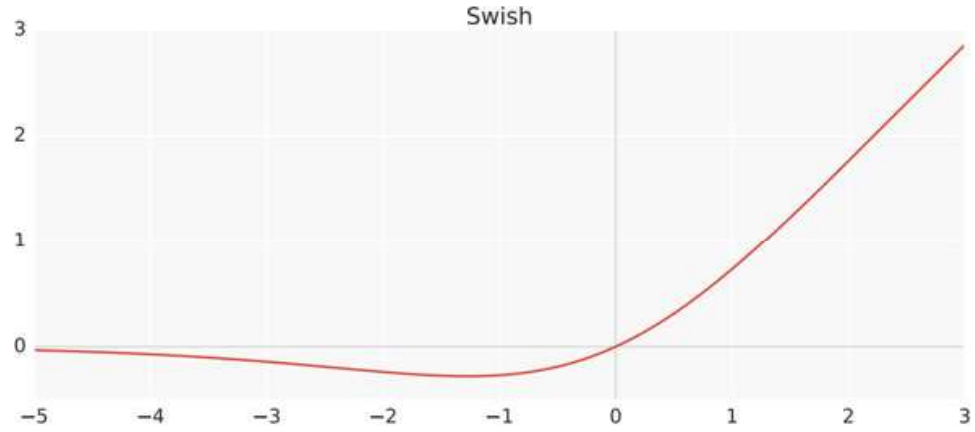

Fig. 2.9 The Swish activation function

- **SWISH Activation Function**:

Self-Gated Activation Function (SWISH) is actually a version of sigmoid function and is given as

$$f(x) = x * \sigma(x)$$

where $\sigma(x)$ is sigmoid of x given as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

SWISH activation function is a non-monotonic function and is shown in Fig. 2.9.

2.5.3 Pooling Layer

In ConvNets, the sequence of convolution layer and activation function layer is followed by an optional pooling or down-sampling layer to reduce the spatial size of the input and thus reducing the number of parameters in the network. A pooling layer takes each feature map output from the convolutional layer and down-samples it, i.e., pooling layer summarizes a region of neurons in the convolution layer. There are few pooling techniques available and the most common pooling technique is max-pooling. Max-pooling simply outputs the maximum value in the input region. The input region is a subset of input (usually 2×2). For example, if input region is of size 2×2, the max-pooling unit will output the maximum of the four values as shown in Fig. 2.10. Other options for pooling layers are average pooling and L2-norm pooling.

Pooling layer operation discards less significant data but preserves the detected features in a smaller representation. The intuitive reasoning behind pooling operation

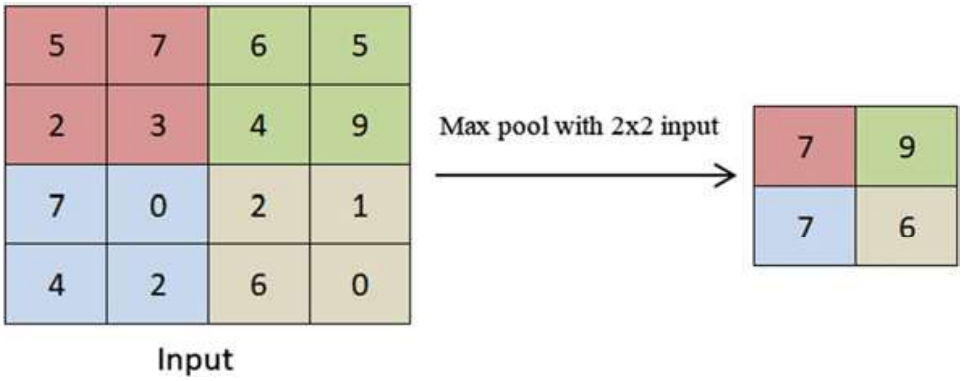

Fig. 2.10 Max-pooling

is that feature detection is more important than feature's exact location. This strategy works well for simple and basic problems but it has its own limitations and does not work well for some problems.

2.5.4 Fully Connected Layer

Convolutional neural networks are composed of two stages: Feature extraction stage and classification stage. In ConvNets, the stack of convolution and pooling layers act as feature extraction stage while as the classification stage is composed of one or more fully connected layers followed by a softmax function layer. The process of convolution and pooling continues until enough features are detected. Next step is to make a decision based on these detected features. In case of classification problem, the task uses the detected features in the spatial domain to obtain probabilities that these features represent each class, that is, obtain the class score. This is done by adding one or more fully connected layers at the end. In fully connected layer, each neuron from previous layer (convolution layer or pooling layer or fully connected layer) is connected to every neuron in the next layer and every value contributes in predicting how strongly a value matches a particular class. Figure 2.11 shows the connection between a convolution layer and a fully connected layer. Like convolutional layers, fully connected layers can be stacked to learn even more sophisticated combinations of features. The output of last fully connected layer is fed to a classifier which outputs the class scores. Softmax and Support Vector Machines (SVMs) are the two main classifiers used in ConvNets. Softmax classifier produces probabilities for each class with a total probability of 1, and SVM which produces class scores and the class having highest score is treated as the correct class.

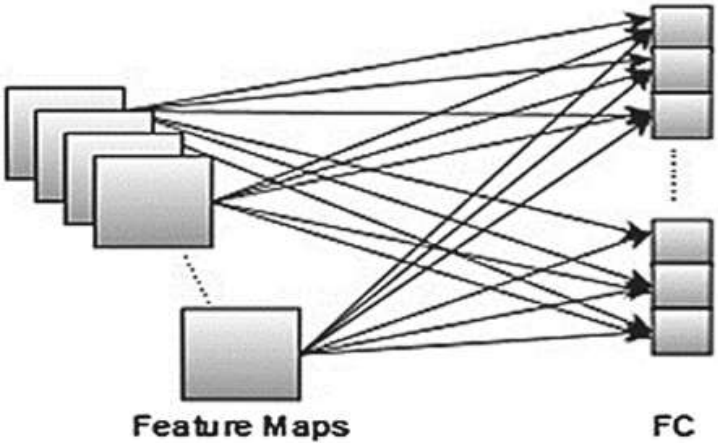

Fig. 2.11 Connection between convolution layer and fully connected layer

2.5.5 Dropout

Deep neural networks consist of multiple hidden layers enabling it to learn more complicated features. It is followed by fully connected layers for decision-making. A fully connected layer is connected to all features, and it is prone to overfitting. Overfitting refers to the problem when a model is trained and it works so well on training data that it negatively impacts the performance of the model on new data. In order to overcome the problem of overfitting, a dropout layer can be introduced in the model in which some neurons along with their connections are randomly dropped from the network during training (See Fig. 2.12). A reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights. Dropout notably reduces overfitting and improves the generalization of the model.

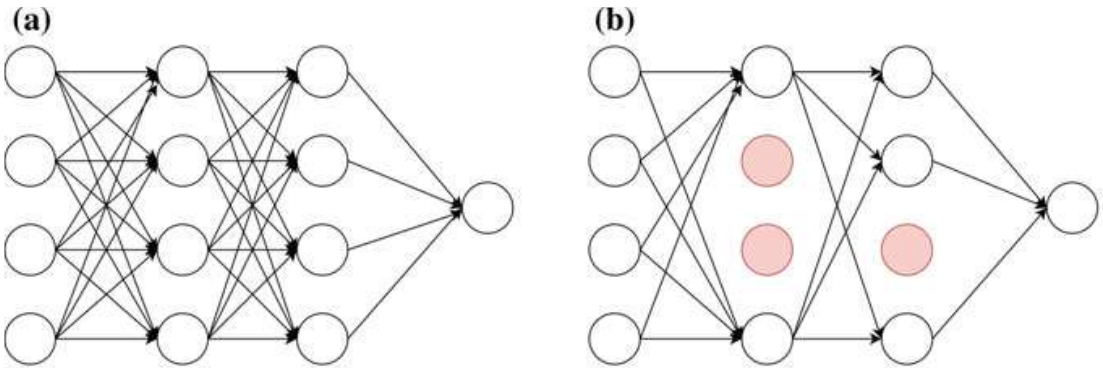

Fig. 2.12 **a** A simple neural network, **b** neural network after dropout