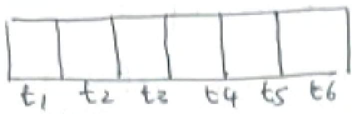
Module 2 — Disha D.N

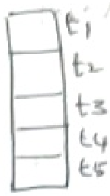
The convolutional Neural Networks:

Definition: Convolutional neural network known as convnet, or CNNs, are special kind of neural network for processing data that has a known grid-like topology like time-series data (1D) or images (2D)

Eg: 1D:



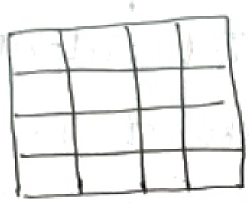
Time series data



1D

2D:

Images



H×H pixels.

→ CNNs are commonly used in image processing and computer vision tasks. They are inspired by the structure and functioning of the visual cortex in the human brain which is responsible for processing visual information.

→ In CNNs, the input data is usually an image, represented as a matrix of pixel values. The network consists of several layers of interconnected neurons, including Convolutional layers, pooling layers and fully connected layers.

→ The convolutional layers are the core of CNN, and they consist of a set of filters that slide over the input image, performing a mathematical operation called convolution. This operation extracts features from the input image such as edges lines and textures.

→ Pooling layers reduce the spatial size of the feature maps produced by the convolutional layers, by summarizi-ng groups of neurons into a single output. This helps to make the network more efficient and less sensitive to small variations in the input image.

→ Fully connected layers are similar to those in a traditional ANN. The output of the last fully connected layer is passed through an activation function to obtain a probability distributions over the possible classes

→ Then CNNs are trained using backpropagation, where the error between the predicted output and actual output is propagated backward through the network to adjust the weights of the neurons. This process is repeated multiple epochs until the network achieves a satisfactory level of accuracy on the training data.

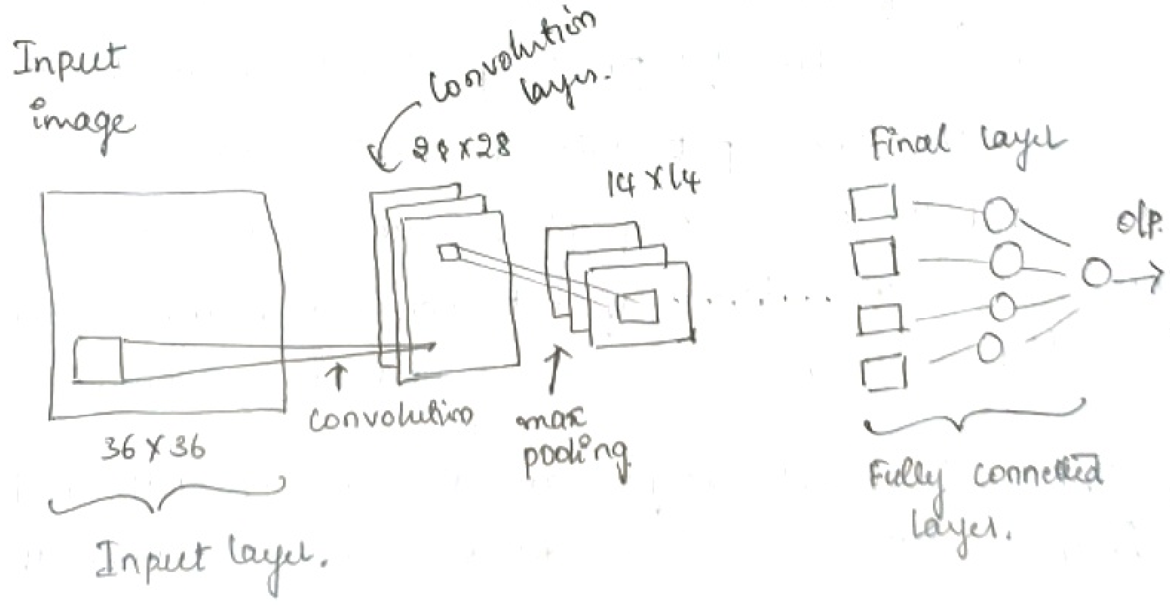


Fig: CNN sample architectures.

Disha D.N

→ Back propagation and Forward propagation in CNN

Forward propagation: It is the process of moving data through the network to produce an output.

Algorithm: forward propagation in CNN

step1: Initialization

→ Initialize the weights and biases of the convolutional layers and fully connected layers randomly or using a pre-trained model

step2: Convolutional layer

→ Perform convolution operation between the input image and the filters in the convolutional layer.

→ Apply activation function (like ReLU) on the output feature maps to introduce non-linearity

→ Add bias term to the output of activation function.

step3: Pooling layer

→ Perform max pooling or average pooling operation on the output feature maps of the convolutional layer.

→ This helps to reduce the spatial size of the feature maps, while retaining the most important features

step4: Fully connected layer

→ Flatten the output feature maps of the pooling layer into a 1D vector.

→ Pass the flattened vector through one or more fully connected layers, where each neuron is connect to all neurons in the previous layer.

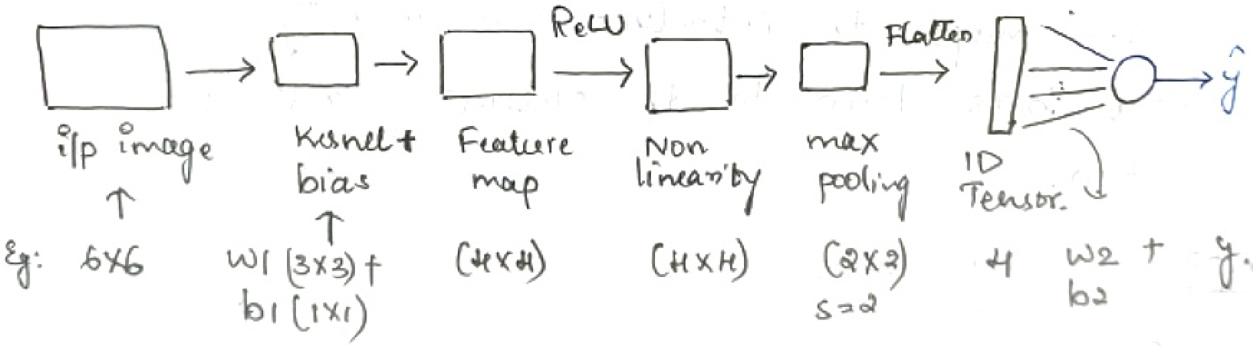
→ Apply activation function on the output of each neuron.

<u>step5:</u> Output layer

→ Use the activation function to convert the last fully connected layer into probability distribution over the possible classes.

→ The class with highest probability is the predicted class.

Architecture in General.



Logical Flow

$$X \to \underset{\substack{\uparrow \\ W_1 b_1}}{\circledast} \to Z_1 \xrightarrow{ReLU} A_1 \xrightarrow[Pool]{max} P_1 \xrightarrow{flatten} F \to Z_2 \xrightarrow{Activation} A_2 \to L$$

$Z_1 = conv(x, W_1) + b_1$ $\qquad$ $F = Flatten(P_1)$

$A_1 = ReLU(z_1)$ $\qquad$ $Z_2 = FW_2 + b_2$

$P_1 = maxpool(A_1)$ $\qquad$ $A_2 - Activation(z_2)$

Backpropagation:

It is the process of the computing the gradient of the loss function with respect to the parameters of the network.

Algorithm : Backpropagation

Step 1: Compute the loss

→ Compute the loss between the predicted class probabilit-es and the class labels using a loss function (like cross-entropy loss)

Step 2: Output layer

→ Compute the gradient of the loss with respect to the output of the ~~softmax~~ activation function in the output layer.

Step 3: Fully connected layer

→ Compute the gradients of the loss with respect to the weights and biases of the fully connected layers using the chain rule of calculus.

→ update the weights and biases of the fully connected layers using gradient descent or a similar optimization algorithm.

Step 4: Pooling layer

→ Propagate the gradients of the loss back through the max pooling or average pooling operation in the pooling layer.

Step 5: Convolutional layer

→ Propagate the gradients of the loss back through the convolutional layer using the backpropagation algorithm

→ Compute the gradients of the loss with respect to the weights and biases of the convolutional layers using the chain rule of calculus.

→ Update the weights and biases of the convolutional layers using gradient descent or a similar optimization algorithm.

Step 6: Repeat.

Back propagation process to find gradient descent on all trainable parameters

Eg: From logical connection of forward propagation we have

$$Z_1 = \text{Conv}(x, w_1) + b_1$$
$$A_1 = \text{ReLU}(Z_1)$$
$$P_1 = \text{maxpool}(A_1)$$
$$F = \text{Flatten}(P_1)$$
$$Z_2 = W_2 F + b_2$$
$$A_2 = \sigma(Z_2)$$

Here we need to obtain gradient descent on

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$$
$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$
$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}$$
$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$X \to \circledast \to Z_1 \xrightarrow{\text{relu}} A_1 \xrightarrow[\text{pool}]{\text{max}} P_1 \xrightarrow{\text{flatten}} F \xrightarrow{w_2 b_2} Z_2 \xrightarrow{\text{sigmoid}} A_2(\hat{y}) \to L$$

$\uparrow$ $w_1 b_1$

$\longleftarrow$ Backpropagation

1st we need to update w_2 and b_2

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$\boxed{\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial w_2}}$$

$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$\boxed{\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial b_2}}$$

Then we need to update w_1 and b_1

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$$

$$\boxed{\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial w_1}}$$

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1}$$

$$\boxed{\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial Z_2} \times \frac{\partial Z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial w_1}}$$

Data augmentation

→ It is a technique used in machine learning, and specifically in convolutional neural networks (CNNs) to increase the amount and diversity of training data by creating new, modified versions of the original data

Advantages of dataaugmentation are as follows.

1. Increase the size of the dataset: Data augmentation is used to artificially increase the size of the dataset by generating new examples from the existing ones.

2. Create more diversity in the dataset: By applying various transformations to the original images, such as rotation, flipping and scaling.

3. Reduce overfitting: Overfitting occurs when CNN becomes too specialized to the training data and is unable to generalize well to new data. By creating more diverse training data, data augment-ation can help prevent CNN from overfitting.

4. Improve accuracy: More examples, model learns better.

Why we cannot use ANN on images?

→ ANN can be used for image classification, but the major problem we get are

* High computational cost
* Overfitting
* loss of spatial arrangement of pixels.

Eg:

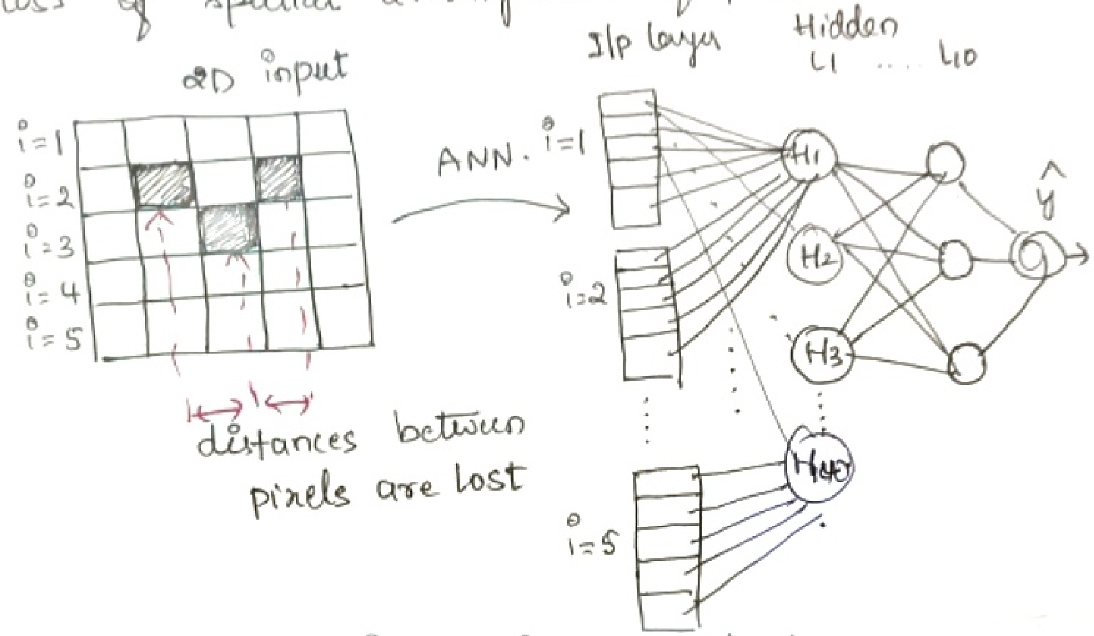
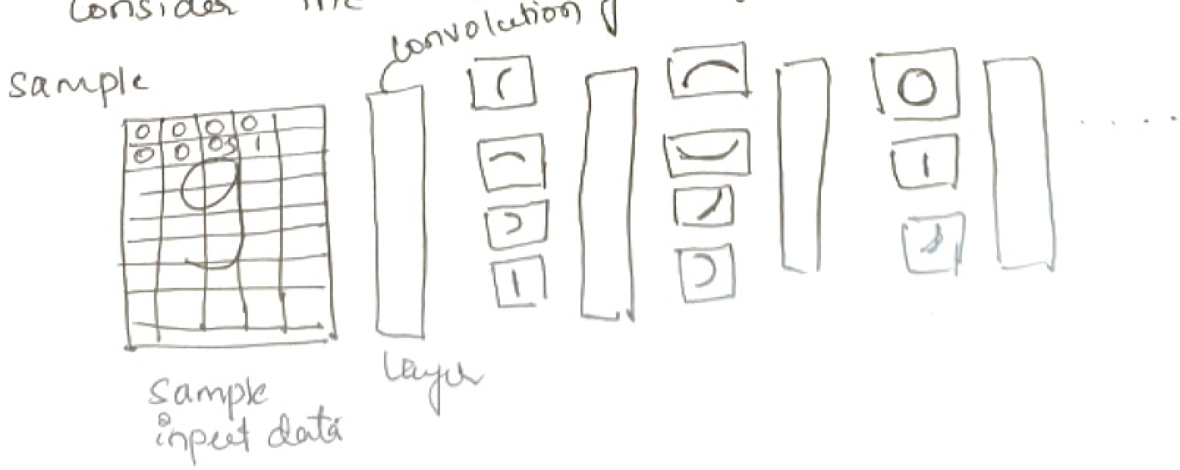


Fig: Converting 2D image pixels to 1D ANN network.

Hence it is not a good idea to use ANN for 2D classification.

CNN working

Consider the following digit identification



→ As seen in figure, convolution layer is present in CNN.

→ Convolution layers are like filters, it will extract features from the sample. Then the features are compared with input data sample.

→ If features are present, then only those will be activated and passed to the 2nd convolution layer where it will merge the previously activated features to form new features.

→ This process is continued until we identify the given samples.

Convolution operation.

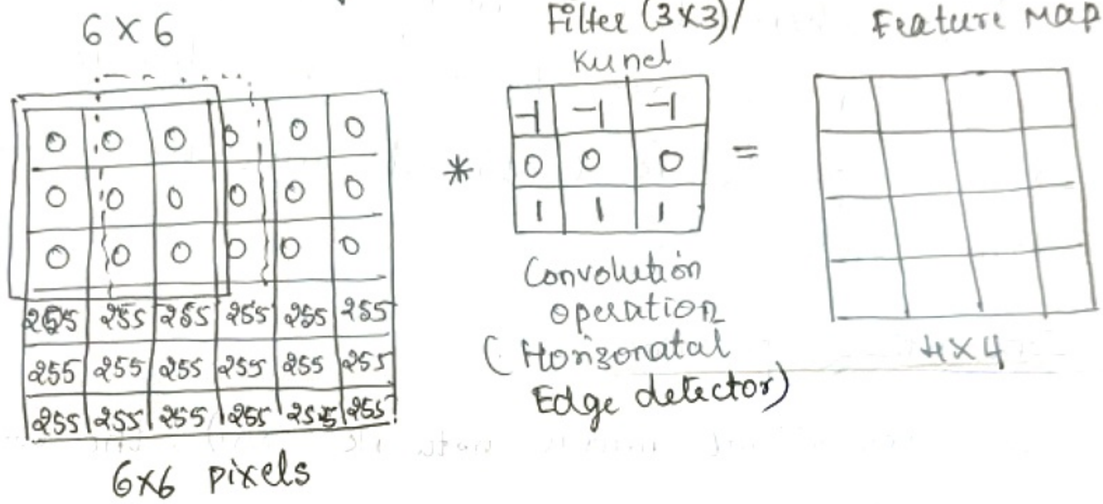
→ In convolutional neural network (CNN), the convolution operation is a mathematical operation that is applied to the input image classification, object detection or segmentation.

→ The convolution operation involves a set of learnable filters, also known as kernels, which slide over the input image and perform a dot product operation between the filter weight and pixel values in the input image. The result of this dot product is a single value, which represents the activation of a particular feature in the input image.

→ The filters in a convolutional layer are typically small in size such as 3X3 or 5X5 and are applied at every location in the input image.

→ The output of a convolution is a feature map that contains the activations of all features across the entire input image.

Eg: Consider a gray scale image,



→ As it is seen in figure, there is an image with 6X6 pixels, with 0 indicates it is black and 255 indicates it is white.

→ Filter size of 3X3 is choosen. Here, in this example horizontal filter is choosen which will detect only horizontal edges.

→ The filter is applied step wise to the input image to get a feature map.

Eg: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = (0*-1 + 0*-1 + 0*-1) + (0*0) + (0*0) + (0*0) + (0*1 + 0*1 + 0*1)$

$= 0$ ← Filled to first value of the feature map

Padding and strides.

→ Padding and strides are two important concepts in convolutional neural networks that are used to control the size of the output feature maps produced by convolutional layers.

Padding :

→ It is the process of adding extra rows and columns of zeros to the edges of an input image before applying a convolutional filter to it.

→ Padding is required mainly because of 2 reasons.

① Preserving spatial dimensions : when we apply a convolutional filter to an input image, the size of the output feature map is smaller than the input image due to the loss of pixels around the edges.

→ This can lead to the loss of important spatial information at the edges of the input image.

→ By adding padding to the input image, we can ensure that the output feature map has the same spatial dimensions as the input image and preserve spatial information.

② Avoiding border effects:

→ when a filter is applied to the input image, the pixels at the edges of the image are used less frequently than the pixels in the center of the image.

→ This can result in border effects where the output feature map has reduced quality or is distorted at the edges. By adding padding to the input image, we can

Then filter is convolved through each column of the input image.

→ The final feature map will be of the format

0	0	0	0
255	255	255	255
255	255	255	255
0	0	0	0

4 × 4

Note: 1. If we change the value of filter we get different edge detectors

2. Filter values are initially choosen randomly, during backpropagation values are updated.

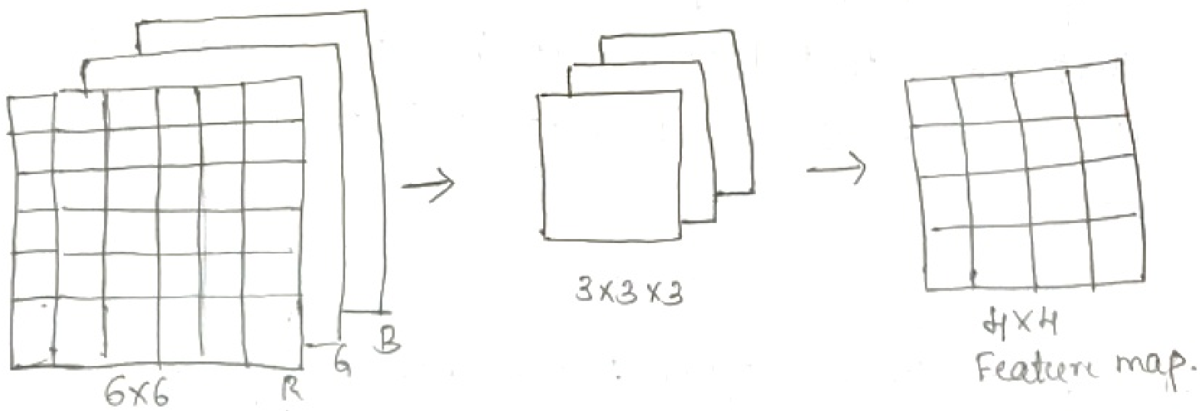
3. Size of the feature map is calculated by the formula $K = n - m + 1$

where K = size of feature map

n = size of input image

m = size of filter

Convolution with RGB.

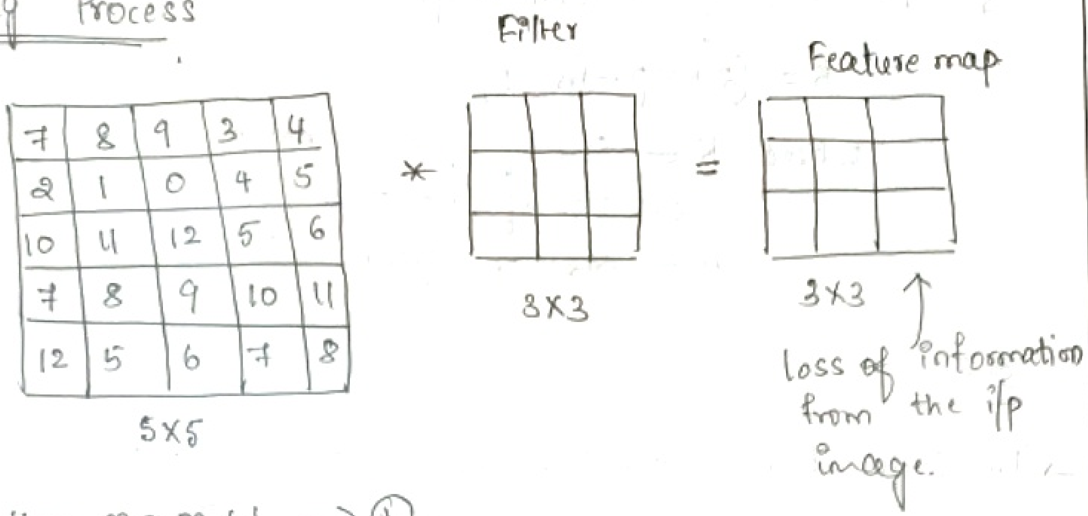


Note: If you apply multiple features on same image (Eg: 3×3 horizontal, 3×3 vertical filter) is applied then we get $4 \times 4 \times \underset{\uparrow}{2}$

No. of filters applied.

ensure that the pixels in the input image are used equally.

Padding Process



$K = n - m + 1 \rightarrow (1)$

$K = 5 - 3 + 1$

$K = \underline{\underline{3}}$

we need to get the feature map size is same as input size

i,e $n = K = 5$ (in this example)

Apply it in formula (1)

$5 = n - m + 1$

$5 + m - 1 = n$

$5 + 3 - 1 = n$

$\Rightarrow n = \underline{\underline{7}}$

→ Hence we need to convert 5X5 image size into 7X7 by adding extra rows and columns called padding

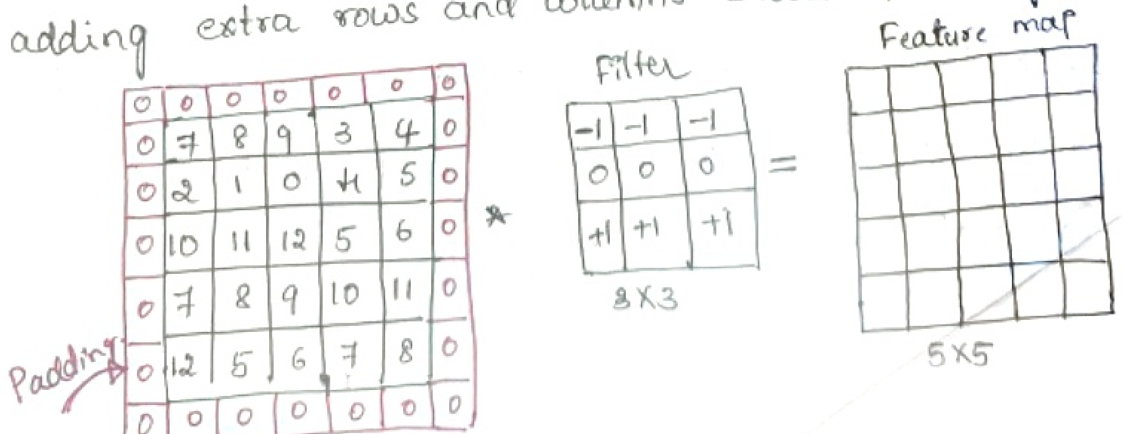


Fig: Padding process

→ Usually the padding values will be zeros, Hence it is called as zero padding.

→ The size of feature map formula after padding is

$$K = (2 + np - m + 1)$$

where $K \leftarrow$ size of feature map
$n \leftarrow$ size of input image
$m \leftarrow$ size of filter

Strides

→ In convolutional neural network, strides refer to the number of steps the filter takes as it moves across the input data to perform convolution

→ If the strides is set to 1, the filter moves one pixel at a time, and output will have the size

$$K = n - m + 1.$$

→ If the strides is set to 2, the filter moves two pixels at a time, resulting in a smaller feature map.

→ Using larger strides values can reduce the output size and computational complexity of the network, but may also result in a loss of information, as the filter skips over some input values.

→ On the other hand, smaller stride values can lead to more accurate feature extraction at the cost of increased computational complexity.

Disadvantages of using strides.

1. Loss of information: Using larger strides may result in loss of information, as the filter skips over some input values.

2. Reduced resolution: Using larger strides can can result in a reduced resolution of the output feature maps, which can affect the ability of the network to detect fine details in the input data.

3. Difficulty in choosing the optimal value: Choosing the optimal stride value can be a challenge.

Pooling layer in CNN.

→ It is a technique used to down sample the feature maps generated by convolutional layers. The pooling operation involves partitioning the feature map into non-overlapping rectangular regions and replacing each region with a single value that summarizes the information in the region.

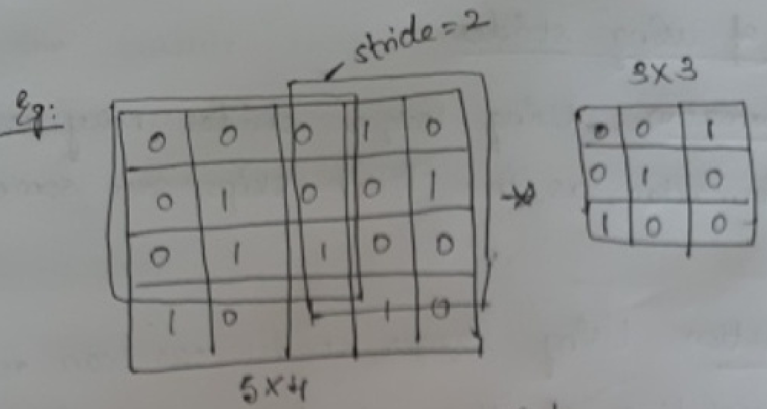
→ The most commonly used operation is max pooling, in which the maximum value within each region is selected as the output value.

→ other types of pooling operations include average pooling, L2-norm pooling etc.

→ Pooling has several benifits

① Reduced complexity: By reducing the spatial resolution

Eg:



* with increase in stride value size of feature map will be reduced.

The size of feature map (K)

$$K = n - m + 1 \quad \text{(Note: stride = 1)}$$

with stride = 2

$$K = \left[\frac{n-m}{2} + 1\right]$$

In general,

$$K = \left[\frac{n-m}{s} + 1\right]$$

with padding,

$$K = \left[\frac{n+2p-f}{2} + 1\right]$$

<u>Advantages</u>

1. <u>Reduced computation</u>: Using large stride values can reduce the computation required for processing input data, making the neural network faster and more efficient.
2. <u>Increased receptive field</u>: large strides can allow to increa-se the receptive field of the filter, allowing it to capture more information from the input data.
3. <u>smaller output size</u>: using larger stride values can reduce the output size of the feature maps, which can help to reduce overfitting and improve the generalization performance of the network.

of the feature maps, pooling reduces the number of parameters and computations required in the subsequent layers of the neural network.

② <u>Translation invariance</u>: Pooling can help to make model more robust to small translations of the input data, as the output values are based on local features rather than specific pixel locations.

<u>Disadvantages</u>

1. <u>loss of information</u>: Pooling discards some of the information in the feature maps which can lead to a loss of fine-grained details

2. <u>Reduced Resolution</u>: Pooling reduces the spatial resolution of the feature maps which can make it more difficult for the model to detect small or detailed features in the input data.

Eg:

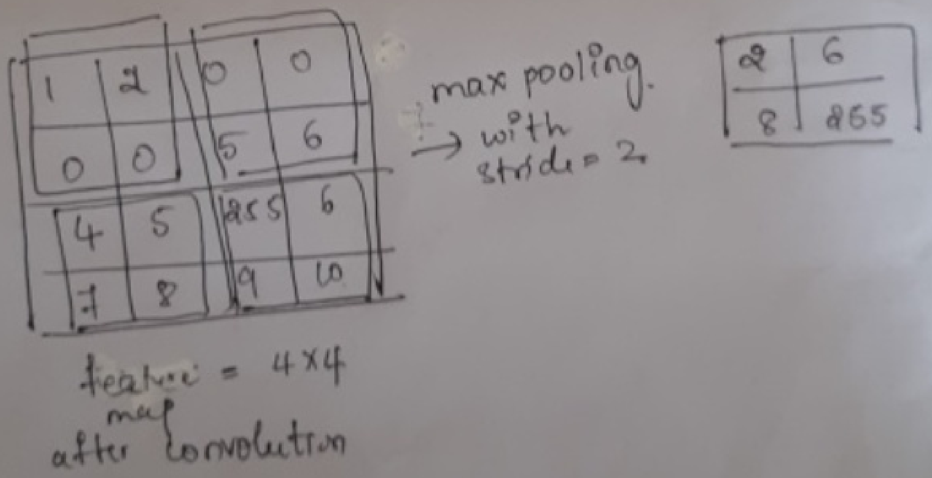


Fig: Max pooling, after convolution operation