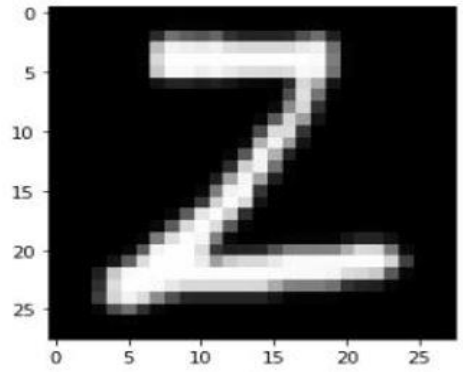
Convolutional Neural Network: A Quick Overview

In the world of AI, Machine Learning, Deep Learning, and Computer Vision, we have come across and heard about various tasks like Image Classification, Object Detection, Image Pattern Detection, Text Classification, and Face Recognition. In this article, I have written a quick overview of Convolutional Neural Networks.

Convolutional Neural Network (CNN or ConvNets) is a Deep Learning technique which is generally used to perform the tasks mentioned above. Here, the input is an image (simply a matrix of pixels) which is fed into a CNN model that assigns some learnable weights and biases to various aspects of an image to analyze input images for recognition and classification. Just like teachers help toddlers recognize different letters and digits through images and diagrams, similarly, computers are also trained on large datasets (millions and thousands of images) for image or text recognition tasks.



A (28 x 28 x 1) pixel grayscale image from the EMNIST dataset for character ***Z***. This is fed as input to a CNN model which then predicts its label based on the learning and features.

The elementary role of a CNN model is to modify the images into a form that is computationally less expensive with no loss of features, which are important for getting good and accurate predictions.

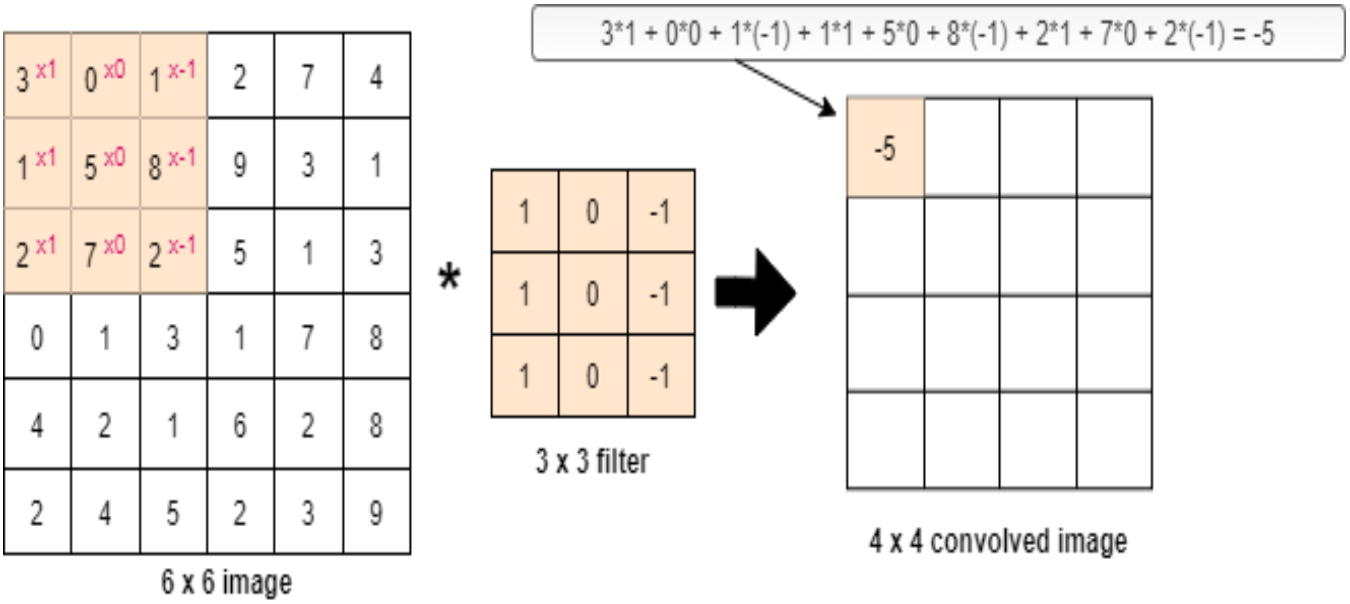
Types of Layers in a CNN model:

1. Convolutional Layer
2. Pooling Layer
3. Fully Connected (Dense Layer)

1. Convolutional Layer: In the Convolutional Layer, the convolution operation is performed to get the position and strength of features of the input image. It takes an input image (say 12 x 12 image) and convolves (dot product) with a filter (say 3 x 3 filter/kernel) to give a convolved image of size 10 x 10 (if p=0 i.e. valid padding and s = 1).

([n + 2p – f]/s+ 1) x ([n + 2p – f]/s+ 1) is the dimension of convolved image after convolution operation.

Note: n = Input image dimensions
p = padding
f = size of filter
s = stride



A convolution operation

Padding: Padding is a hyperparameter of a Convolutional Layer. It is a process of adding zeros to our input images. If we use the convolutional layer without setting the padding parameter, then the output image shrinks and there are chances of information/features from edges getting lost.

Types of Padding

Valid Padding	**Same Padding**
No Padding. The output image shrinks depending on input & filter size.	Padding so the output image has the same size as the input image.
p = 0	**p = (f – 1) / 2**
(n x n) * (f x f) => (n – f + 1) x (n – f + 1)	(n + 2p) x (n + 2p) * (f x f) => (n x n)

Types of Padding

Valid Padding

No Padding. The output image shrinks depending on input & filter size.

p = 0

(n x n) * (f x f) => (n – f + 1) x (n – f + 1)

Same Padding

Padding so the output image has the same size as the input image.

p = (f – 1) / 2

(n + 2p) x (n + 2p) * (f x f) => (n x n)

Note: * is a convolution operation

A Quick Note: Filter/kernel is a matrix of size (f x f) and f – a hyperparameter, is usually odd like 3 x 3, 5 x 5, 7 x 7, etc.

No. of channels of Filter = No of channels of image.

Strides: Strides is another hyperparameter in the convolution layer which tells about the number of pixels to shift the filter over the input matrix. If we set s = 3, shifting filter over input image by 3 pixels.

2. Pooling Layer: Usually used after a Convolutional Layer. The pooling layer is responsible for reducing the size of the Convolved Features. In this layer, there are no parameters and weights to learn for the model. It aims to reduce the size of the representation to speed up the computation and to make some features more robust i.e. extracting dominant features from convolved images.

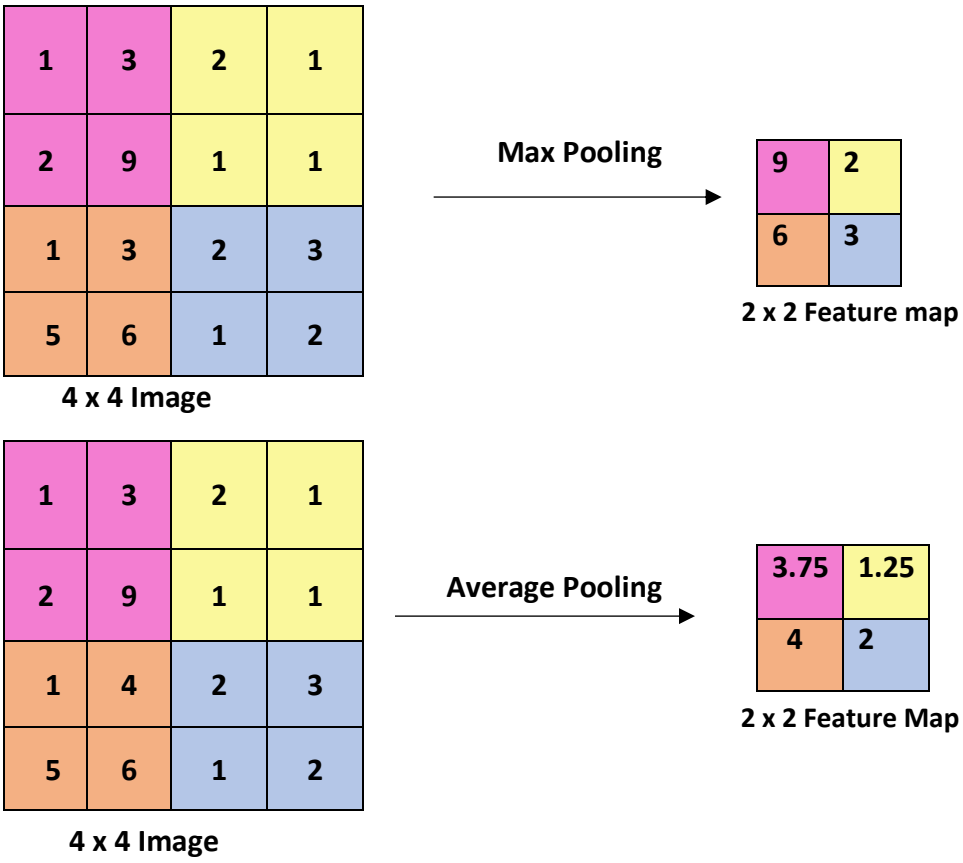
Types of Pooling

Max Pooling

Calculates the maximum value from the part of image covered by filter

Average Pooling

Calculates the average value from the part of image covered by filter



A Quick Note: Feature Maps in CNN captures features which is the output of applying filters to the input image.

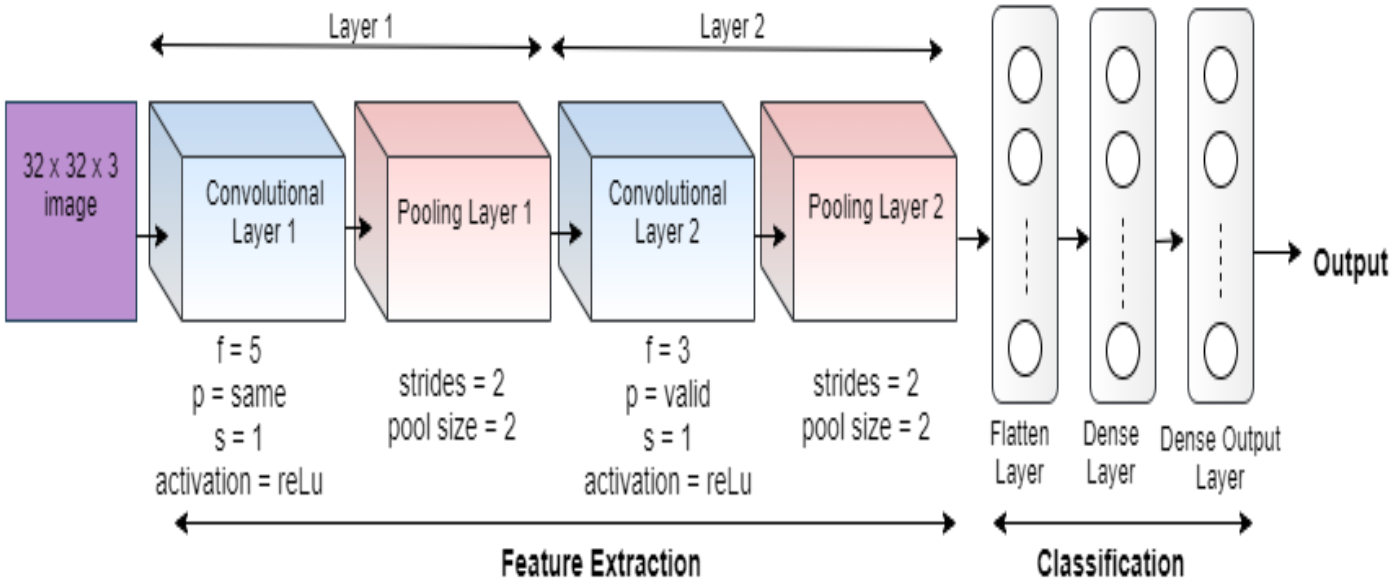
3. Fully Connected (Dense): A Fully Connected Layer (Dense Layer) takes the output of the previous layers (usually Pooling Layers) and flattens it to a single vector. The 1st fully connected layer takes in the features and assigns weights to them, and the output Dense Layer gives the probabilities for each class using an activation function.

A general strategy for creating a CNN Model:-

1. Input an image into the convolutional layer. Select appropriate parameters (filters, strides, and padding). Perform convolution on the image and use an activation function depending on the task to be performed.
2. Add a Pooling Layer and perform pooling to reduce dimensionality size. Max or Avg Pooling can be used based on functionality.
3. Add convolutional layers to the model as per accuracy/loss i.e. the metric chosen and depth of Neural Network required based on image complexity.

4. Flatten the output (Flatten Layer) and feed into a fully connected dense layer (FC Layer).
5. Output the classes using an activation function using Dense Layer. Additionally, we can use Batch Normalization and Dropout Layers to improve the performance of our model.

Let's take an example to understand the dimensions of images at each layer of a CNN model. Let's take a simple CNN model as shown below:



Basic CNN Architecture

The input is (32 x 32 x 3) dimensional image which is fed into a Convolutional Layer.

In **1st Convolutional Layer**, f = 5 -> a filter matrix of size (5 x 5) that's convolved with (32 x 32 x 3) image with *same* padding and s = 1 to get a (32 x 32 x n) dimensional image. Let's assume there are n filters of size (5 x 5). Hence, dimension of image is (32, 32, n).

Calculation: p = (f-1)/2 = (5-1)/2 = 2

([n + 2p – f]/s + 1) = ([32 + 2(2) – 5]/1 + 1) = 32

Using p = *same*, after calculations we are getting an output image of the same size as the input image.

In **Pooling Layer 1**, we get a matrix with features having either maximum or average value. For (32 x 32 x n) convolved image with a stride of s = 2 and pool size = 2, we get a (16 x 16 x n) image.

In **2nd Convolutional Layer**, f = 3 -> a filter matrix of size (3 x 3) that's convolved with (16 x 16 x n) image with *valid* padding to get a (14 x 14) dimensional image. Here, s = 1, and the filter moves by 1 pixel. Let's assume there are n' filters of size (3 x 3). Hence dimension of the convolved image is (14, 14, n').

Calculation: p = 0

$([n + 2p – f]/s + 1) = ([16 + 2(0) – 3]/1 + 1) = 14$

In **Pooling Layer 2**, either a maximum pool or an average pool can be used. For (14 x 14 x n') convolved image we get a (7 x 7 x n') image since we have used a stride of s = 2 and pool size = 2.

This image is then fed to the Flatten Layer to convert the 3D image into a 1D vector of size 7 * 7 * n` = 49*n`. This is then fed into Dense Layer having x units/neurons with some normalization techniques/layers to improve the model's performance. Another Output Dense Layer with y classes (depending on the problem being considered) outputs labels or probabilities.

How do we calculate no. of parameters for a Convolutional Layer and Fully Connected Layer?

 # *Parameters in Conv. Layer* *:* [(f x f x no. of channels in previous layer)+ 1] x no. of filters

 # *Parameters in FC Layer* *:* [no. of units in current layer x no of units in previous layer] + no. of units in current layer

Note: There are no parameters for Pooling Layer because it's used to reduce the dimensionality of the image and has no learning of parameters while training the model.

Let's take an example to calculate no. of parameters for a Convolutional Layer and FC Layer:

No. of parameters in Convolutional Layer 1 of '***Basic CNN Architecture'*** *figure :-*

Here, f = 5, no. of filters = n = 64(say) and no. of channels in previous layer = 3

Therefore, [(5 x 5 x 3) +1] x 64 = *4864 no. of parameters*

No. of parameters in Fully Connected Layer (Dense Layer) of '***Basic CNN Architecture'*** *figure :-*

Here, no. of units in current layer = 120 (say) and no. of units in previous layer = 400 (say)

Therefore, [120 x 400] + 64 = *48064 no. of parameters*