Convolutional Neural Networks

Convolutional Networks

- Neural Networks receive an input (a single vector), and transform it through a series of hidden layers.
- Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in previous layer, and where neurons in a single layer function completely independent and do not share any connections.
- The last fully-connected layer is called “output layer” and in classification settings it represents the class scores.
- Regular Neural Nets don’t scale well to full images.

Convolutional Networks

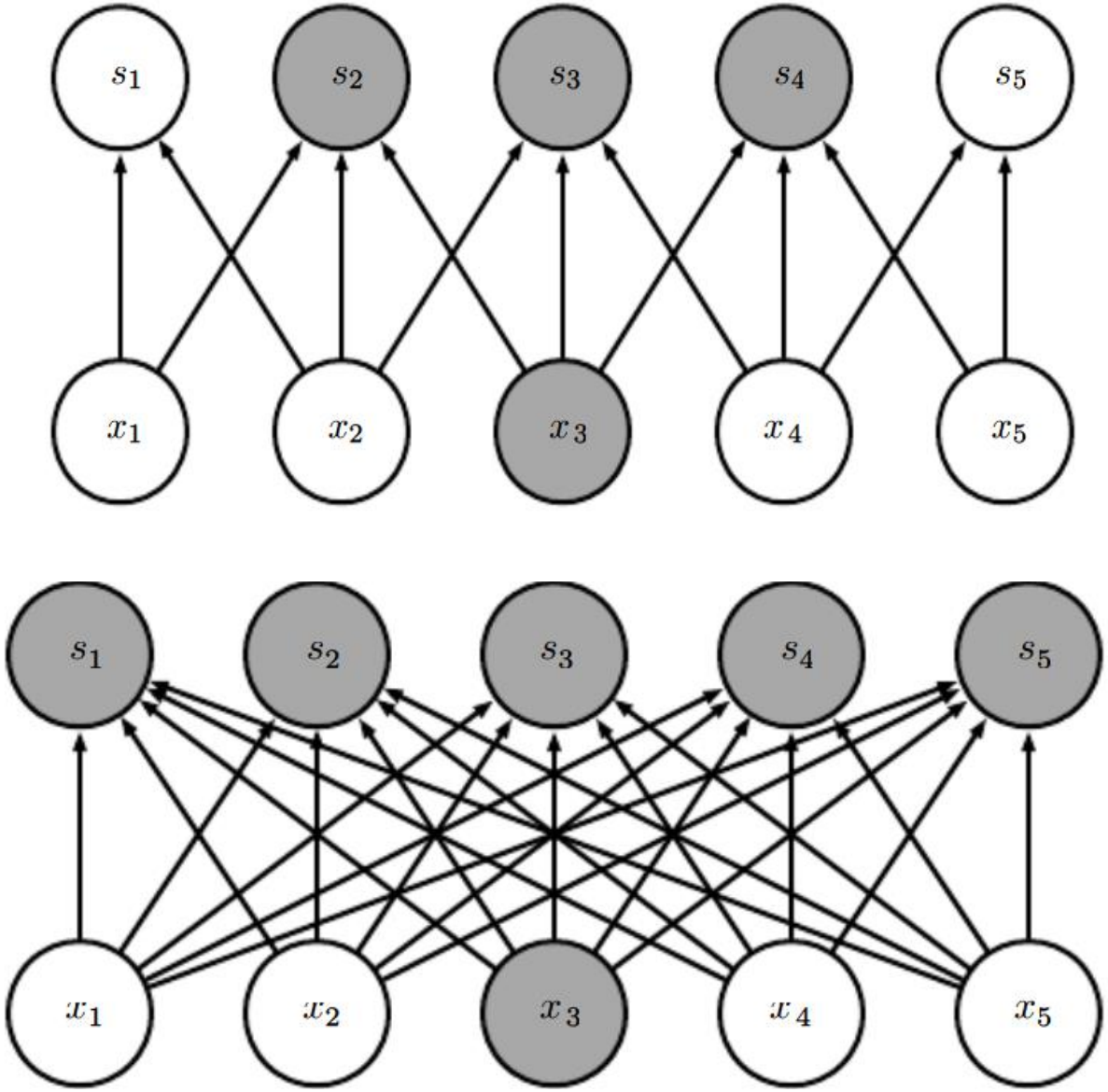
- Convolutional networks (LeCun, 1989), also known as convolutional neural networks (CNNs), are a specialized kind of neural networks for processing data that has a known, grid-like topology.
- Examples include time-series data that can be thought of as a 1D grid taking samples at regular time intervals, and image data that can be thought of as a 2D grid of pixels.
- Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- Convolution leverages three important ideas that can help improve a machine learning system:
  - **sparse interactions**
  - **parameter sharing**
  - **equivariant representations.**

Convolutional Networks

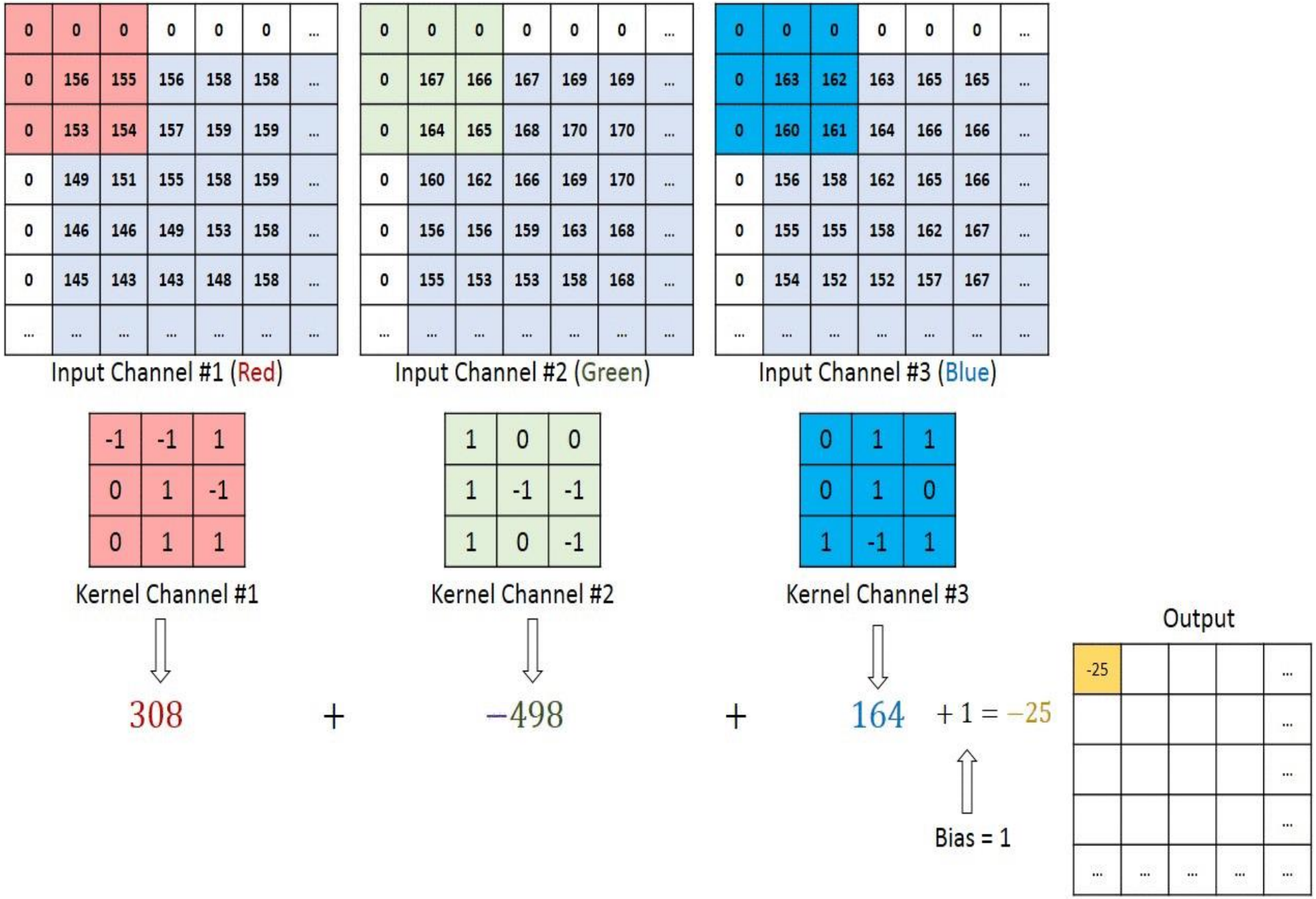
a. Sparse interactions

- Convolutional networks have **sparse interactions**, also called as sparse connectivity or sparse weights.
- It is done by making kernel smaller than the input, e.g., in processing an image, the image might have thousands or millions of pixels, but we can detect small, meaningful features such as edges with kernels that occupy only tens or hundreds of pixels.
- We need to store fewer parameters that both reduces memory requirements and improves its statistical efficiency, which are usually quite large.
- For m input & n output, matrix mult. needs m×n paramtrs & algorithm has O(m×n) runtime per example.
- Limiting no. of connections each output to have as ***k***, the sparsely connected approach requires only k×n parameters and O(k×n) runtime per example.

Convolutional Networks



Convolutional Networks



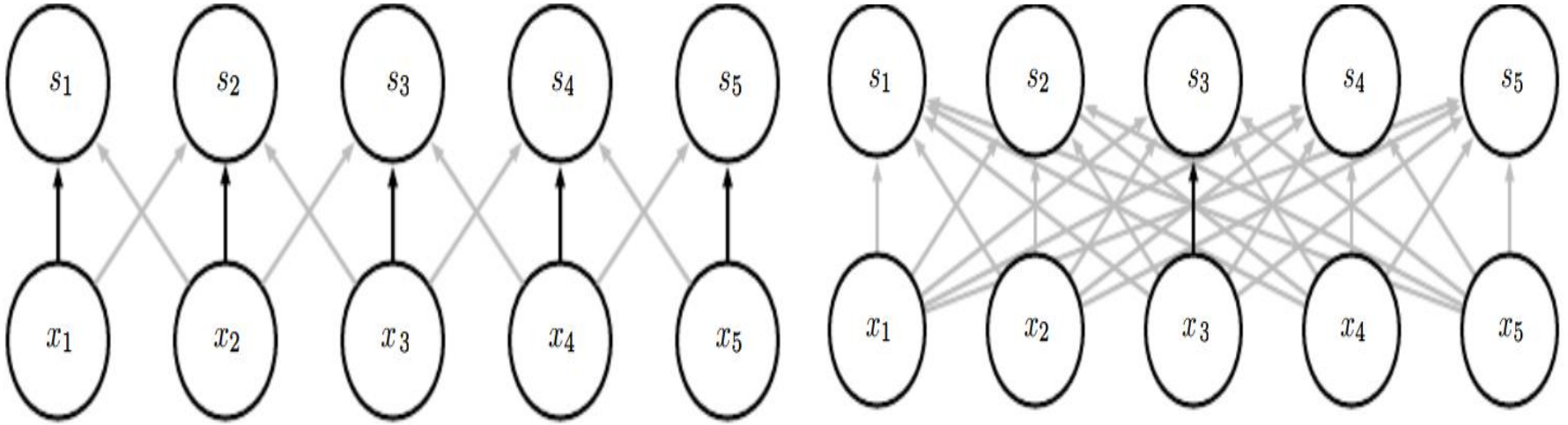
Convolutional Networks

b. Parameter sharing

- It is used in Conv Layers to control the number of parameters. In a real-world example, let there be 55*55*96 = 290,400 neurons in 1st Conv Layer, and each has 11*11*3 = 363 weights and 1 bias. Together, this adds up to 290400 * 364 = 105,705,600 parameters on 1st layer of ConvNet alone. Clearly, this number is very high.
- In parameter sharing, a network has tied weights, as the value of weight applied to one input is tied to the value of a weight applied elsewhere.
- In a convolutional neural net, each member of the kernel is used at every position of the input (except perhaps some of the boundary pixels, depending on the design decisions regarding the boundary).

Convolutional Networks

- Parameter sharing used by convolution means that rather than learning a separate set of parameters for every location, we learn only one set.
- It does not affect the runtime of forward propagation, but it helps to reduce the storage of model to k parameters.
- Since ***k*** is several orders of magnitude less than m and, m & n are usually roughly the same size, ***k*** is practically insignificant compared to m×n. Thus, it is dramatically more efficient than the dense matrix multiplication.



Convolutional Networks

c. Equivariant Representation

- In convolution, the particular form of parameter sharing causes the layer to have a property called equivariance to translation.
- *A function is equivariant whenever if the input changes, the output changes in the same way, for example, a function f(x) is equivariant to a function g if f(g(x))=g(f(x)).*
- Let *g* be a function that translates input, i.e., shifts it, the convolution function is equivariant to *g.* For eample, let *I* be a brightness function of an image, and g be a function mapping one image to another image, such that $I' = g(I)$ is image with $I'(x, y) = I(x - 1, y)$, i.e., g shifts each pixel of *I* one unit to right.
- Appling this transformation to *I*, and then convolution, the result will be the same as if we apply convolution to *I',* followed by *g* to the output.

Convolutional Networks

- Convolution is not naturally equivariant to some other transformations, such as changes in the scale or rotation of an image.
- Other mechanisms are necessary for handling these kinds of transformations.
- Some kinds of data cannot be processed by neural networks defined by matrix multiplication with a fixed-shape matrix.
- Convolution enables processing of some of these kinds of data.
- A ConvNet is a sequence of layers, each transforms one volume of activations to another through a differentiable function.

Convolutional Networks

There are 3 main types of layers in a conv. Neural network: Conv. Layer, Pooling Layer, and Fully-Connected Layer

a. Convolutional Layer

- This layer is the core building block of a CNN that does most of the computational heavy lifting.
- CONV layer's parameters consist of a set of learnable filters.
- Each filter is small spatially (along width and height), but extends through the full depth of the input volume, e.g., a typical filter on 1st layer of a ConvNet may have size 5x5x3.
- During forward pass, we slide (convolve) each filter across width & height of input volume and compute dot products between the entries of the filter and input at any position.
- As we slide filter over the width and height of input volume we produce a 2-D activation map (feature map) that gives the responses of that filter at every spatial position.

Convolutional Networks

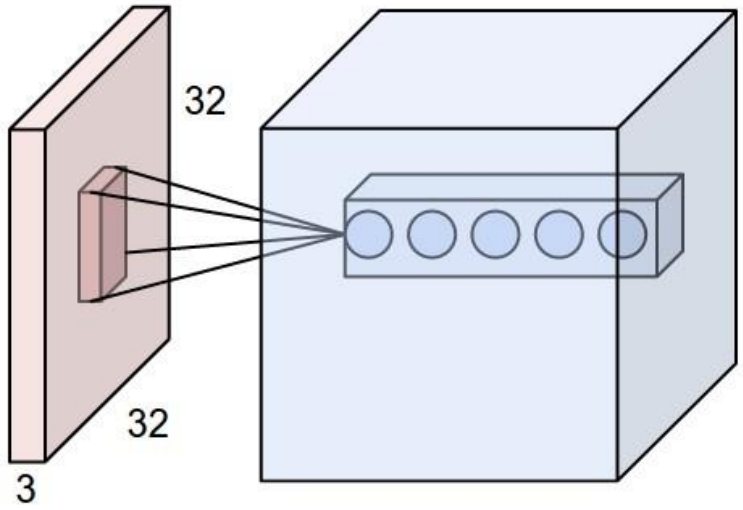
- Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network.
- We will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-D activation map.
- We stack these activation maps along the depth dimension and produce the output volume.
- The neuron connectivities, their arrangement in space, and their parameter sharing scheme are important in CNNs.
- Local Connectivity: When dealing with high-dim. inputs such as images, it is impractical to connect neurons to all neurons in previous volume; rather we connect each neuron to only a local region of input volume.

Convolutional Networks

- The spatial extent of this connectivity is a hyperparameter, called the receptive field of the neuron, which is filter size.
- The extent of connectivity along the depth axis is always equal to the depth of the input volume.
- Connections are local in 2D space (along width and height), but always full along the entire depth of input volume.
- Example1. Let input volume have size [32x32x3], (e.g. an RGB image). If the receptive field (or filter size) is 5x5, then each neuron in Conv Layer will have weights to a [5x5x3] region in the input volume, for a total of 5*5*3 = 75 weights (and +1 bias parameter). Notice that the extent of the connectivity along the depth axis must be 3, since this is the depth of the input volume.
- Example2. let an input volume have size [16x16x20]. Then using receptive field size of 3x3, every neuron in Conv Layer would have a total of 3*3*20 = 180 connections to input volume. The connectivity is local in 2D space (e.g. 3x3), but full along the input depth (20).

Convolutional Networks

- An example input volume in red (e.g. a 32x32x3 image), and an example volume of neurons in first Conv. layer.
- Each neuron in Conv. layer is connected only to a local region in input volume spatially, but to the full depth. There are multiple neurons (5 in here) along the depth, all looking at the same region in input: lines that connect this column of 5 neurons do not represent weights (i.e. these 5 neurons do not share the same weights, but they are associated with 5 different filters), they just indicate that these neurons are connected to or looking at the same receptive field or region of the input volume, i.e. they share the same receptive field but not the same weights.



Convolutional Networks

- Three hyperparameters control the size of output volume: depth, stride and zero-padding.
- The depth of output volume is a hyperparameter: it corresponds to no. of filters we would like to use, each learning to look for something different in the input.
- For example, if first Conv. layer takes as input the raw image, then different neurons along the depth dim. may activate in presence of various oriented edges, or blobs of color.
- We refer to a set of neurons that are all looking at the same region of the input as a depth column (some people prefer the term fibre).
- Stride refers to sliding the filter. When the stride is 1 then we move the filters one pixel at a time. When stride is 2 then the filters jump 2 pixels at a time. This produces smaller output volumes spatially.

Convolutional Networks

- Sometimes it is convenient to pad the input volume with zeros around the border. The size of this zero-padding is a hyperparameter.
- The nice feature of zero padding is that it allows us to control the spatial size of the output volumes.
- We can compute the spatial size of the output volume as a function of the input volume size (*W*), receptive field size of Conv Layer neurons (F), stride with which they are applied (S), and amount of zero padding used (P) on the border. The number of neurons that "fit" is (W − F + 2P)/S + 1.
- For example, for a 7x7 input and 3x3 filter with stride 1 and pad 0, we would get a 5x5 output. With stride 2 we would get a 3x3 output.
- We cannot take any stride as it may lead to fraction neurons, which in not possible.

Convolutional Networks

- Constraints on strides: Spatial arrangement hyperparameter have mutual constraints.
- For example, for input size W=10, no zero-padding (P=0), and filter size F=3, stride S=2, we have (W−F+2P)/S+1 = (10−3+0)/2+1 = 4.5, i.e. not an integer, indicating that neurons don't "fit" neatly & symmetrically across the input.
- So, this setting of the hyperparameters is invalid, and a ConvNet library could throw an exception or zero pad the rest to make it fit, or crop the input to make it fit.
- If all neurons in a single depth slice are using the same weight vector, then the forward pass of CONV layer can in each depth slice be computed as a convolution of the neuron's weights with the input volume (Hence the name: Convolutional Layer).
- This is why it is common to refer to the sets of weights as a filter (or a kernel), that is convolved with the input.

Convolutional Networks

b. POOLING

- Similar to Conv. layer, the pooling layer is responsible for reducing the spatial size of the convolved feature.
- It helps to **decrease computational complexity required to process the data** through dimensionality reduction.
- It is also useful for **extracting dominant features,** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.
- Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation and hence to also control overfitting.
- This layer summarizes the features present in a region of the feature map generated by a convolution layer.
- So, further operations are performed on the summarised features instead of precisely positioned features generated by the convolution layer.

Convolutional Networks

- A typical layer of a Conv network consists of 3 stages.
- In first stage, the layer performs several convolutions in parallel to produce a set of linear activations.
- In second stage, each linear activation is run through a nonlinear activation function, such as ReLU activation function. This stage is also called detector stage.
- In 3rd stage, we use a pooling function to modify the output of the layer further. Following are the types of pooling.
- **Max Pooling:** It is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.
- So, the output after max-pooling is a feature map having most prominent features of the previous feature map.
- **Average Pooling:** It computes the average of the elements present in the region of feature map covered by the filter.

Convolutional Networks

- So, while max pooling gives the most prominent feature in a particular patch of the feature map, the average pooling gives the average of features present in the patch.
- **Global Pooling:** It reduces each channel in the feature map to a single value.
- So, an $\mathbf{n_h \times n_w \times n_c}$ feature map is reduced to $\mathbf{1 \times 1 \times n_c}$ feature map.
- This is equivalent to using a filter of dimensions $\mathbf{n_h \times n_w}$ i.e. the dimensions of feature map.
- Further, it can be either global max pooling or global average pooling.
- A pooling function replaces the output of network at a certain location with a summary statistic of the nearby outputs.
- For example, the max pooling operation reports maximum output within a rectangular neighborhood.

Convolutional Networks

- A pooling layer accepts:
  - a volume of size W1×H1×D1
  - requires two hyperparameters: spatial extent F & stride S
- It produces a volume of size W2×H2×D2,

  where: W2 = (W1−F)/S+1, H2 = (H1−F)/S+1, D2 = D1
- It introduces zero parameters since it computes a fixed function of the input.
- For pooling layers, it is not common to pad the input using zero-padding.
- Other popular pooling functions include the average of a rectangular neighborhood, L2 norm of a rectangular neighborhood, or a weighted average based on the distance from the central pixel.

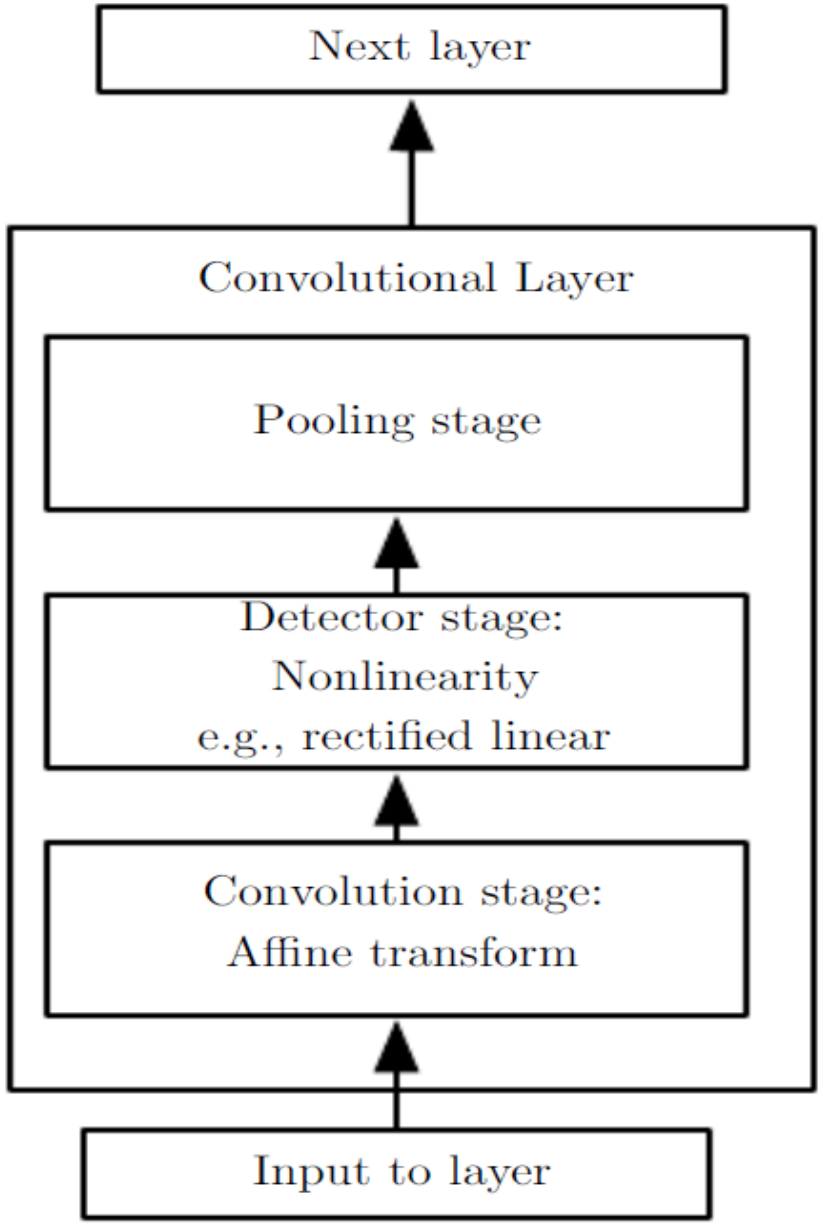
Convolutional Networks

- In all cases, the pooling helps making the representation to become approximately invariant to small translations of the input, which means that if we translate the input by a small amount, the values of most of the pooled outputs do not change.

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Convolutional Networks



Convolutional Networks

- In some contexts, it is more important to preserve the location of a feature, for example, if we want to find a corner defined by two edges meeting at a specific orientation, we need to preserve the location of edges well enough to test whether they meet.
- The use of pooling can be viewed as adding an infinitely strong prior that the function the layer learns must be invariant to small translations. As a result, it can greatly improve the statistical efficiency of the network.
- We can imagine a convolutional net as being similar to a fully connected network, but with an infinitely strong prior over its weights.
- This infinitely strong prior says that the weights for one hidden unit must be identical to the weights of its neighbor, but shifted in space.

Convolutional Networks

- This prior also says that the weights must be zero, except for in the small, spatially contiguous receptive field assigned to that hidden unit.
- Overall, we can think of the use of convolution as introducing an infinitely strong prior probability distribution over the parameters of a layer.
- This prior says that the function the layer should learn contains only local interactions and is equivariant to the translation.
- One key insight is that the convolution and pooling can cause underfitting.
- Like any prior, the convolution and pooling are only useful when the assumptions made by the prior are reasonably accurate. If a task relies on preserving precise spatial information, then using pooling on all features can increase the training error.

Convolutional Networks

c. Fully connected Layer

- Adding a fully-connected layer is a (usually) cheap way of learning non-linear combinations of high-level features as represented by the output of the convolutional layer.
- Fully-connected layer is learning a possibly non-linear function in that space.
- We have converted our input image into a suitable form for our multi-level perceptron, we flatten the image into a column vector.
- The flattened output is fed to a feed-forward neural network and backpropagation is applied to every iteration of training.
- The model is able to distinguish between the dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

Convolutional Networks

- There are various architectures of CNNs available that have been key in building algorithms.
- Some of them are listed below:
  - LeNet
  - AlexNet
  - VGGNet
  - ResNet
  - ZFNet
  - GoogLeNet

Convolutional Networks

LeNet – It is among the first published CNNs to capture wide attention for its performance on computer vision tasks.

- In 1989, LeCun's team published the first study to successfully train CNNs via backpropagation (LeCun et al., 1989).
- At a high level, LeNet (LeNet-5) consists of two parts:

  (i) a conv. encoder consisting of two convolutional layers; 1st conv. layer has 6 output channels, while 2nd layer has 16.

  (ii) a dense block consisting of 3 fully connected layers.
- Each pooling operation (stride 2) reduces dimensionality by a factor of 2 via spatial downsampling.
- The convolutional block emits an output with shape given by (batch size, number of channel, height, width).

Convolutional Networks

AlexNet

- It has 8 layers with learnable parameters.
- The input to the model is RGB images.
- It has 5 convolution layers with a combination of max-pooling layers.
- Then it has 3 fully connected layers.
- The activation function used in all layers is Relu.
- It uses two Dropout layers.
- The activation function used in the output layer is Softmax.
- Total number of parameters in this architecture is 62.3 million.

Convolutional Networks

VGGNet

- VGG stands for Visual Geometry Group; it has two variants: VGG-16 and VGG-19.
- The number of layers with VGG-16 or VGG-19 consist of 16 and 19 convolutional layers, respectively.
- This architecture is the basis of ground-breaking object recognition models.
- VGG's **convolutional layers** leverage a minimal receptive field, i.e., 3×3, the smallest possible size that still captures up/down and left/right.
- There are also 1×1 convolution filters acting as a linear transformation of the input.
- This is followed by a ReLU unit, which is a huge innovation from AlexNet that reduces training time.
- The stride is one pixel to keep the spatial resolution preserved after convolution.

Convolutional Networks

- **Hidden layers:** All the hidden layers in VGG network use ReLU.
- It does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time.
- **Fully-Connected Layers:** It has three fully connected layers.
- Out of three layers, the first two have 4096 channels each, and the third has 1000 channels, 1 for each class.

Convolutional Networks

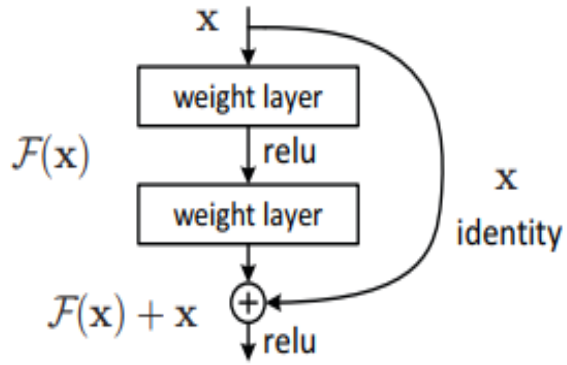
ResNet

- Residual Network (ResNet), introduced by Ren et al. in 2015, is one of the popular and most successful deep learning models so far.
- ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance.
- Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper.
- While AlexNet had only 5 conv. layers, the VGG network and GoogleNet (also codenamed Inception_v1) had 19 and 22 layers respectively.
- However, increasing network depth does not work by simply stacking layers together.

Convolutional Networks

- Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small.
- As a result, as network goes deeper, its performance gets saturated or even starts degrading rapidly.
- Before ResNet, there had been several ways to deal the vanishing gradient issue, for instance, some work adds an auxiliary loss in a middle layer as extra supervision, but none seemed to really tackle the problem once and for all.
- The core idea of ResNet is introducing a so-called “identity shortcut connection” .
- The problem of training very deep networks has been relieved with the introduction of Residual blocks and the ResNet model is made up of these blocks.

Convolutional Networks



- In this figure, it can be noticed that there is a direct connection that skips some layers of the model. This connection is called 'skip connection' and is the heart of residual blocks.
- The output is not the same due to this skip connection.
- Without the skip connection, input 'X gets multiplied by the weights of the layer followed by adding a bias term.
- Then comes the activation function, f() and we get the output as H(x). **H(x)=f(wx + b) or H(x)=f(x)**

Convolutional Networks

- With the introduction of a new skip connection technique, the output is H(x) is changed to **H(x)=f(x)+x**
- But the dimension of the input may be varying from that of the output which might happen with a conv. layer or pooling layers.
- This problem can be handled with two approaches:
  - Zero is padded with the skip connection to increase its dimensions.
  - 1×1 convolutional layers are added to the input to match the dimensions.
- In such a case, the output is: **H(x)=f(x)+w1.x**
- Here an additional parameter w1 is added but no parameter is added when using first approach.
- These skip connections technique in ResNet solves the problem of vanishing gradient in deep CNNs by allowing alternate shortcut path for the gradient to flow through.

Convolutional Networks

ZFNet : It takes input as **224x224x3** images.

- Next, **96 convolutions of 7x7** with a **stride of 2** are performed, followed by **ReLU** activation, **3x3 max pooling with stride 2** and **local contrast normalization**.
- Followed by it are **256 filters of 3x3** each which are then again **local contrast normalized** and **pooled**, as follows:

$$\sqrt{\frac{1}{3rc}\sum_{i=1}^{r}\sum_{j=1}^{c}\sum_{k=1}^{3}(x_{i,j,k}-\bar{x})^2} \qquad \text{where} \qquad \bar{x}=\frac{1}{3rc}\sum_{i=1}^{r}\sum_{j=1}^{c}\sum_{k=1}^{3}x_{i,j,k}$$

- The third and fourth layers are identical with **384 kernels of 3x3 each**.
- The fifth layer has **256 filters of 3x3, followed by 3x3 max pooling with stride 2** and **local contrast normalization**.
- The sixth and seventh layers house **4096 dense** units each.
- Finally, we feed into a **Dense layer of 1000** neurons i.e. the number of classes in ImageNet.

Convolutional Networks

GoogleNet

- Researchers discovered that an increase of layers and units within a network led to a significant performance gain.
- But increasing the layers to create more extensive networks came at a cost.
- Large networks are prone to overfitting and suffer from either exploding or vanishing gradient problem.
- The GoogLeNet architecture solved most of the problems that large networks faced, mainly through the Inception module's utilisation.
- The Inception module is a neural network architecture that leverages feature detection at different scales through convolutions with different filters and reduced the computational cost of training an extensive network through dimensional reduction.

Convolutional Networks

- The Inception Network was one of the major breakthroughs in the fields of Neural Networks, particularly for CNNs.
- There are 3 versions of Inception Networks, named as Inception Version 1, 2, and 3.
- The first version (2014), as the name "GoogleNet" suggests, developed by Google.
- GoogleNet is 22 layers deep, with 27 pooling layers. There are 9 inception modules stacked linearly in total.
- The ends of the inception modules are connected to the global average pooling layer.
- GoogLeNet architecture presented in the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14) solved computer vision tasks such as image classification and object detection.
- Today it is used for other computer vision tasks such as face detection and recognition, adversarial training etc.