Unit 5
Convolutional Neural Networks

Prepared By: Arjun Singh Saud
Asst. Prof. CDCSIT

Neurons in Human Vision

- Within fractions of seconds, we can identify objects within our field of view, without thought or hesitation.
- Not only can we name objects we are looking at, we can also perceive their depth, perfectly distinguish their outline representation, and separate the objects from their backgrounds.
- Somehow our eyes take in raw 3-D color data, but our brain transforms that information into more meaningful primitives—lines, curves, and shapes—that might indicate, for example, that we're looking at a house or cat.

Neurons in Human Vision

- Foundational to the human sense of vision is the neuron. Specialized neurons are responsible for capturing light information in the human eye.
- This light information is then preprocessed, transported to the visual cortex of the brain, and then finally analyzed to completion. Neurons are single-handedly responsible for all of these functions.

The Shortcomings of Feature Selection

- Feature selection (FS) is an important research topic in the area of data mining and machine learning. FS aims at dealing with the high dimensionality problem.
- It is the process of selecting the relevant features and removing the irrelevant, redundant and noisy ones, intending to obtain the best performing subset of original features.
- In order to use traditional machine learning techniques to teach a computer to “see,” we need to provide our program with a lot more features to make accurate decisions.

The Shortcomings of Feature Selection

- Huge teams of computer vision researchers would take years to debate about the usefulness of different features.
- As the recognition problems became more and more intricate, researchers had a difficult time coping with the increase in complexity.
- In summary, we can say that feature selection is *"NP hard problem"*.

The Shortcomings of Feature Selection

- Huge teams of computer vision researchers would take years to debate about the usefulness of different features.
- As the recognition problems became more and more intricate, researchers had a difficult time coping with the increase in complexity.
- In summary, we can say that feature selection is *"NP hard problem"*.
- Todays deep learning models are capable of selecting useful features from dataset automatically and hence explicit feature selection is not needed.

The Shortcomings of Feature Selection

- CNN are deep learning models that combines convolution and pooling layers with fully connected deep neural network, where convolution layers main responsibility is to select features from the given image.
- Thus rigorous feature selection techniques to select image features in not needed while using CNNs for recognizing and identifying images.

Vanilla Deep Neural Networks Don’t Scale

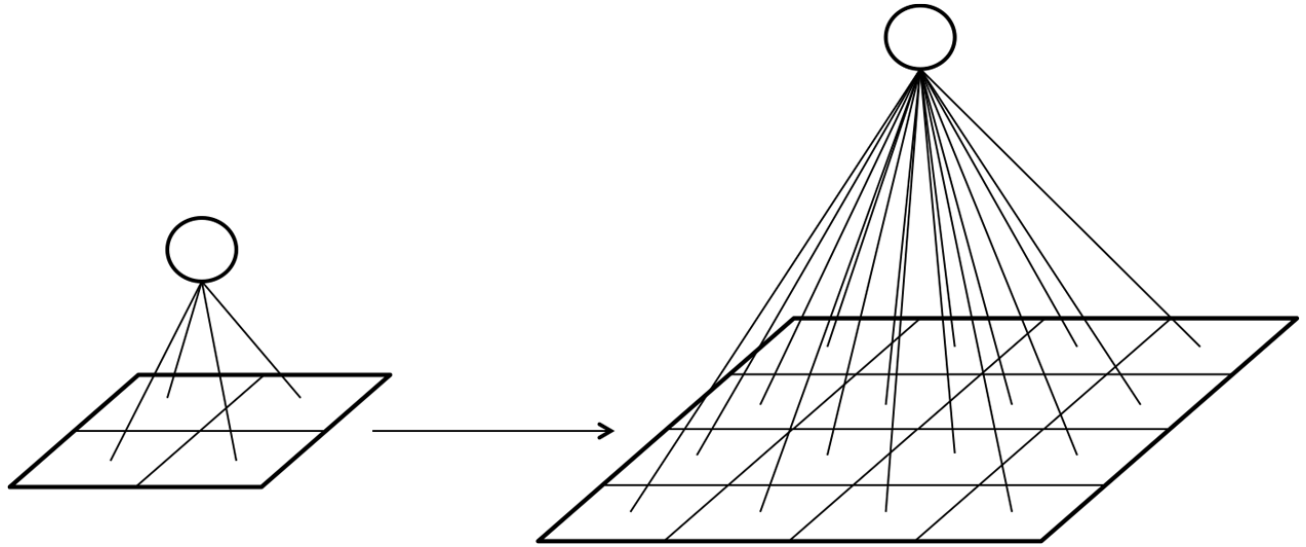
- The fundamental goal in applying deep learning to computer vision is to remove the cumbersome, and ultimately limiting, feature selection process.
- Deep neural networks are perfect for this process because each layer of a neural network is responsible for learning and building up features to represent the input data that it receives.
- A naive approach might be for us to use a vanilla deep neural network to achieve the image classification task.

Vanilla Deep Neural Networks Don't Scale

- If we attempt to tackle the image classification problem in this way, however, we'll quickly face a pretty daunting challenge, visually demonstrated in the Figure of next slide.
- If images are only 28 x 28 pixels and are black and white, a neuron in a fully connected hidden layer would have 784 incoming weights.
- However, it does not scale well as images grow larger. For example, for a full-color 200 x 200 pixel image, input layer would have 200 x 200 x 3 = 120,000 weights.

Vanilla Deep Neural Networks Don’t Scale

- If the neural network have multiple layers, these parameters add up quite quickly. Clearly, this full connectivity is not only wasteful, but also means that we’re much more likely to overfit to the training dataset.

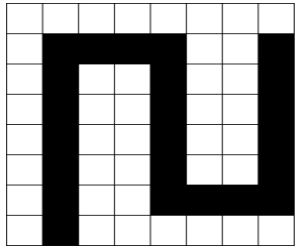


Vanilla Deep Neural Networks Don't Scale

- The convolutional network takes advantage of the fact that we're analyzing images, and sensibly constrains the architecture of the deep network so that we drastically reduce the number of parameters in our model.
- Inspired by how human vision works, layers of a convolutional network have neurons arranged in three dimensions, so layers have a width, height, and depth.
- The neurons in a convolutional layer are only connected to a small, local region of the preceding layer, so we avoid the wastefulness of fully-connected neurons.

Filters and Feature Maps

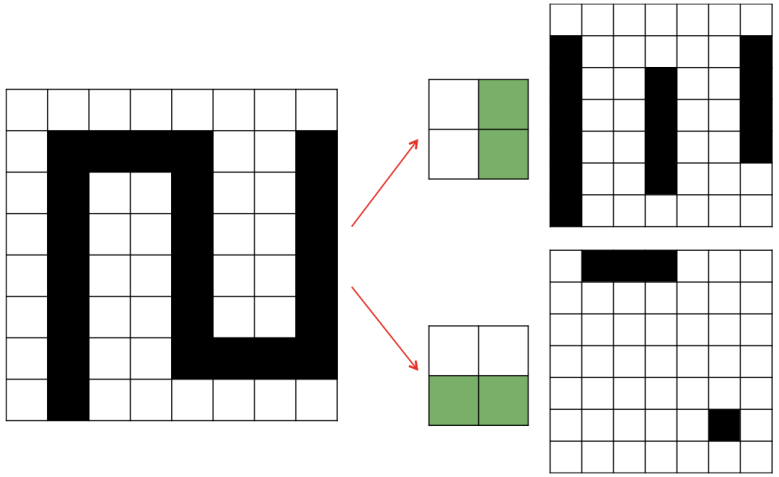
- Filter is essentially a feature detector, and to understand how it works, let's consider the toy image given below.



- Let's say that we want to detect vertical and horizontal lines in the image. One approach would be to use an appropriate feature detector, as shown in Figure of next slide . For example, to detect vertical lines, we would use the feature detector on the top, slide it across the entire image. If we have a match. We keep track of our answers in the matrix in the top right.

Filters and Feature Maps

- If there's a match, we shade the appropriate box black. If there isn't, we leave it white. This result is our *feature map*, and it indicates where we've found the feature we're looking for in the original image.
- We can do the same for the horizontal line detector (bottom), resulting in the feature map in the bottom-right corner.



Filters and Feature Maps

- Different filters can create different feature maps in a CNN by highlighting different patterns or features in the input data.
- We can extract different features from the input data by using different filters with different values. For example, a filter that detects horizontal edges might have values like left matrix.
- Similarly, a filter that detects vertical edges might have values like right matrix.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Filters and Feature Maps

- Generation of feature maps from input image and filters can be shown numerically as below.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

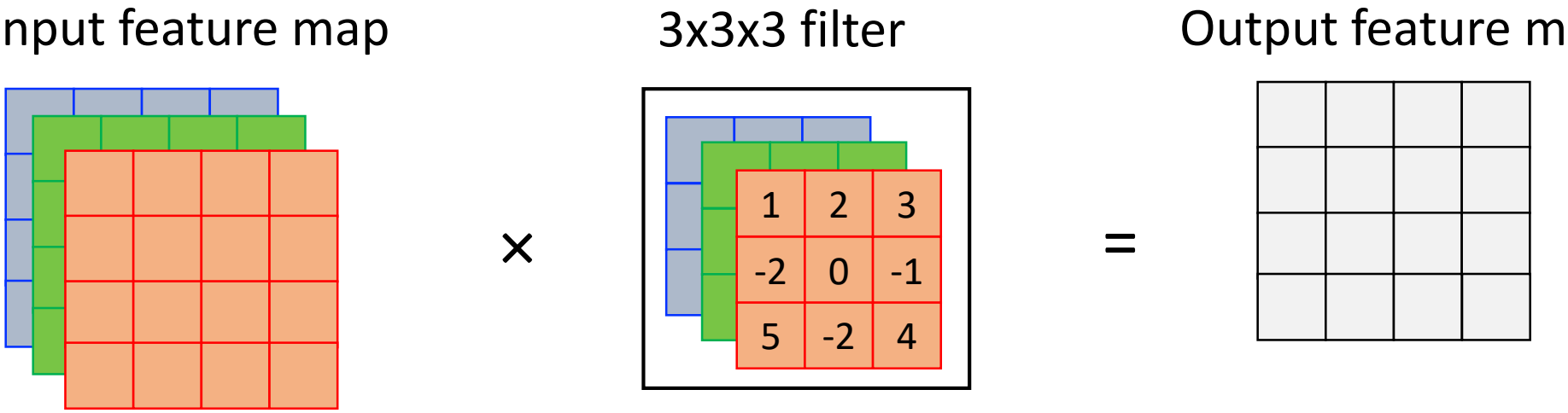
Filter / Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Filters and Feature Maps

- The depth of a filter in a CNN must match the depth of the input image. The number of color channels in the filter must remain the same as the input image.
- We perform the convolution operation described above. The only difference is this time we do the sum of matrix multiply in 3D instead of 2D, but the result is still a scalar.



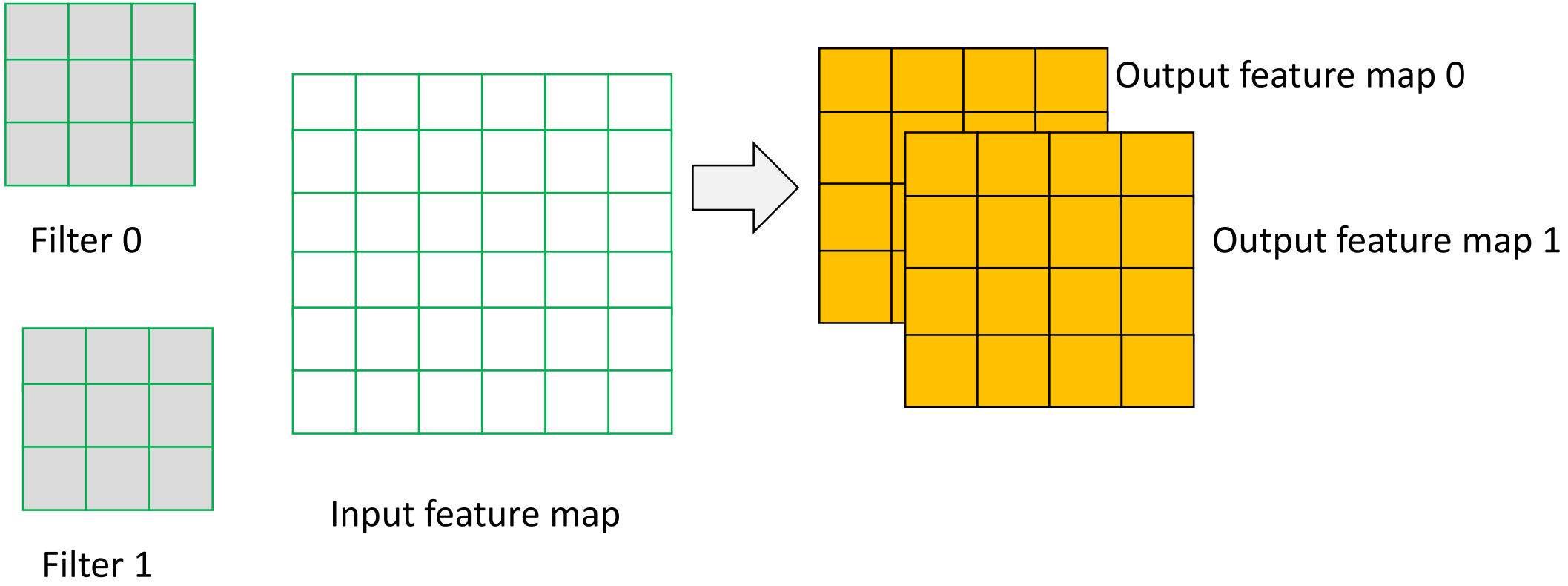
Description of Convolution Layer

- Convolution layer of CNN is responsible for extracting features from input image.
- First, a convolutional layer takes in an input volume. This input volume has the following characteristics: *width, height, depth, and zero padding*.
- This volume is processed by a total of *k* filters. These filters have a number of hyperparameters, which are described as follows:
- Their *spatial extent e*, which is equal to the filter's height and width. Their *stride s*, or the distance between consecutive applications of the filter on the input volume. The *bias b* which is added to each component of the convolution.

Description of Convolution Layer

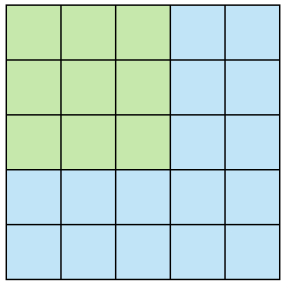
- Filter move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.
- The size of the Feature Map obtained after convolution operation is controlled by three parameters that we need to decide before the convolution step is performed: *depth, stride, and padding*.
- Depth corresponds to the number of filters we use for the convolution operation. If we use 3 different filters, three different feature maps are produced. We can think of these three feature maps as stacked 2d matrices.

Description of Convolution Layer

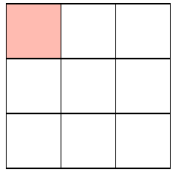


Description of Convolution Layer

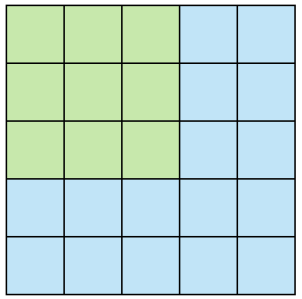
- Stride is the number of pixels by which we slide our filter matrix over the input matrix.
- When the stride is 1 then we move the filters one pixel at a time (left figure). When the stride is 2, then the filters jump 2 pixels at a time (right figure) as we slide them around.
- Having a larger stride will produce smaller feature maps.



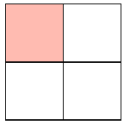
Stride 1



Feature Map



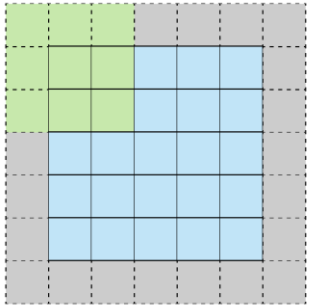
Stride 2



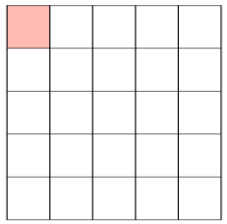
Feature Map

Description of Convolution Layer

- Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.
- A nice feature of zero padding is that it allows us to control the size of the feature maps.
- Adding zero-padding is also called *wide convolution*, and not using zero-padding would be a *narrow convolution*.



Stride 1 with Padding



Feature Map

Description of Convolution Layer

- Size of convolved feature map is given as below:

$$width(w_{out}) = \frac{(w_{in} - e + 2p)}{s} + 1$$

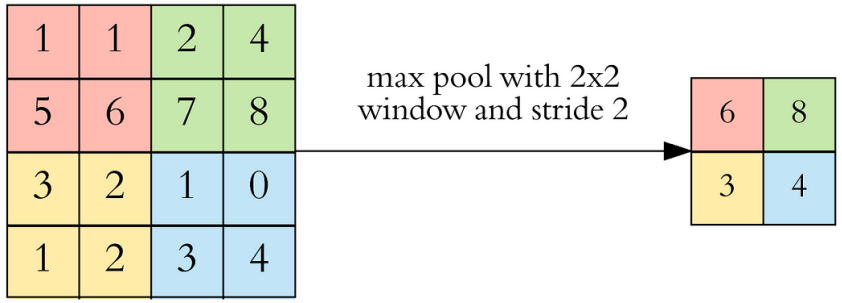
$$height(h_{out}) = \frac{(h_{in} - e + 2p)}{s} + 1$$

Description of Pooling Layer

- After a convolution operation we usually perform pooling to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats overfitting.
- Pooling layers down samples each feature map independently, reducing the height and width, keeping the depth intact.
- The most common type of pooling is max pooling which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters.

Description of Pooling Layer

- It slides a window over its input, and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride.
- Here is the result of max pooling using a 2x2 window and stride 2. Since both the window size and stride are 2, the windows are not overlapping.



Description of Pooling Layer

- The resulting dimensions of each pooled feature map are as follows:

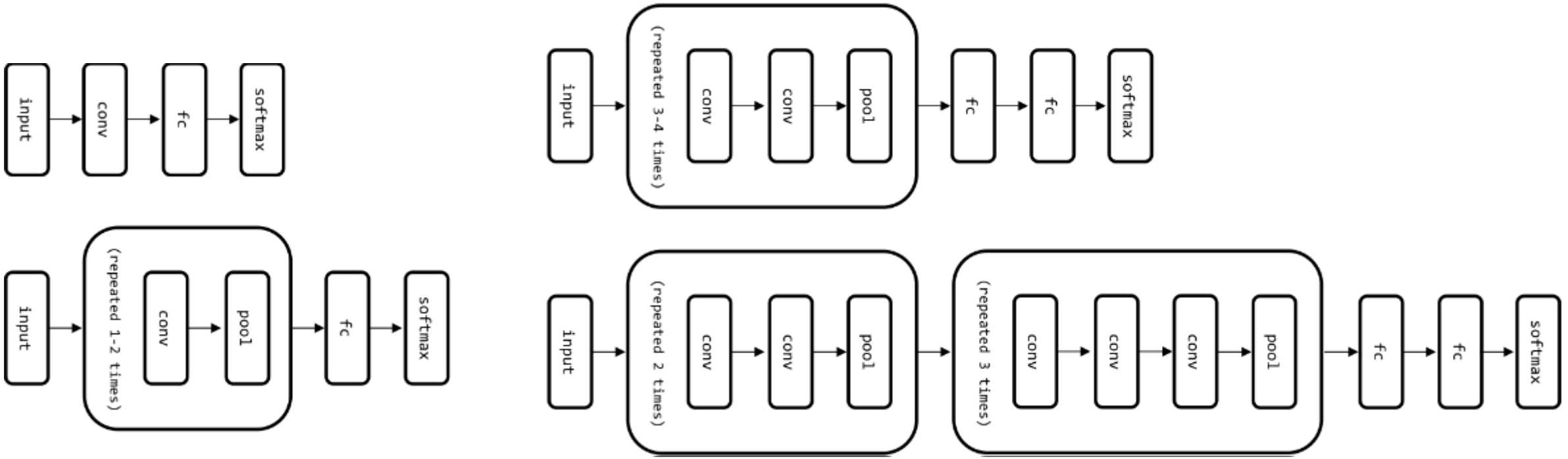
$$width(w_{out}) = \frac{(w_{in} - e)}{s} + 1$$

$$height(h_{out}) = \frac{(h_{in} - e)}{s} + 1$$

Architecture of CNN

- Convolutional Neural Network (CNN) is a deep learning model that is widely used for recognizing and classifying image data.
- CNN is very useful as it minimizes human effort by automatically detecting the features.
- There are several architectures of CNN that are depicted in the figure of next slide.

Architecture of CNN



Architecture of CNN

- One theme we notice as we build deeper networks is that we reduce the number of pooling layers and instead stack multiple convolutional layers.
- This is generally helpful because pooling operations are inherently destructive.
- Stacking several convolutional layers before each pooling layer allows us to achieve richer representations.

Architecture of CNN

- There are three types of layers that make up the CNN which are the *convolutional layers, pooling layers, and fully-connected (FC) layers*.
- When these layers are stacked, a CNN architecture will be formed.
- In addition to these three layers, there are two more important parameters which are the *dropout layer* and the a*ctivation function*.

Architecture of CNN

Convolutional Layer

- This layer is the first layer that is used to extract the various features from the input images.
- In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM.
- By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

Architecture of CNN

Convolutional Layer

- The output is termed as the Feature map which gives us information about the image such as the corners and edges.
- Later, this feature map is fed to other layers to learn several other features of the input image.
- The convolution layer in CNN passes the result to the next layer once applying the convolution operation in the input.
- Convolutional layers in CNN benefit a lot as they ensure the spatial relationship between the pixels is intact.

Architecture of CNN

Pooling Layer

- In most cases, a Convolutional Layer is followed by a Pooling Layer.
- The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs.
- This is performed by decreasing the connections between layers and independently operates on each feature map.
- Depending upon method used, there are several types of Pooling operations. It basically summarizes the features generated by a convolution layer.

Architecture of CNN

Pooling Layer

- In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section.
- The total sum of the elements in the predefined section is computed in Sum Pooling.
- This CNN model generalizes the features extracted by the convolution layer, and helps the networks to recognize the features independently.

Architecture of CNN

Fully Connected Layer

- The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers.
- These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.
- In this, the input image from the previous layers are flattened and fed to the FC layer.

Architecture of CNN

Fully Connected Layer

- The flattened vector then undergoes few more FC layers with aim of leaning non-linear mappings between image features and output.
- In this layer classification process begins to take place. The reason behind using multiple fully connected layers is to achieve better performance than a single FC layer.

Architecture of CNN

Fully Connected Layer

- The flattened vector then undergoes few more FC layers with aim of leaning non-linear mappings between image features and output.
- In this layer classification process begins to take place. The reason behind using multiple fully connected layers is to achieve better performance than a single FC layer.

Examples

- Consider an 8-bit grayscale image having resolution 6x6 and filter of size 3x3. Show convolution operation on the image by assuming stride=1 and No padding.
- Consider an 24-bit RGB image having resolution 5x5 and filters of size 3x3. Show convolution operation on the image by assuming stride=1 and No padding.
- Consider an 24-bit RGB image having resolution 5x5 and filters of size 3x3. Show convolution operation on the image by assuming stride=2 and zero padding.

Image Preprocessing Operations to Enable More Robust Models

- Image preprocessing operations includes, but is not limited to, adjustments to the size, orientation, and color.
- The purpose of pre-processing is to raise the image's quality so that we can analyze it more effectively. Preprocessing allows us to eliminate unwanted distortions and improve specific qualities that are essential for the application we are working on.
- Image preprocessing is also used to reduce the size of images so the deep learning models can be trained and tested in shorter time.

Image Preprocessing Operations to Enable More Robust Models

- Besides this, image preprocessing operation are used to increase the size of training data using various data augmentation techniques.
- Some of the widely used image preprocessing operations are: *rescaling, noise removal, modifying contrast, modifying orientation, modifying color, transpose, modifying saturation etc*.
- Applying these transformations helps us build networks that are robust to the different kinds of variations that are present in natural images, and make predictions with high fidelity in spite of potential distortions.

Accelerating Training with Batch Normalization

- Normalization is a data pre-processing tool used to bring the numerical data to a common scale without distorting its shape. Normalization allows us to use higher value of learning rate and hence accelerates training to DNNs.
- Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change.
- This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models.

Accelerating Training with Batch Normalization

- Batch-Normalization (BN) is an algorithmic method which makes the training of Deep Neural Networks (DNN) faster and more stable.
- It consists of normalizing activation vectors from hidden layers using mean and variance of the current batch.
- In practice, we typically normalize the value of activation potential rather than output activations, although sometimes debated whether we should normalize before or after activation.

Accelerating Training with Batch Normalization

- Given a vector of linear combinations from the previous layer $z^{[l]}$ for each observation in a dataset, we can calculate the mean and variance as:

$$\mu = \frac{1}{m}\sum_i z_i^{[l]} \qquad \sigma^2 = \frac{1}{m}\sum_i (z_i^{[l]} - \mu)^2$$

- Using these values of mean and SD, we can normalize the vectors $z^{[l]}$ as follows.

Accelerating Training with Batch Normalization

- When we use Batch Norm, after calculating the z-value, we apply batch norm and then the activation function on that.
- Batch normalization is one of the reasons why deep learning has made such outstanding progress in recent years.
- Batch normalization enables the use of higher learning rates, greatly accelerating the learning process.
- It also enabled the training of deep neural networks with sigmoid activations that were previously deemed too difficult to train due to the vanishing gradient problem.

Visualization in CNN

- Deep learning neural networks are generally opaque, meaning that although they can make useful and skillful predictions, it is not clear how or why a given prediction was made.
- CNNs have internal structures that are designed to operate upon two-dimensional image data, and as such preserve the spatial relationships for what was learned by the model.
- Specifically, the two-dimensional filters learned by the model can be inspected and visualized to discover the types of features that the model will detect, and the activation maps output by convolutional layers can be inspected to understand exactly what features were detected for a given input image.

Visualization in CNN

- We can visualize CNN model architecture by calling summary() method: *model.summary()*.
- CNN uses learned filters to convolve the feature maps from the previous layer. Filters are two- dimensional weights and these weights have a spatial relationship with each other. The steps to be followed to visualize the filters are.
  - Iterate through all the layers of the model using model.layers
  - If the layer is a convolutional layer, then extract the weights and bias values using *get_weights()* for that layer.
  - Normalize the weights for the filters between 0 and 1
  - Plot the filters for each of the convolutional layers and all the channels.

Visualization in CNN

- Feature maps are generated by applying Filters or Feature detectors to the input image or the feature map output of the prior layers.
- Feature map visualization will provide insight into the internal representations for specific input for each of the Convolutional layers in the model. The steps to visualize the feature maps are listed below.

Visualization in CNN

- Define a new model, that will take an image as the input.
- Load the input image for which we want to view the Feature map to understand which features were prominent to classify the image.
- Convert the image to NumPy array
- Normalize the array by rescaling it
- Run the input image through the visualization model to obtain all intermediate representations for the input image.
- Create the plot for all of the convolutional layers and the max pool layers but not for the fully connected layer.

CNN for Style Transfer

- Style transfer is the technique of recomposing images in the style of other images.
- One of these algorithms is called neural style. The goal of neural style is to be able to take an arbitrary photograph and render it as if it were painted in the style of a famous artist.
- Neural style transfer is an optimization technique used to take two images—a content image and a style reference image (such as an artwork by a famous painter)—and blend them together so the output image looks like the content image, but "painted" in the style of the style reference image.

CNN for Style Transfer

- This is implemented by optimizing the output image to match the content statistics of the content image and the style statistics of the style reference image. These statistics are extracted from the images using a convolutional network.
- The base idea on which Neural Style Transfer is proposed is “it is possible to separate the style representation and content representations in a CNN, learned during a computer vision task.
- Training a style transfer model requires two networks: a pre-trained feature extractor and a transfer network.