Convolutional Neural Network

Tushar B. Kute,
http://tusharkute.com

Convolutional Neural Network

- Convolutional Neural Networks are a powerful artificial neural network technique.
- These networks preserve the spatial structure of the problem and were developed for object recognition tasks such as handwritten digit recognition.
- They are popular because people are achieving state-of-the-art results on difficult computer vision and natural language processing tasks.

Convolutional Neural Network

- Given a dataset of gray scale images with the standardized size of 32 × 32 pixels each, a traditional feedforward neural network would require 1,024 input weights (plus one bias).
- This is fair enough, but the flattening of the image matrix of pixels to a long vector of pixel values looses all of the spatial structure in the image.
- Unless all of the images are perfectly resized, the neural network will have great difficulty with the problem.

Convolutional Neural Network

- Convolutional Neural Networks expect and preserve the spatial relationship between pixels by learning internal feature representations using small squares of input data.
- Feature are learned and used across the whole image, allowing for the objects in the images to be shifted or translated in the scene and still detectable by the network.
- It is this reason why the network is so useful for object recognition in photographs, picking out digits, faces, objects and so on with varying orientation.

Convolutional Neural Network

- In summary, below are some of the benefits of using convolutional neural networks:
  - They use fewer parameters (weights) to learn than a fully connected network.
  - They are designed to be invariant to object position and distortion in the scene.
  - They automatically learn and generalize features from the input domain.

Convolutional Neural Network

- There are three types of layers in a Convolutional Neural Network:

  1. Convolutional Layers.
  2. Pooling Layers.
  3. Fully-Connected Layers.

Convolutional Neural Network

- Convolutional layers are comprised of filters and feature maps.
- Filters
  - The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron.
  - The input size is a fixed square called a patch or a receptive field. If the convolutional layer is an input layer, then the input patch will be pixel values.
  - If they deeper in the network architecture, then the convolutional layer will take input from a feature map from the previous layer.

Convolutional Neural Network

- Feature Maps
  - The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time.
  - Each position results in an activation of the neuron and the output is collected in the feature map.
  - You can see that if the receptive field is moved one pixel from activation to activation, then the field will overlap with the previous activation by (field width - 1) input values.

Convolutional Neural Network

- Feature Maps
  - The distance that filter is moved across the input from the previous layer each activation is referred to as the stride.
  - If the size of the previous layer is not cleanly divisible by the size of the filters receptive field and the size of the stride then it is possible for the receptive field to attempt to read off the edge of the input feature map.
  - In this case, techniques like zero padding can be used to invent mock inputs with zero values for the receptive field to read.

Convolutional Neural Network

- Pooling Layers
  - The pooling layers down-sample the previous layers feature map.
  - Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map.
  - As such, pooling may be consider a technique to compress or generalize feature representations and generally reduce the overfitting of the training data by the model.

Convolutional Neural Network

- Pooling Layers
  - They too have a receptive field, often much smaller than the convolutional layer.
  - Also, the stride or number of inputs that the receptive field is moved for each activation is often equal to the size of the receptive field to avoid any overlap.
  - Pooling layers are often very simple, taking the average or the maximum of the input value in order to create its own feature map.

Convolutional Neural Network

- Fully Connected Layers
  - Fully connected layers are the normal flat feedforward neural network layer.
  - These layers may have a nonlinear activation function or a softmax activation in order to output probabilities of class predictions.
  - Fully connected layers are used at the end of the network after feature extraction and consolidation has been performed by the convolutional and pooling layers.
  - They are used to create final nonlinear combinations of features and for making predictions by the network.

Convolutional Neural Network

- Worked Example
  - You now know about convolutional, pooling and fully connected layers.
  - Let’s make this more concrete by working through how these three layers may be connected together.

Convolutional Neural Network

- Worked Example
- Image Input Data
  - Let’s assume we have a dataset of gray scale images. Each image has the same size of 32 pixels wide and 32 pixels high, and pixel values are between 0 and 255, e.g. a matrix of 32 × 32 × 1 or 1,024 pixel values.
  - Image input data is expressed as a 3-dimensional matrix of width × height × channels.
  - If we were using color images in our example, we would have 3 channels for the red, green and blue pixel values, e.g. 32 × 32 × 3.

Convolutional Neural Network

- Worked Example
- Convolutional Layer
  - We define a convolutional layer with 10 filters and a receptive field 5 pixels wide and 5 pixels high and a stride length of 1.
  - Because each filter can only get input from (i.e. see) 5 × 5 (25) pixels at a time, we can calculate that each will require 25 + 1 input weights (plus 1 for the bias input).
  - Dragging the 5 × 5 receptive field across the input image data with a stride width of 1 will result in a feature map of 28 × 28 output values or 784 distinct activations per image.

Convolutional Neural Network

- Worked Example
- We have 10 filters, so that is 10 different 28 × 28 feature maps or 7,840 outputs that will be created for one image.
- Finally, we know we have 26 inputs per filter, 10 filters and 28 × 28 output values to calculate per filter, therefore we have a total of 26 × 10 × 28 × 28 or 203,840 connections in our convolutional layer, we want to phrase it using traditional neural network nomenclature.
- Convolutional layers also make use of a nonlinear transfer function as part of activation and the rectifier activation function is the popular default to use.

Convolutional Neural Network

- Pool Layer
  - We define a pooling layer with a receptive field with a width of 2 inputs and a height of 2 inputs.
  - We also use a stride of 2 to ensure that there is no overlap. This results in feature maps that are one half the size of the input feature maps.
  - From 10 different 28 × 28 feature maps as input to 10 different 14 × 14 feature maps as output.
  - We will use a max() operation for each receptive field so that the activation is the maximum input value.

Convolutional Neural Network

- Fully Connected Layer
  - Finally, we can flatten out the square feature maps into a traditional flat fully connected layer.
  - We can define the fully connected layer with 200 hidden neurons, each with 10 × 14 × 14 input connections, or 1,960 + 1 weights per neuron.
  - That is a total of 392,200 connections and weights to learn in this layer.
  - We can use a sigmoid or softmax transfer function to output probabilities of class values directly.

CNN : Best Practices

- Now that we know about the building blocks for a convolutional neural network and how the layers hang together, we can review some best practices to consider when applying them.
  - Input Receptive Field Dimensions: The default is 2D for images, but could be 1D such as for words in a sentence or 3D for video that adds a time dimension.
  - Receptive Field Size: The patch should be as small as possible, but large enough to see features in the input data. It is common to use 3 × 3 on small images and 5 × 5 or 7 × 7 and more on larger image sizes.

CNN : Best Practices

- Stride Width: Use the default stride of 1. It is easy to understand and you don’t need padding to handle the receptive field falling off the edge of your images. This could be increased to 2 or larger for larger images.
- Number of Filters: Filters are the feature detectors. Generally fewer filters are used at the input layer and increasingly more filters used at deeper layers.
- Padding: Set to zero and called zero padding when reading non-input data. This is useful when you cannot or do not want to standardize input image sizes or when you want to use receptive field and stride sizes that do not neatly divide up the input image size.

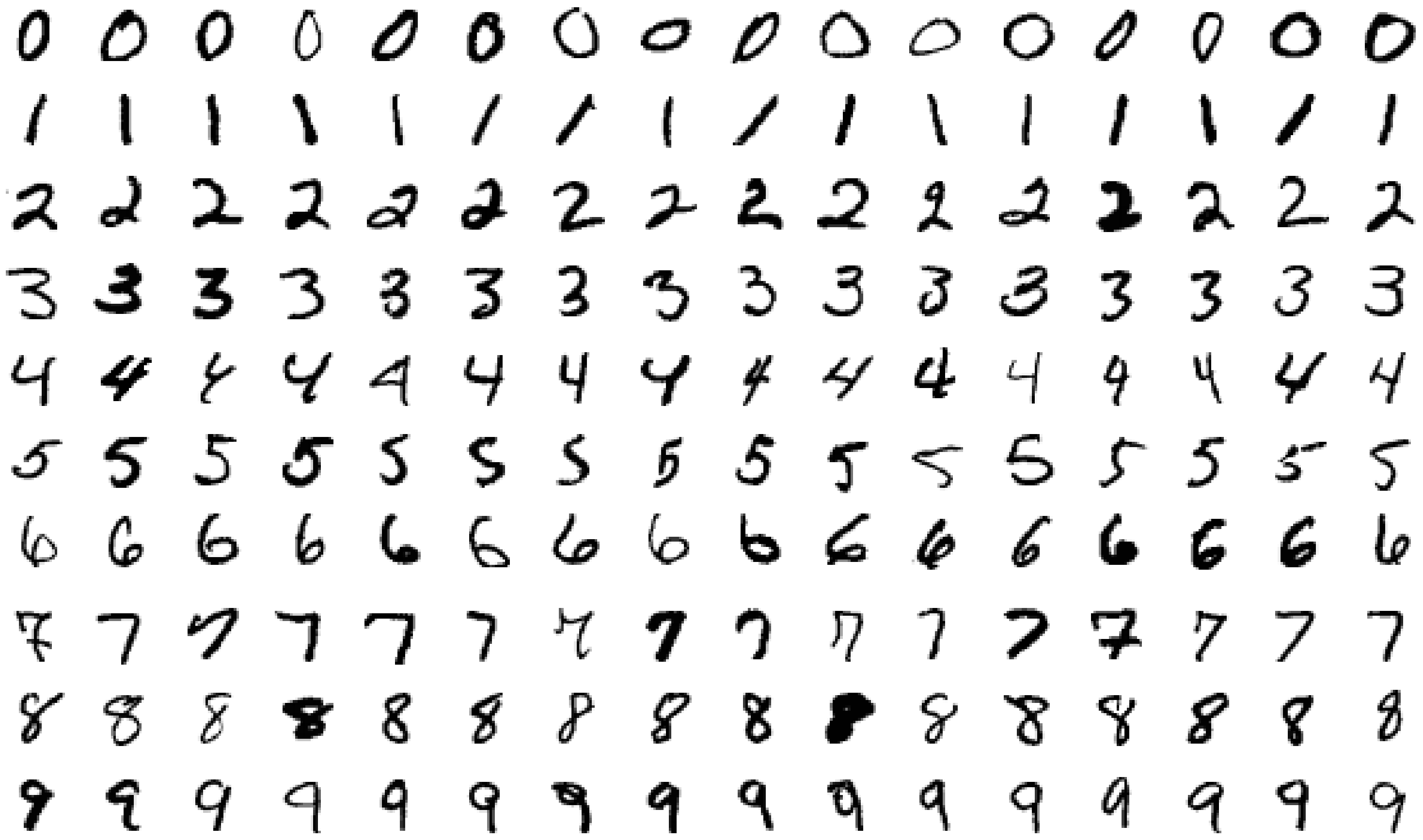
CNN : Best Practices

- Pooling: Pooling is a destructive or generalization process to reduce overfitting. Receptive field size is almost always set to 2 × 2 with a stride of 2 to discard 75% of the activations from the output of the previous layer.
- Data Preparation: Consider standardizing input data, both the dimensions of the images and pixel values.

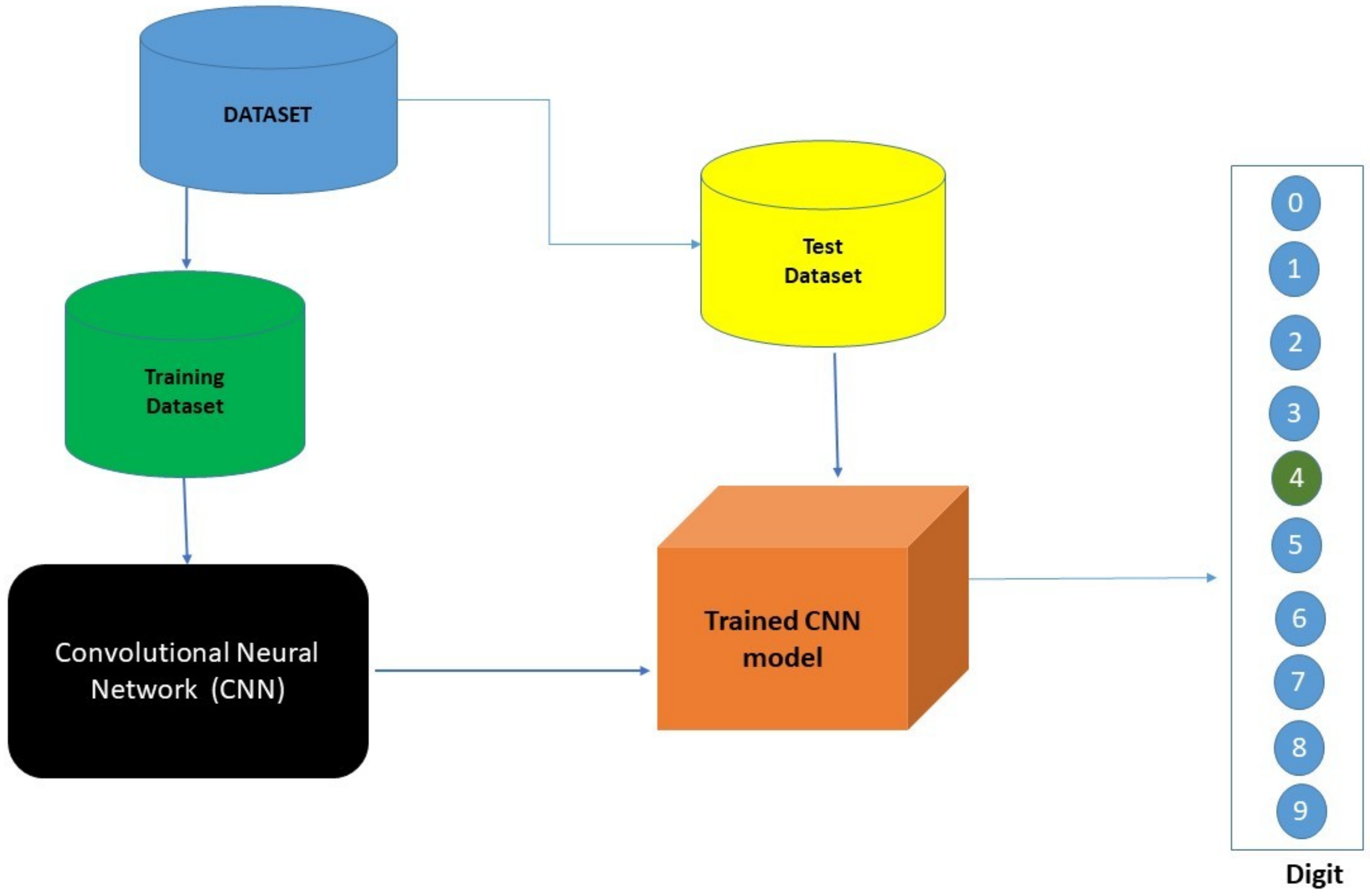
CNN : Best Practices

- Pattern Architecture: It is common to pattern the layers in your network architecture.
- This might be one, two or some number of convolutional layers followed by a pooling layer.
- This structure can then be repeated one or more times. Finally, fully connected layers are often only used at the output end and may be stacked one, two or more deep.
- Dropout: CNNs have a habit of overfitting, even with pooling layers. Dropout should be used such as between fully connected layers and perhaps after pooling layers.

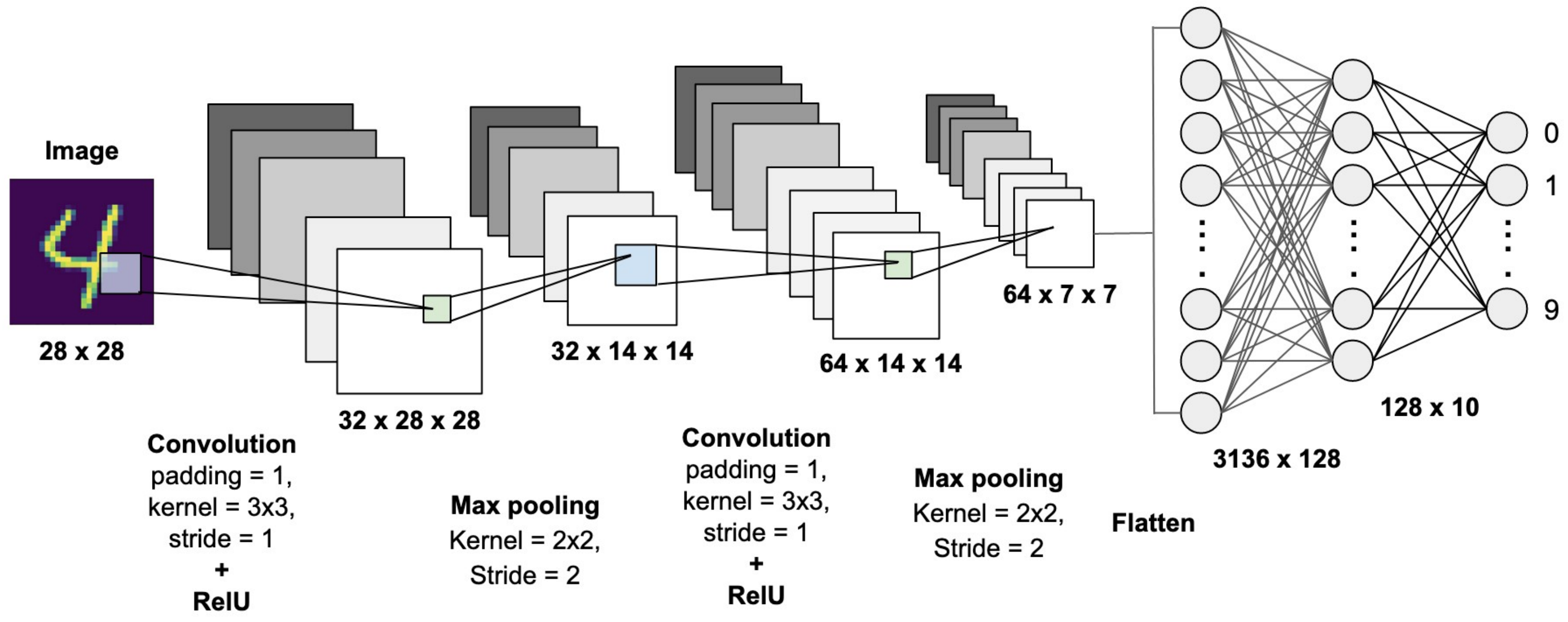
Example: MNIST Handwritten Chars



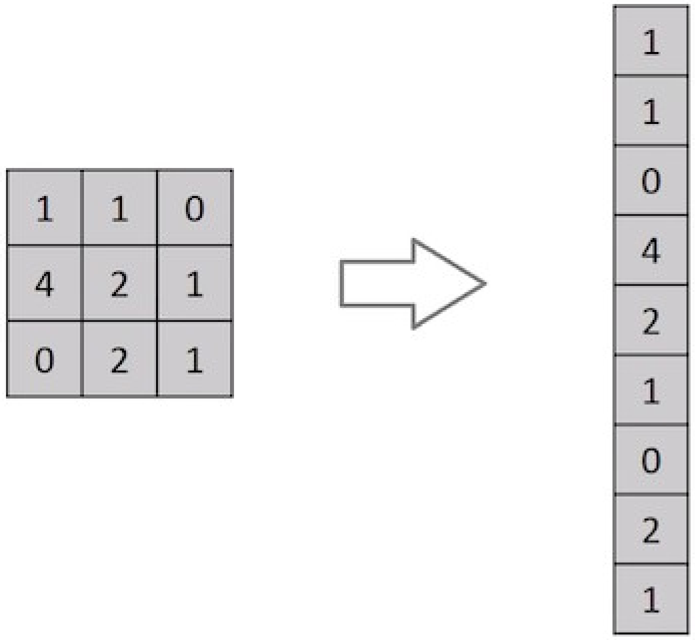
Building CNN



CNN Architecture



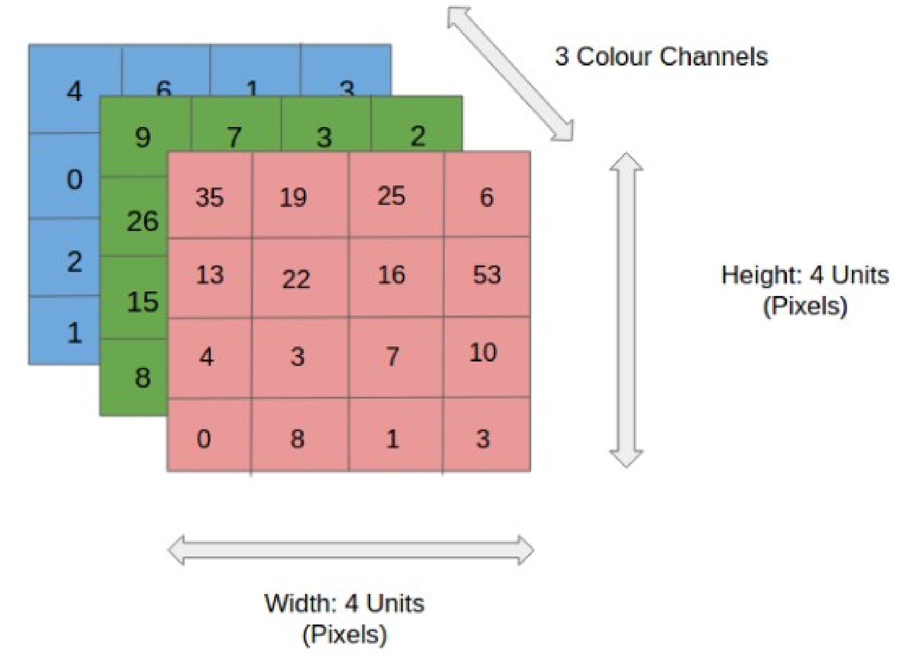
Why ConvNet of FF Nets?



Why ConvNet over FF NN ?

- A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters.
- The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.
- In other words, the network can be trained to understand the sophistication of the image better.

Input image formatting



The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

Convolution Layer

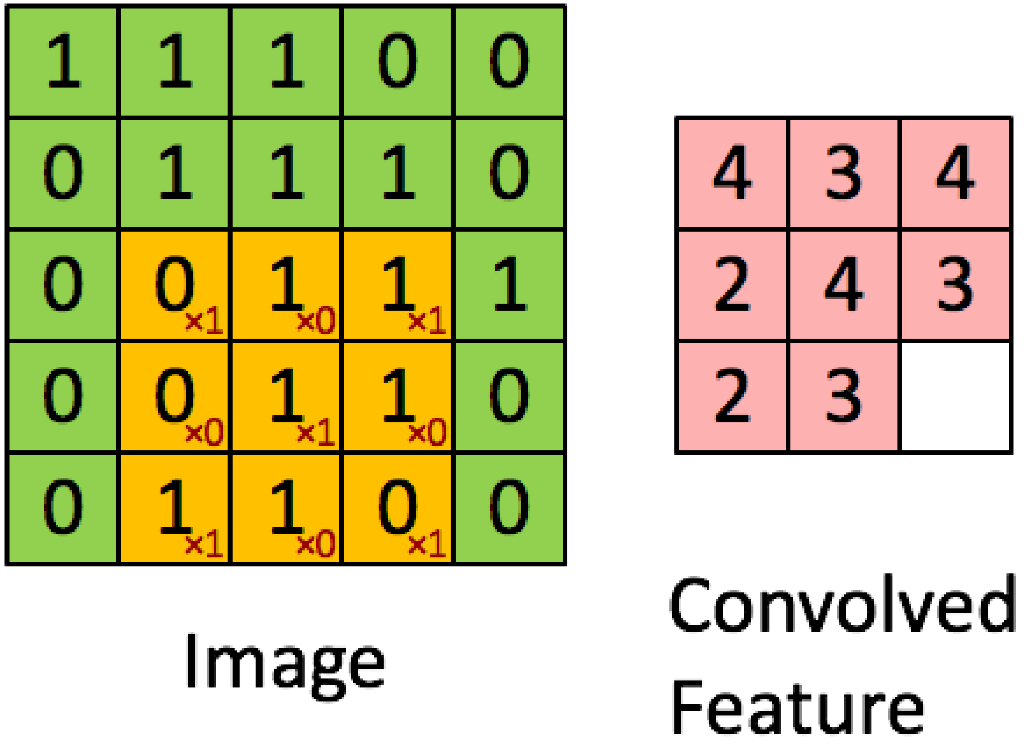
- Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)
- The green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.
- Kernel/Filter, K =

  1 0 1

  0 1 0

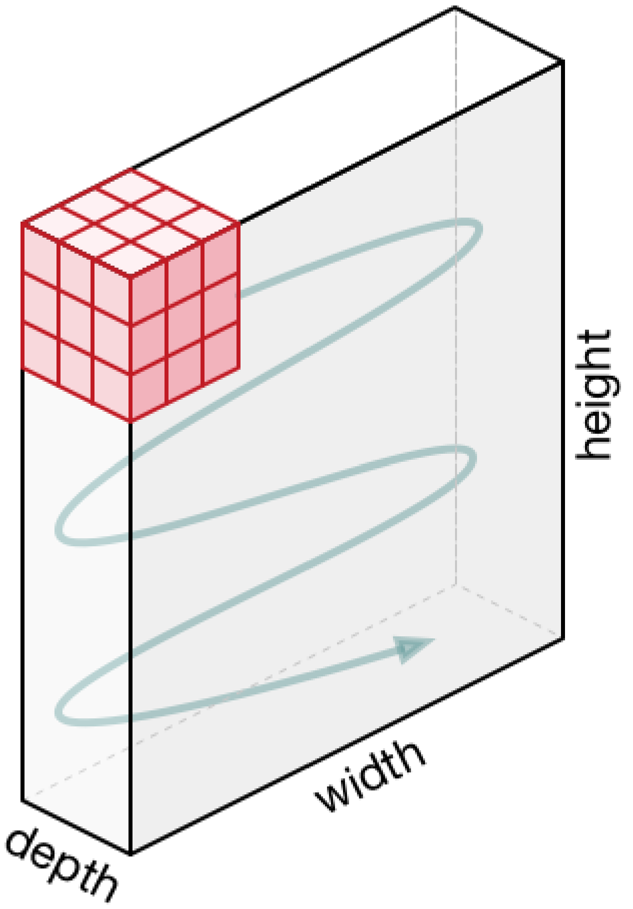
  1 0 1

Convolution Layer

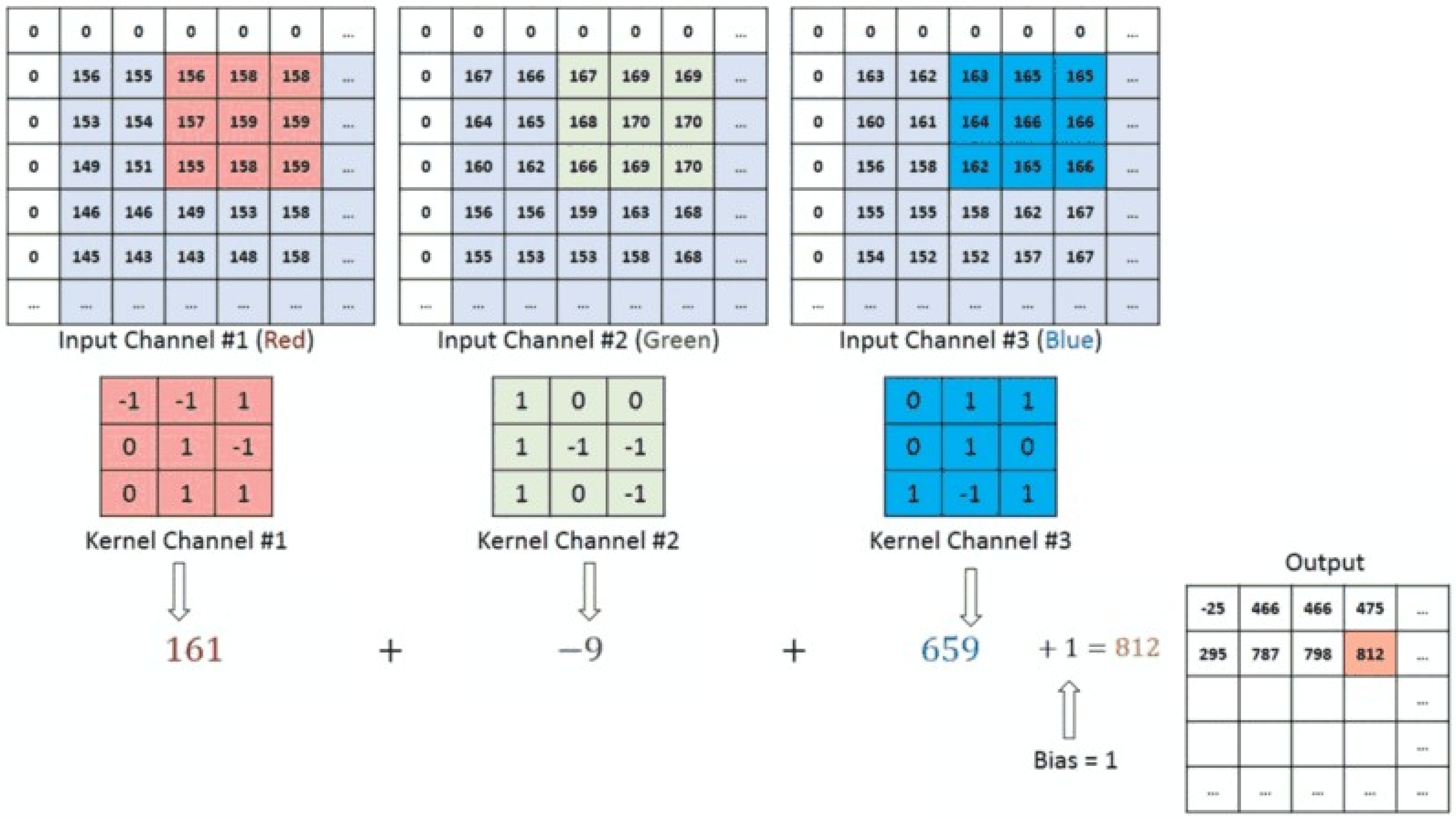


Movement of the Kernel

- The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.
- The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops dow to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.



Movement of the Kernel



Input Channel #1 (Red)

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

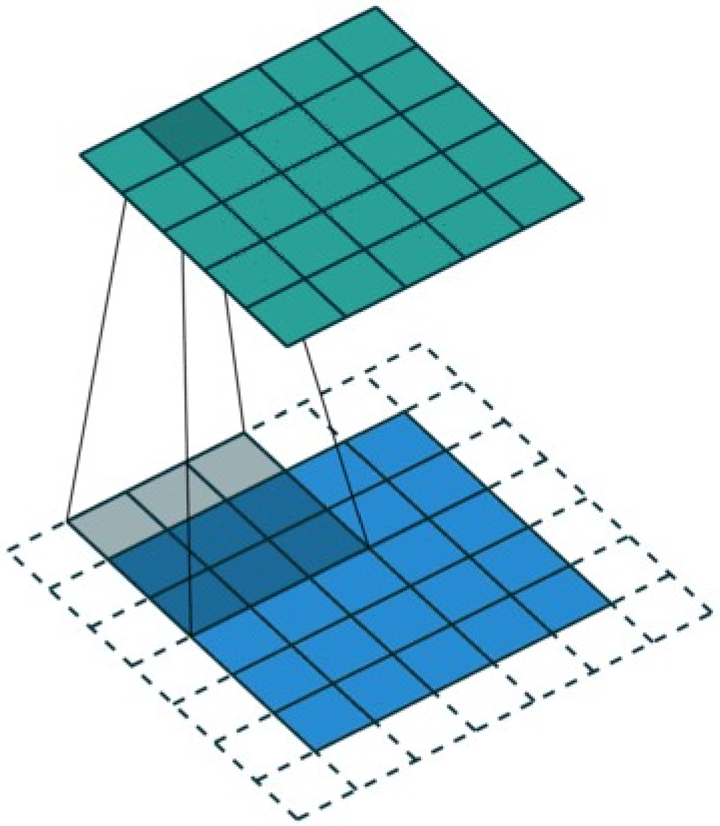
$$161 + -9 + 659 + 1 = 812$$

Bias = 1

Output

-25	466	466	475	...
295	787	798	812	...
				...
				...
...	...	...	...	...

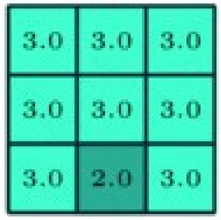
Same Padding

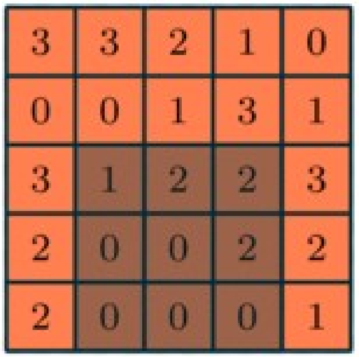


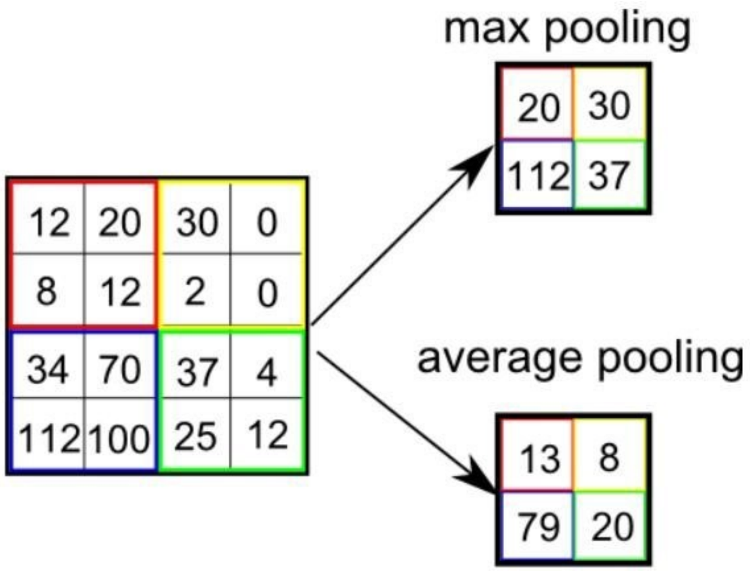
Pooling Layer

- The Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction.
- It is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.
- There are two types of Pooling: Max Pooling and Average Pooling.
  - Max Pooling returns the maximum value from the portion of the image covered by the Kernel.
  - Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

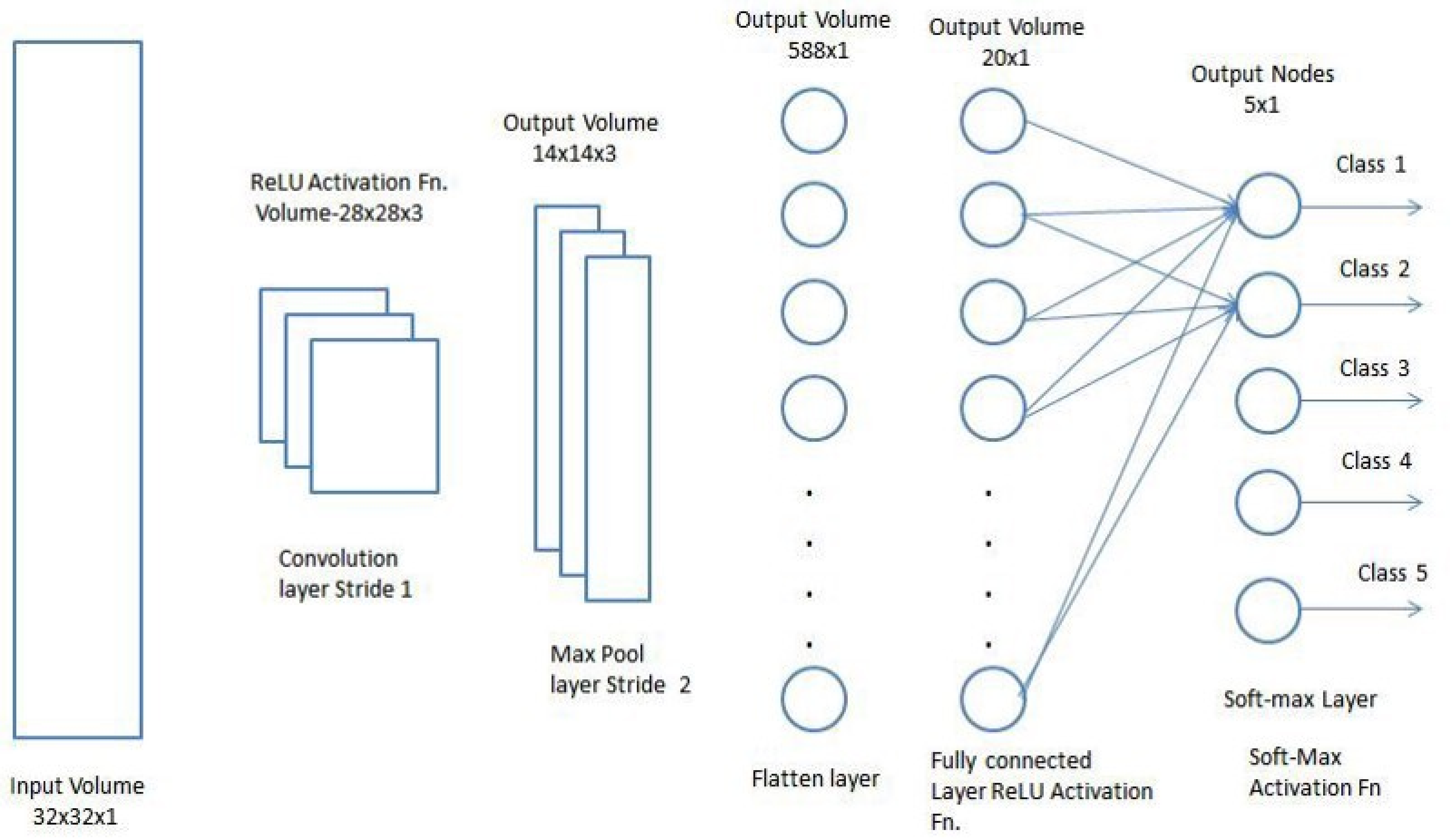
Pooling Layer







Classification



Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
http://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com