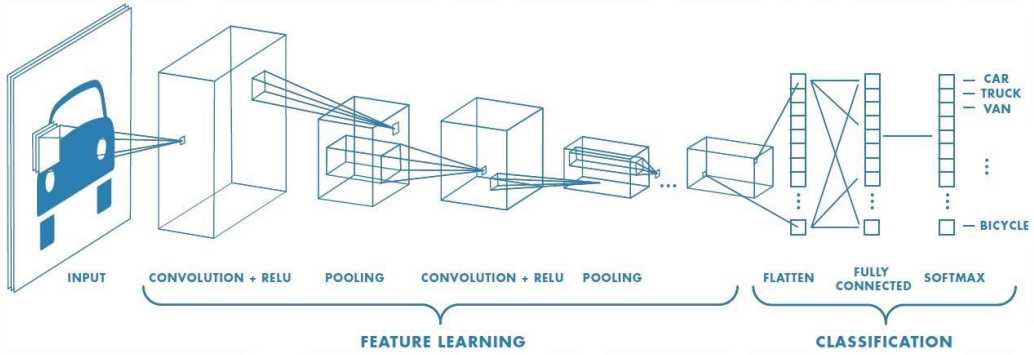
6.2.1 Convolutional Neural Network (CNN)

Dr. Sultan Alfarhood

Introduction

- A **convolutional neural network (CNN)** is a special kind of FFNN that significantly reduces the number of parameters in a deep neural network with many units without losing too much in the quality of the model.
- Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as **image classification**
  - Have also found success in natural language processing for text classification.



Convolutional Neural Network Layers

Convolutional layer

- Its purpose is to detect the presence of a set of features in the images received as input.

Pooling layer

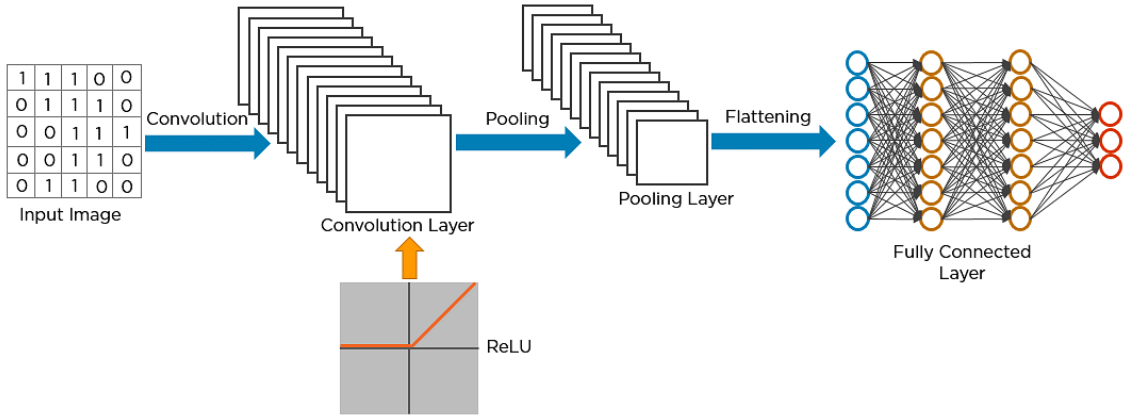
- Reducing the spatial volume of input image after convolution.

Flatten Layer

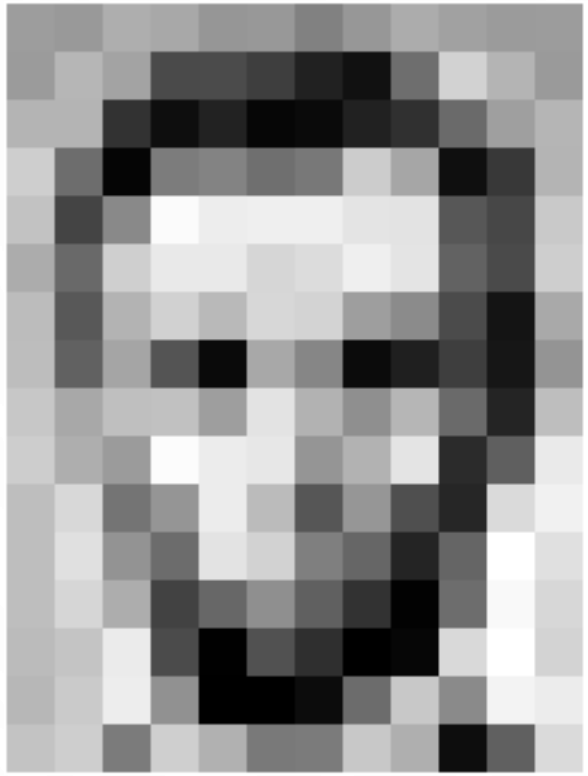
- Used to make the multidimensional input one-dimensional.

Fully Connected Layer

- The last fully-connected layer classifies the image as an input to the network: it returns a vector of size N, where N is the number of classes



Representation of Image as a Grid of Pixels

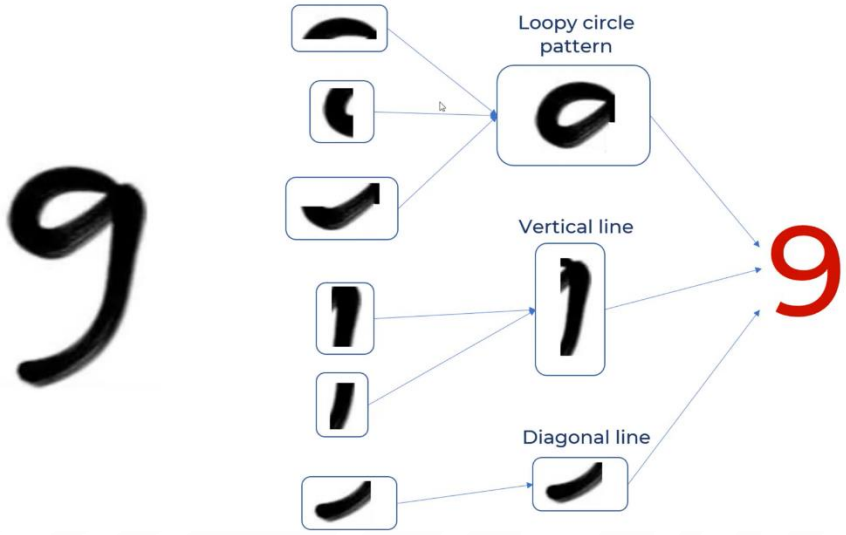


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

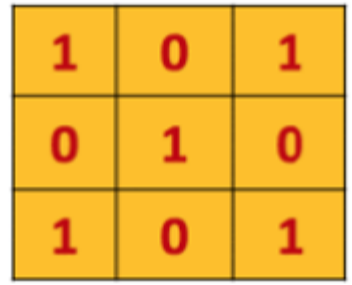
Convolutional layer

- The convolutional layer is the key component of convolutional neural networks and is always at least their first layer.
- Its purpose is to detect the presence of a set of features in the images received as input.
  - This is done by convolution filtering: the principle is to “drag” a window representing the feature on the image, and to calculate the convolution product between the feature and each portion of the scanned image.
  - A feature is then seen as a **filter**
- The filter matrix (one for each filter in each layer) and bias values are trainable parameters that are optimized using gradient descent with backpropagation.

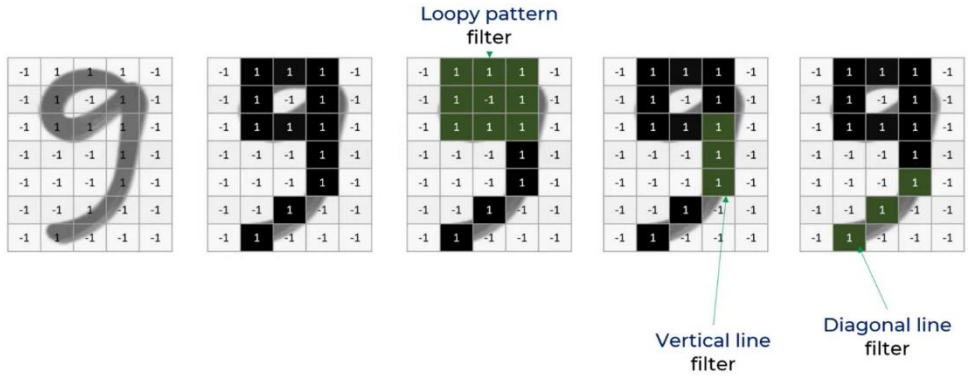


Filter (Kernel)

- A **filter** provides a measure for how close a patch or a region of the input resembles a feature
- A **feature** may be any prominent aspect – a vertical edge, a horizontal edge, an arch, a diagonal, etc.
- In CNN , **filters** detect patterns such as edges in an image by detecting the changes in intensity values of the image.

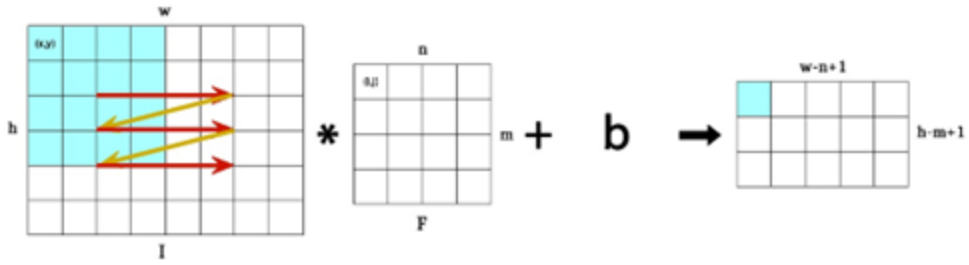


Filter / Kernel



The role of bias in Neural Networks

- **Bias** allows you to shift the activation function by adding a constant (the given bias) to the input



Example

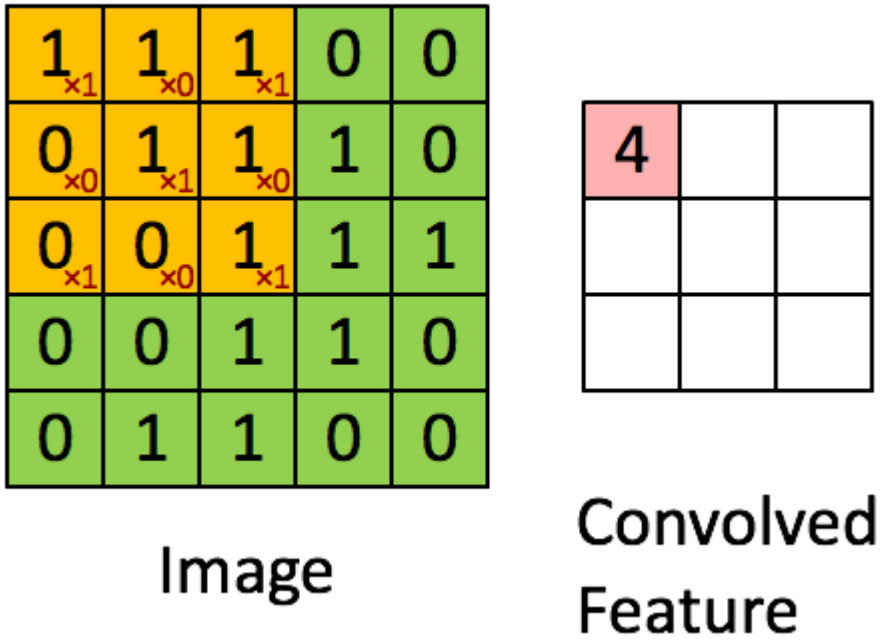
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

1	0	1
0	1	0
1	0	1

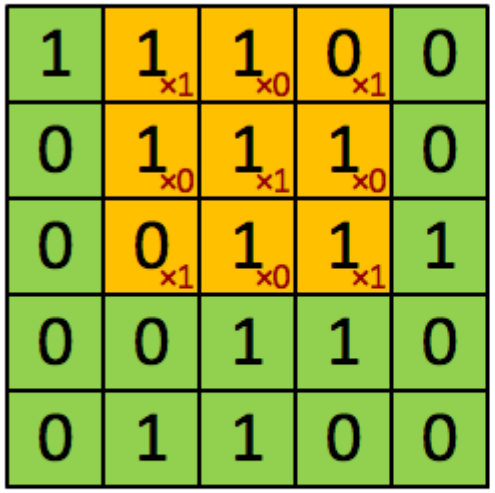
Filter

Example

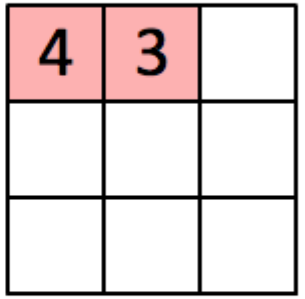


Assuming Bias=0

Example



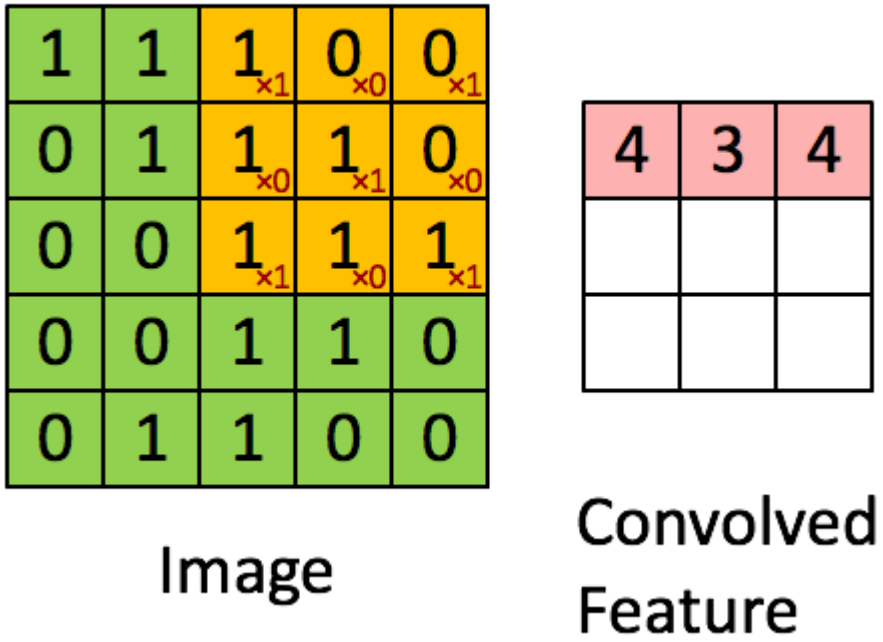
Image



Convolved Feature

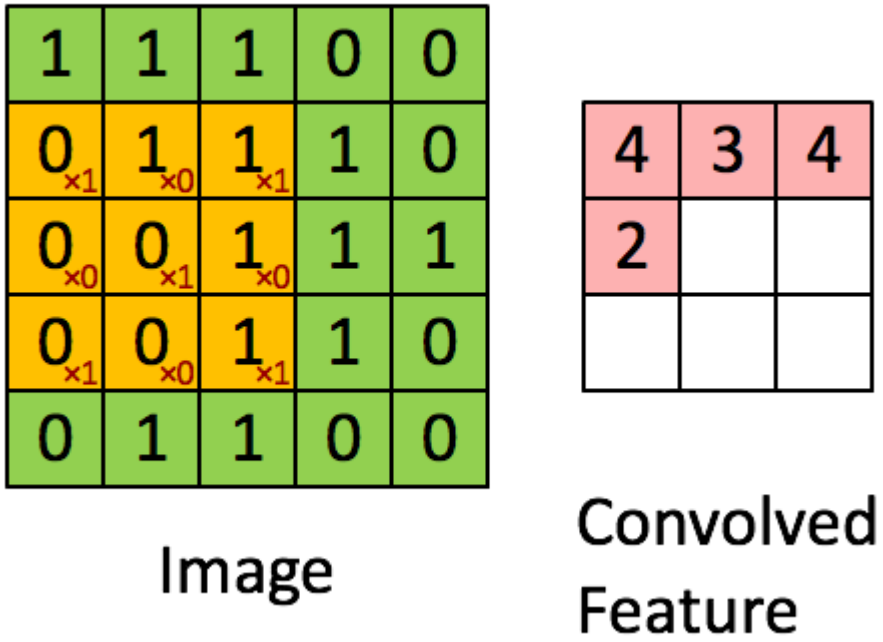
Assuming Bias=0

Example



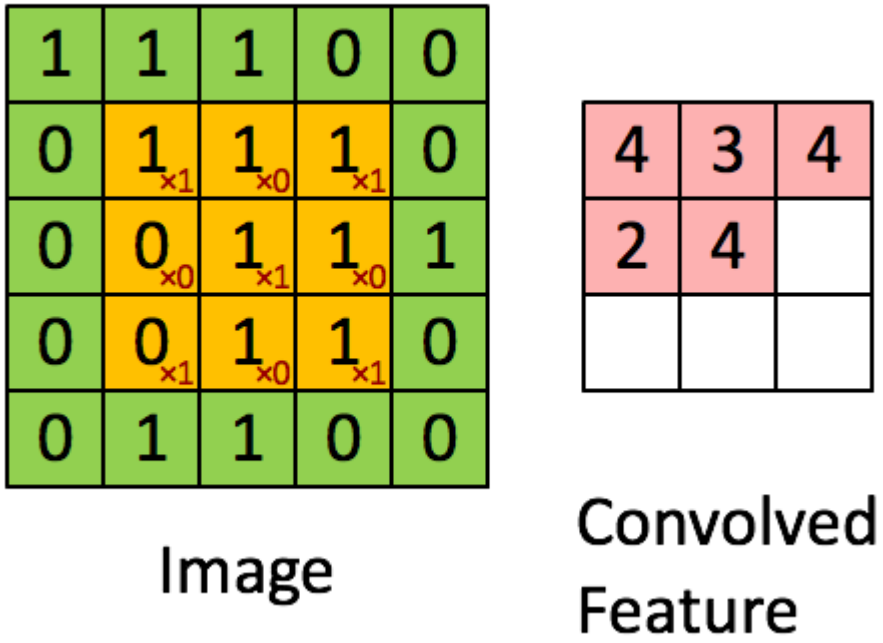
Assuming Bias=0

Example



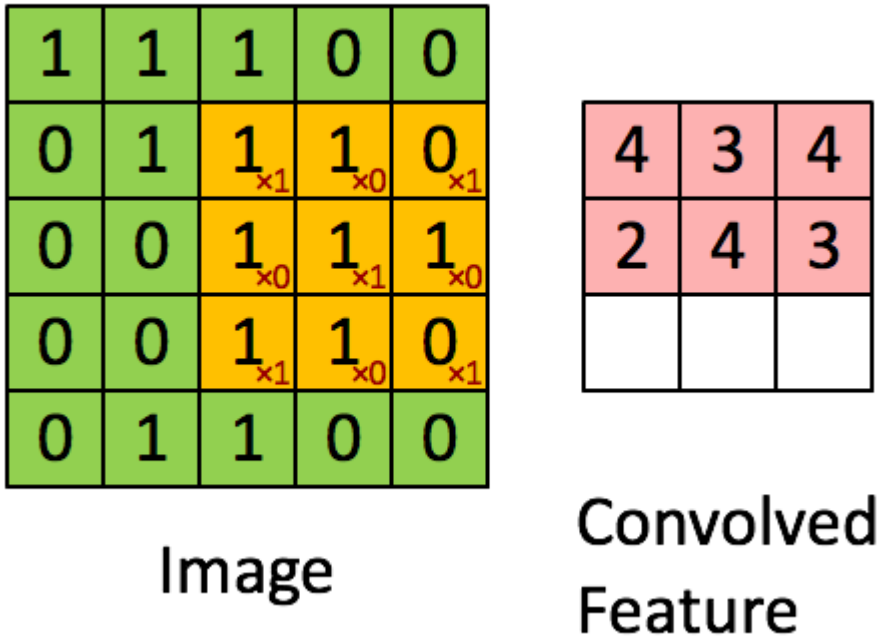
Assuming Bias=0

Example



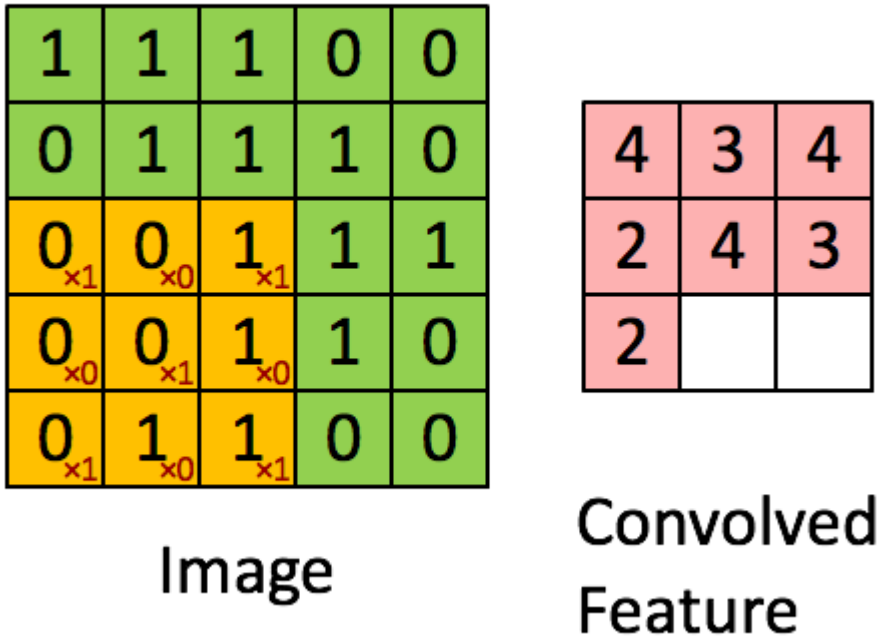
Assuming Bias=0

Example



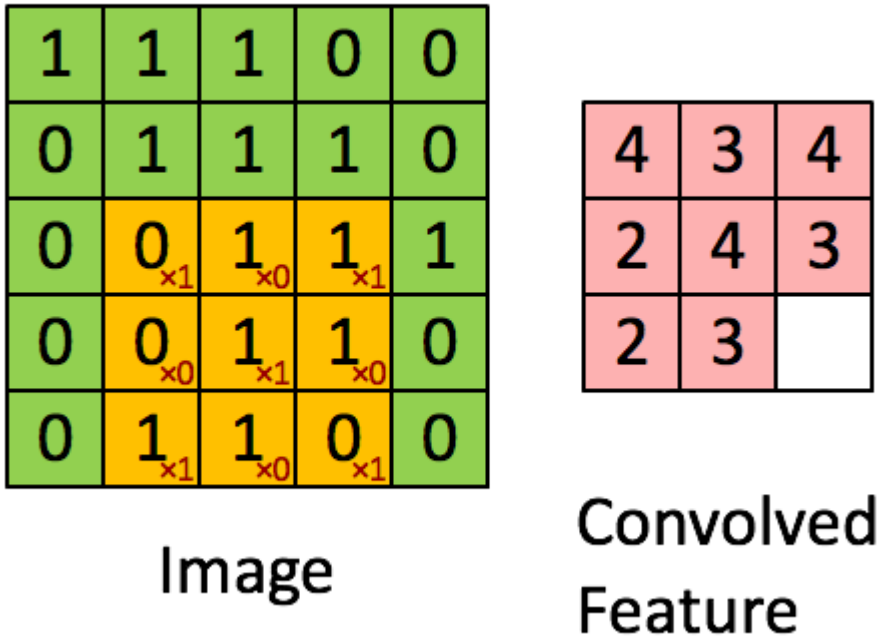
Assuming Bias=0

Example



Assuming Bias=0

Example



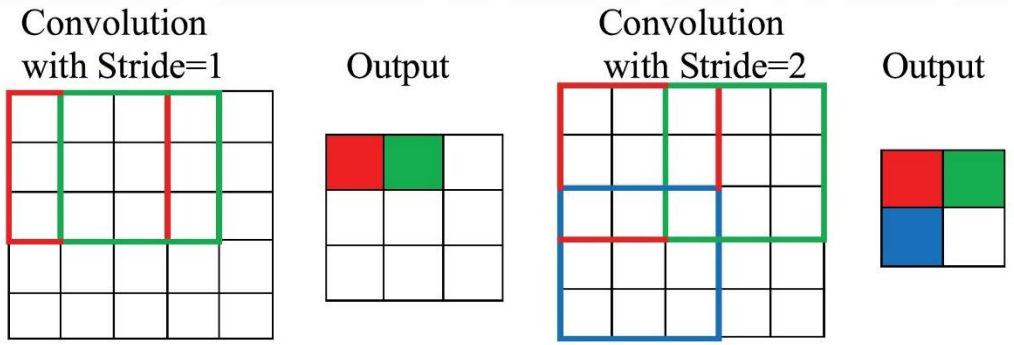
Assuming Bias=0

Example

Assuming Bias=0

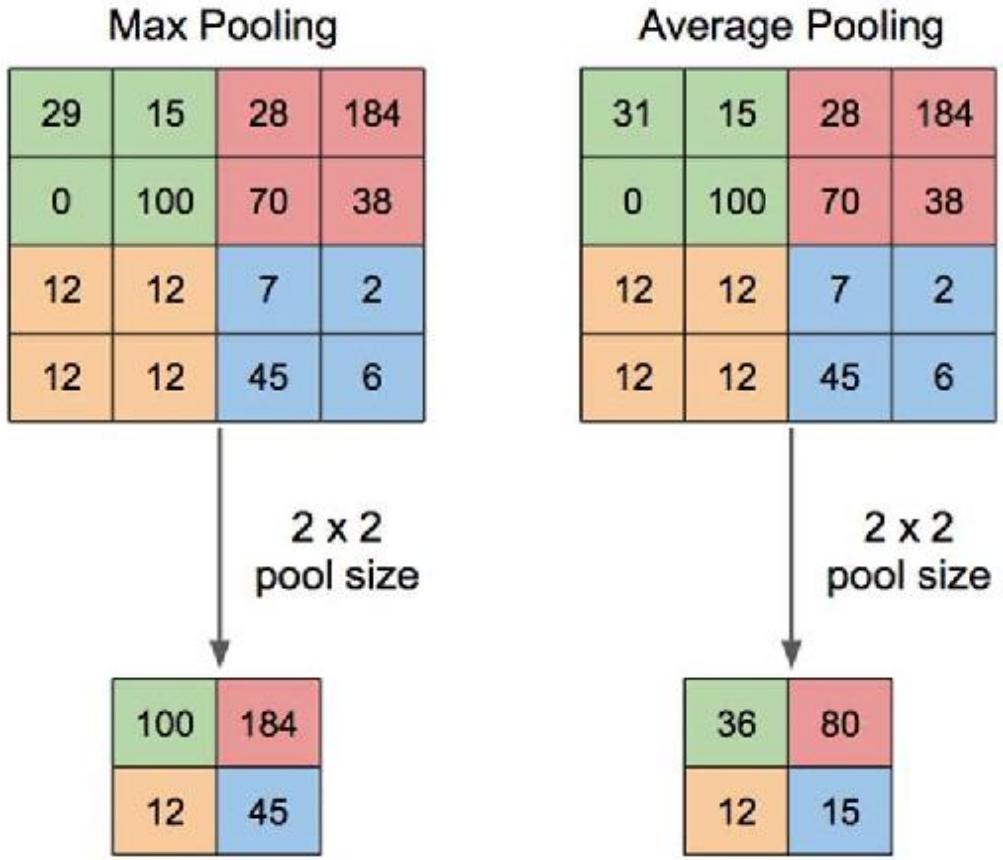
Stride

- Stride is the step size of the moving window.
  - how many steps we are moving in each steps in convolution.



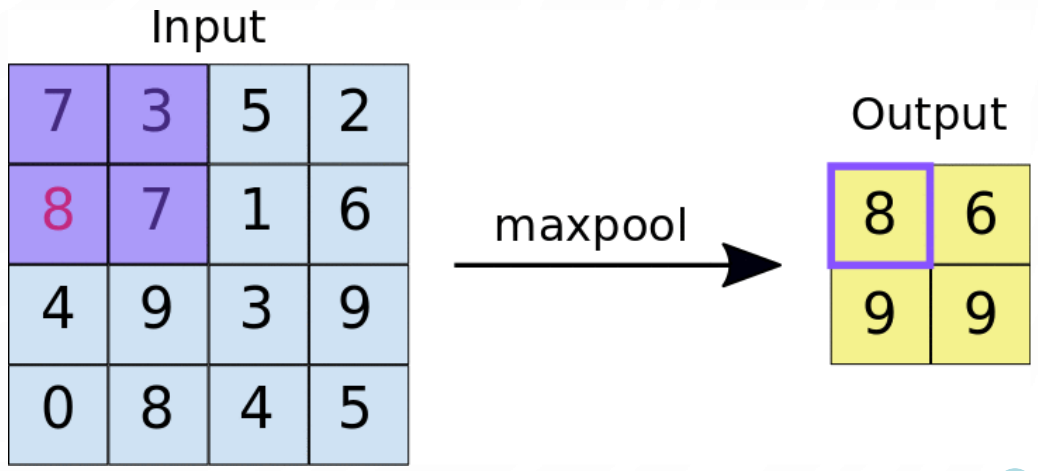
Pooling Layer

- The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs.
- There are two types of poolings:
  - **Max pooling**
    - This works by selecting the maximum value from every pool.
    - Max Pooling retains the most prominent features of the feature map, and the returned image is sharper than the original image.
  - **Average pooling**
    - This pooling layer works by getting the average of the pool.
    - Average pooling retains the average values of features of the feature map.
    - It smoothes the image while keeping the essence of the feature in an image.

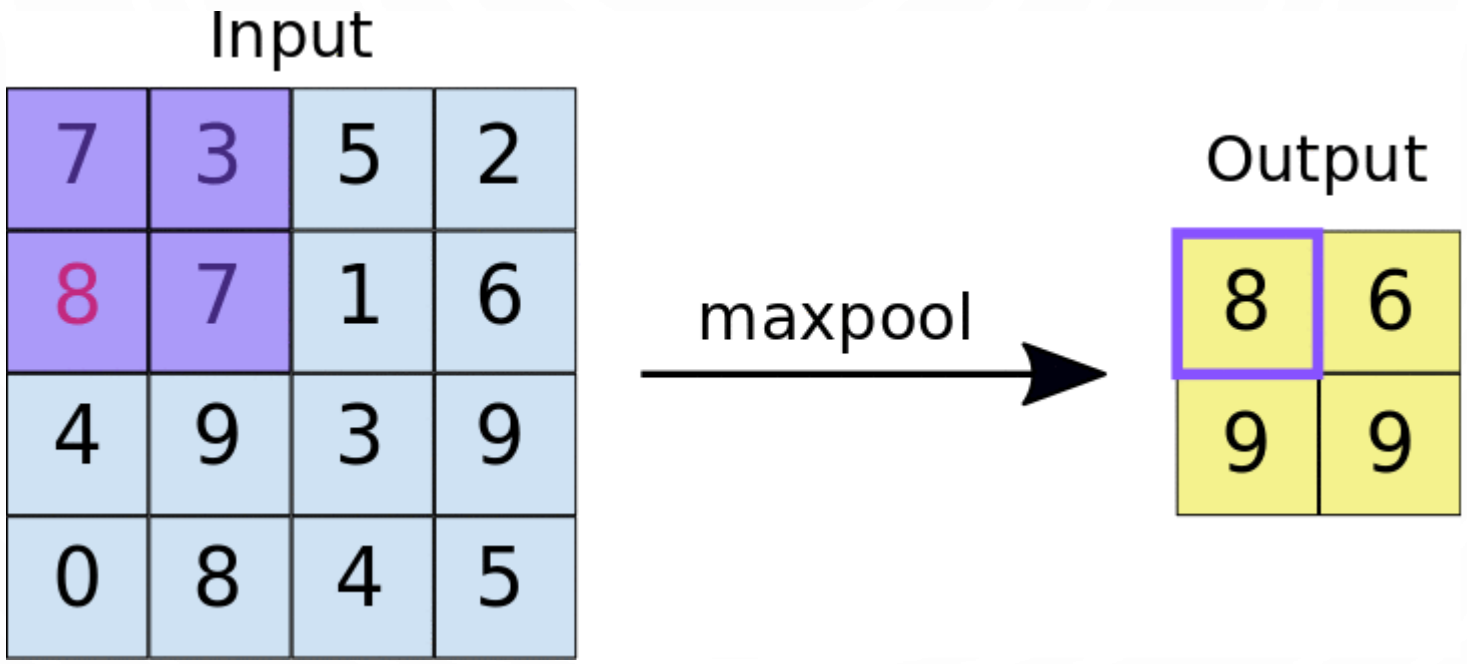


Max Pooling

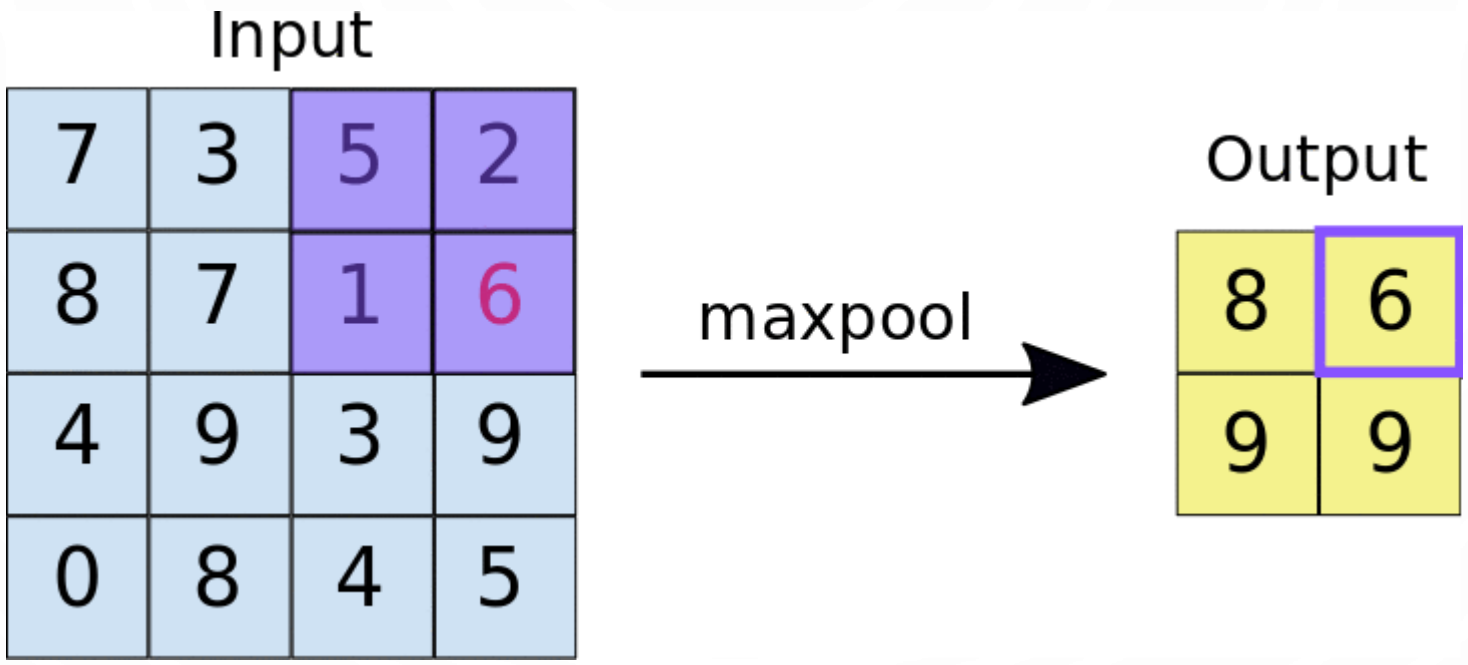
- Steps:
  1. Pick sliding matrix dimensions
  2. Pick the strides
  3. Slide your matrix across the filtered images
  4. Take the maximum value in the sliding matrix
  5. Repeat until the whole matrix (original) has been traversed



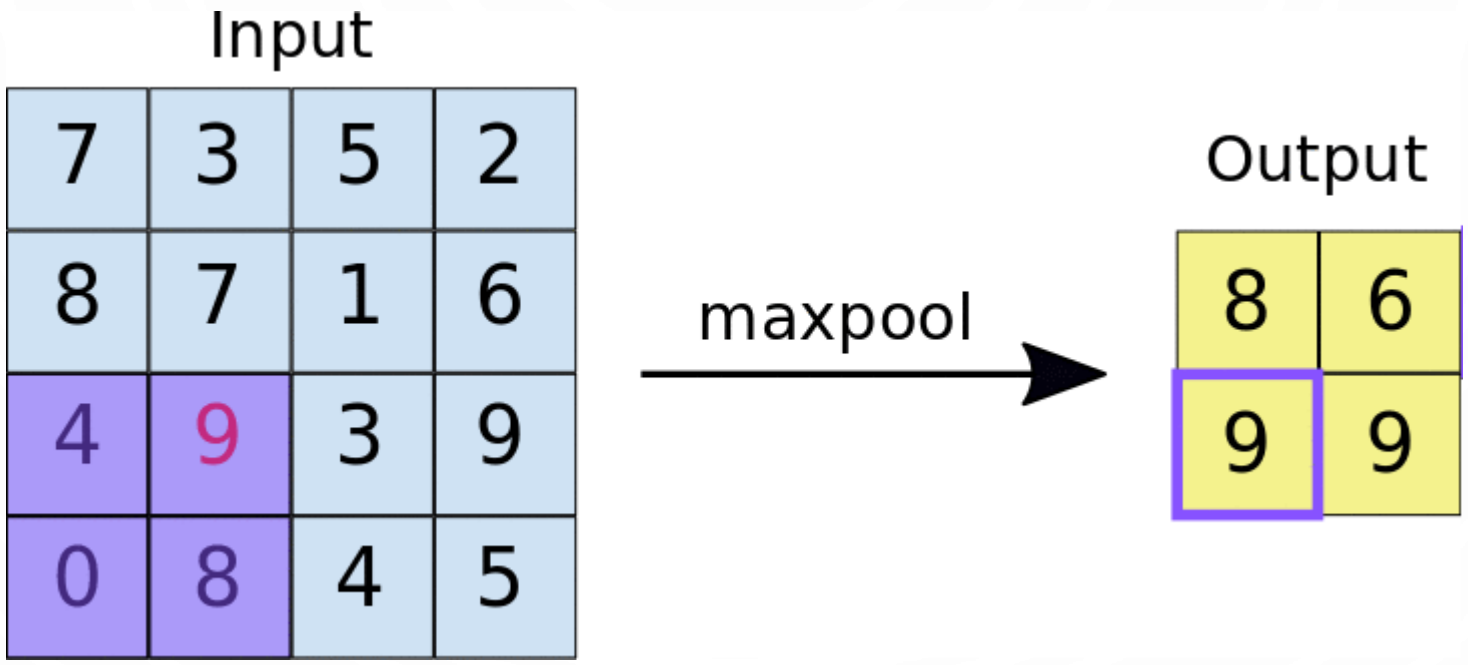
Max Pooling Example



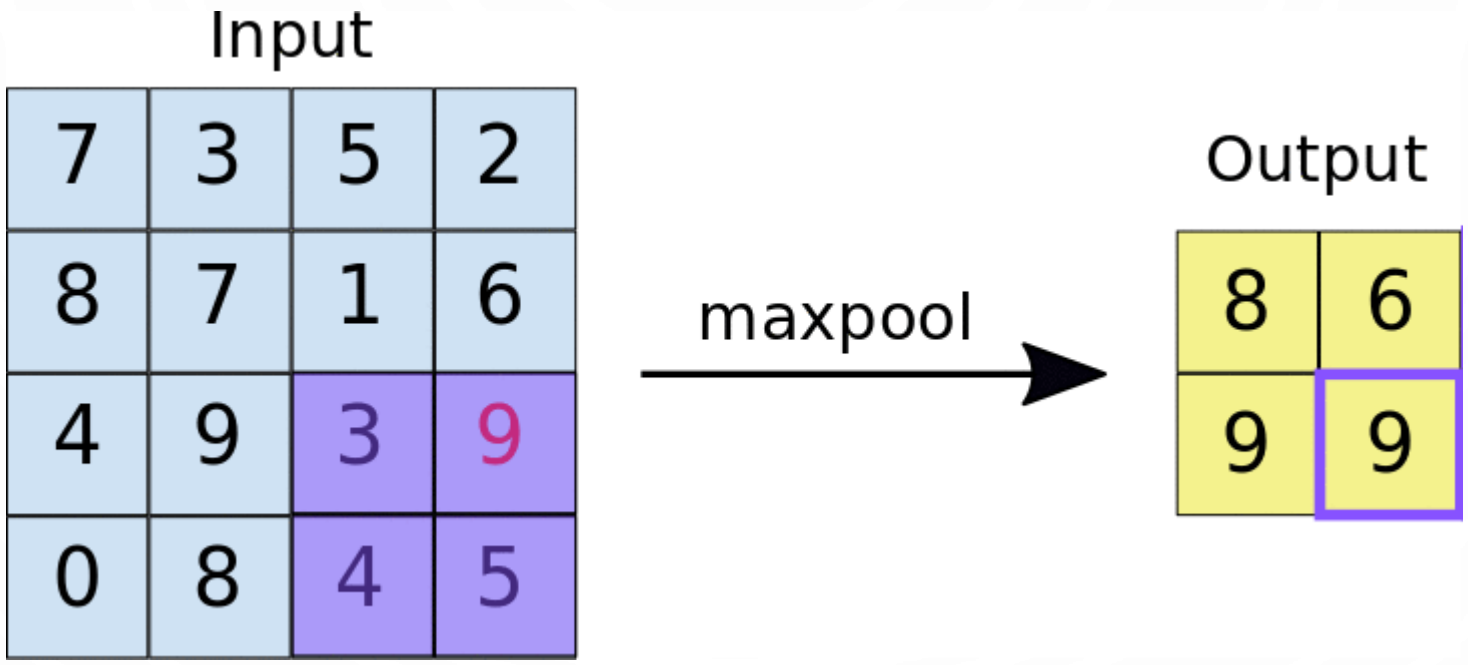
Max Pooling Example



Max Pooling Example



Max Pooling Example



Why to use Max Pooling?

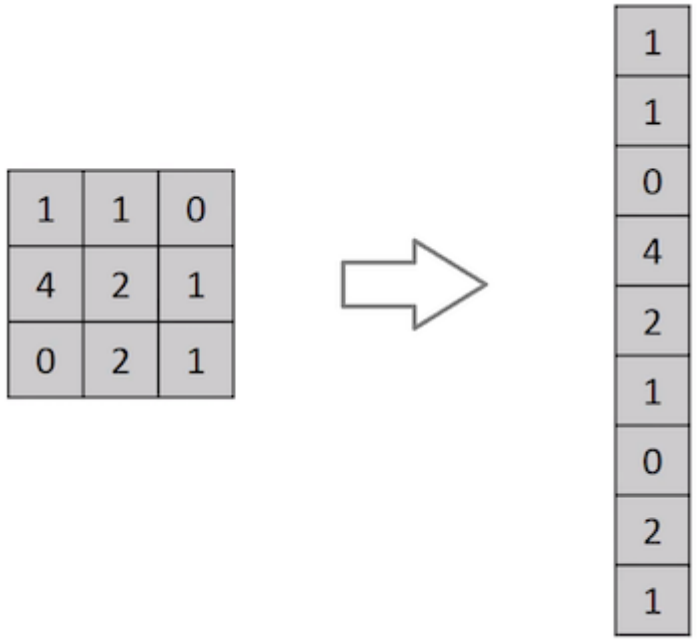
After discarding non-max pixels and selecting only the max pixels, the data is compressed while retaining **maximal information** through the activated pixels (with highest values).

Decreases dimensionality.

Decreases the number of **parameters** due to the network looking at less dimensions of images at a time which in turn **reduces computational** load.

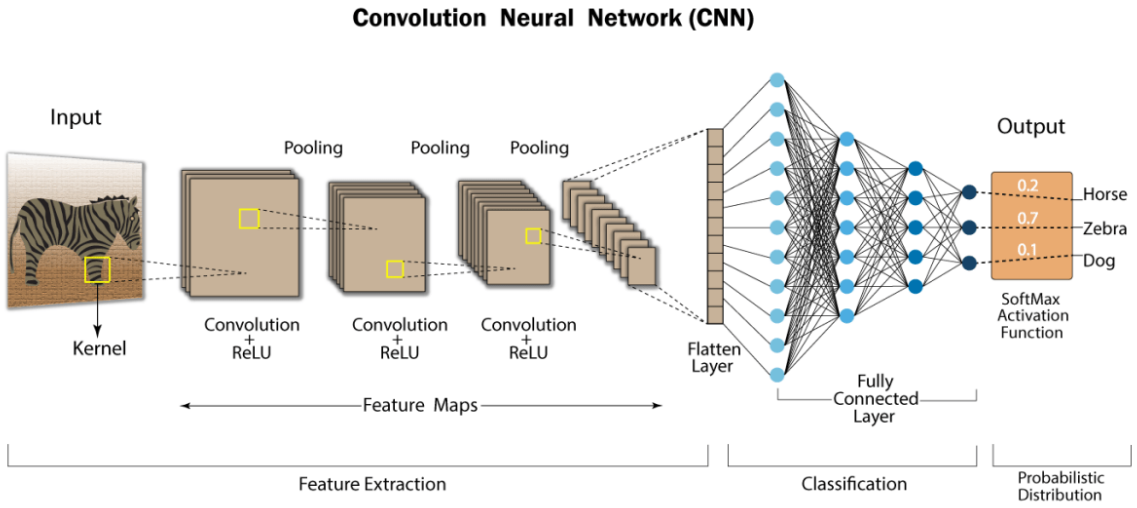
Flatten layer

- **Flatten layer** is used to make the multidimensional input one-dimensional
- It is commonly used in the transition from the convolution layer to the full connected layer.

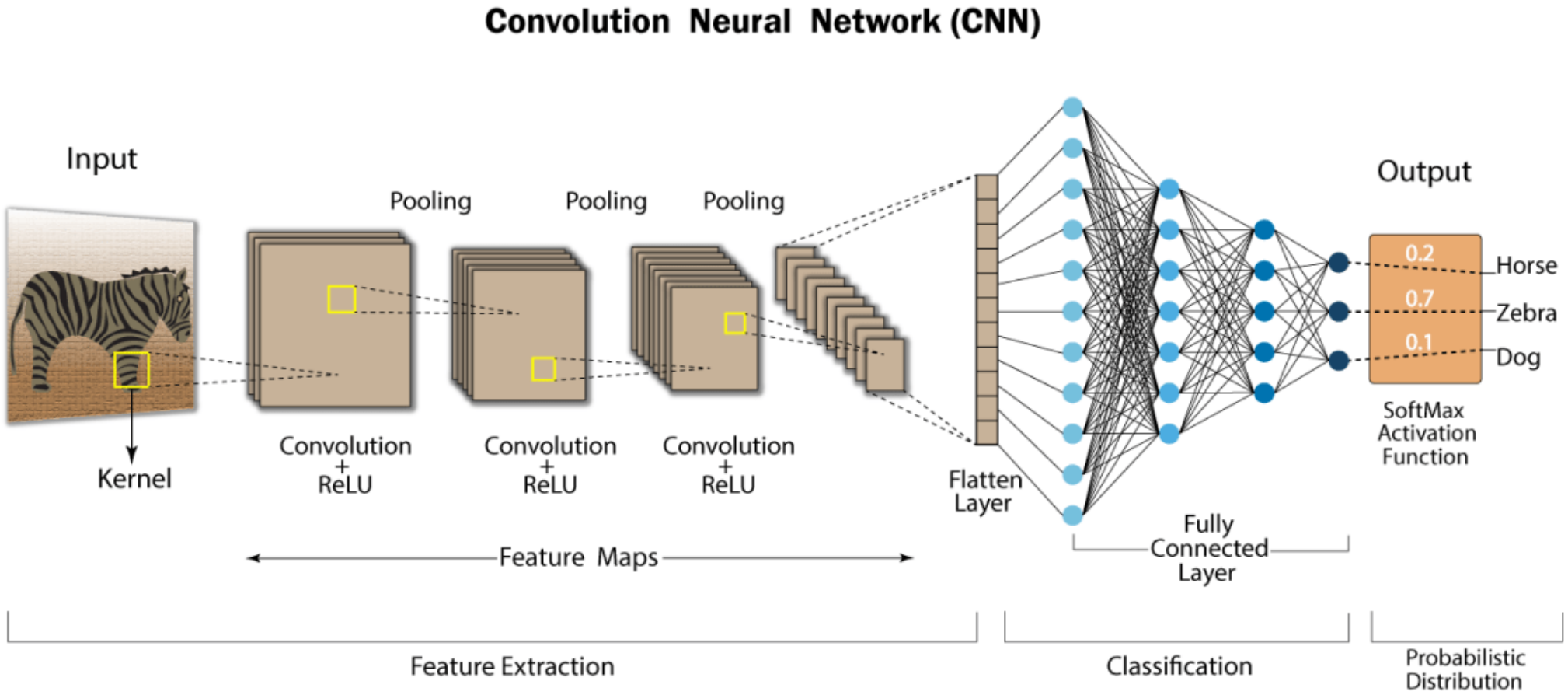


Fully Connected Layers

- The fully-connected layer is always the last layer of a neural network
- The last fully-connected layer classifies the image as an input to the network
  - It returns a vector of size N, where N is the number of classes in our image classification problem.
  - Each element of the vector indicates the probability for the input image to belong to a class.
- Usually applies an activation function
  - Logistic if N=2
  - Softmax if N>2

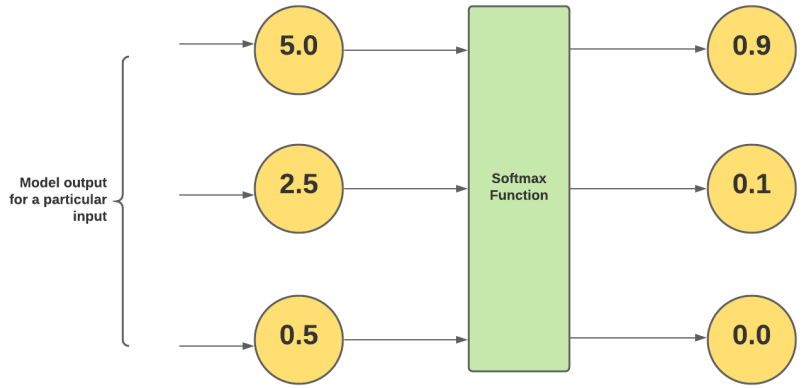


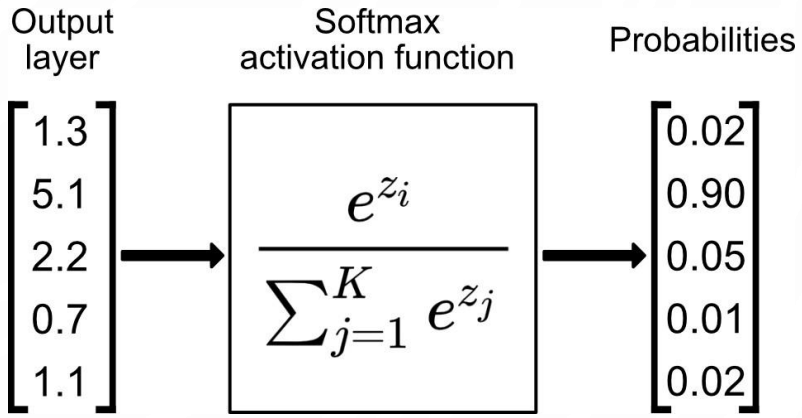
Convolutional Neural Network Example



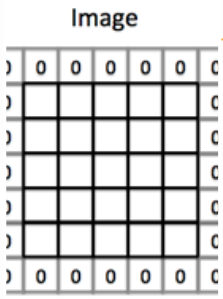
SoftMax

- **SoftMax** is used only for the output layer, for neural networks that need to classify inputs into multiple categories.
- It normalizes the outputs for each class between 0 and 1 and divides by their sum.



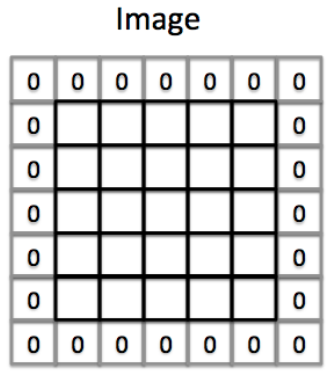


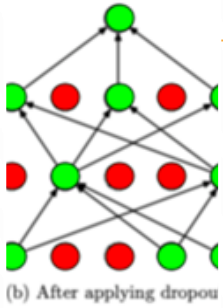
Other Layers & Operations



Padding

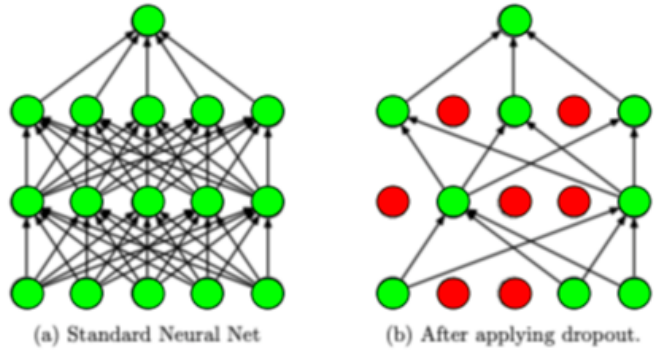
- A process of surrounding the image (or input) before convolving it with a filter (usually contain zeroes).





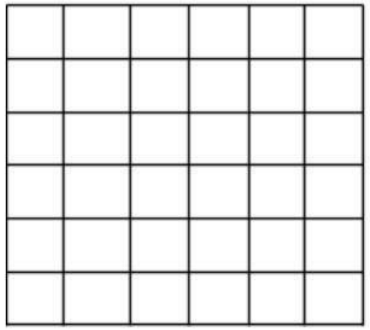
Dropout

- A mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others.

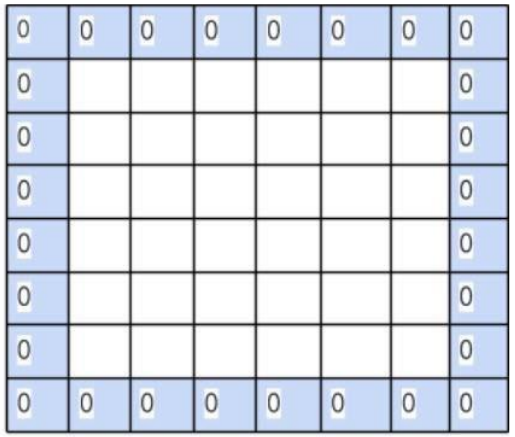


Padding

- **Padding** allows getting a larger output matrix; it's the width of the square of additional cells with which you surround the image (or volume) before you convolve it with the filter.
- The cells added by padding usually contain zeroes.



6x6 image



6x6 image with 1 layer of zero padding

Padding Layer Example (Animated)

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

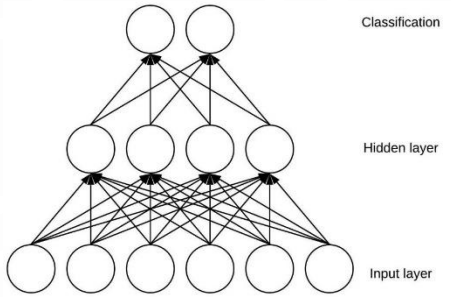
Kernel

0	-1	0
-1	5	-1
0	-1	0

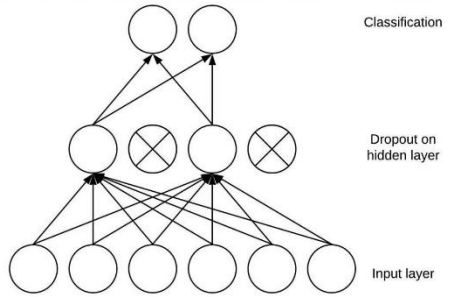
114				

Dropout

- **Dropout** refers to dropping out the nodes (input and hidden layer) in a neural network.
  - All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network.
  - The nodes are dropped by a dropout probability of *p*.
- Dropout layers are important in training CNNs because they **prevent overfitting** on the training data.
  - If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner.
  - This, in turn, would prevent the learning of features that appear only in later samples or batches



Without Dropout



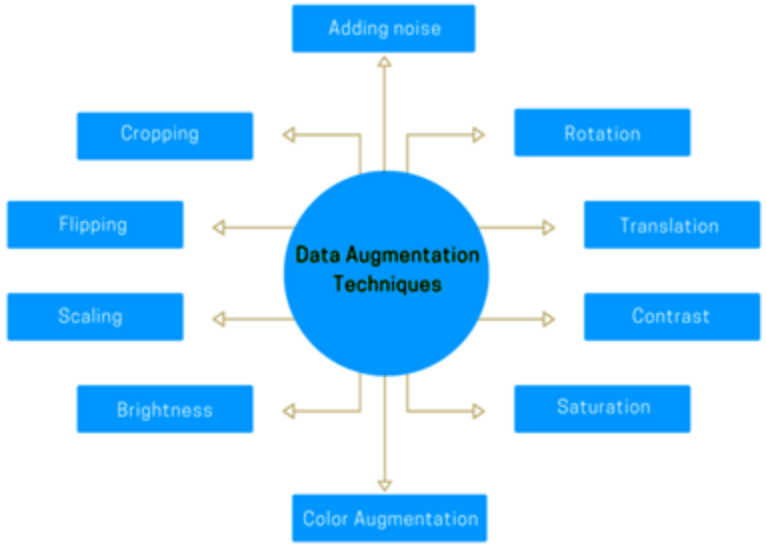
With Dropout

Data Augmentation

- Data augmentation is a process of artificially increasing the amount of data by generating **new data points from existing data**.
- This includes adding minor alterations to data or using machine learning models to generate new data points in the latent space of original data to amplify the dataset.

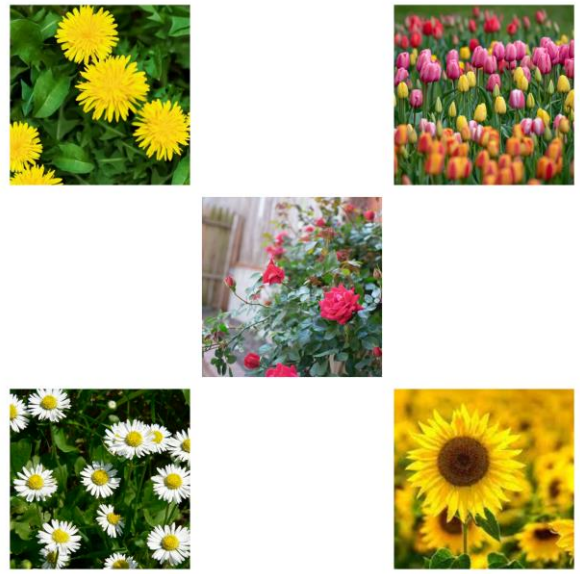
Data Augmentation Techniques

- **Position**
  - **Center Crop**: Crops the given image at the center. Size is the parameter given by the user.
  - **Random Crop**: Crop the given image at a random location.
  - **Random Vertical Flip**: Vertically flips the given image randomly with a given probability.
  - **Random Horizontal flip**: Horizontally flip the given image randomly with a given probability.
  - **Random Rotation**: Rotate the image by some angle.
  - **Resize**: Resize the size of the input image to a given size.
  - **Random Affine**: Random affine transformation of the image keeping center invariant.
- **Color**
  - **Brightness:** One way to augment is to change the brightness of the image. The resultant image becomes darker or lighter compared to the original one.
  - **Contrast:** The contrast is defined as the degree of separation between the darkest and brightest areas of an image. The contrast of the image can also be changed.
  - **Saturation:** Saturation is the separation between the colors of an image.



Python Example

- This tutorial shows how to classify images of flowers.
- This tutorial follows a basic machine learning workflow:
  - Examine and understand data
  - Build an input pipeline
  - Build the model
  - Train the model
  - Test the model
  - Improve the model and repeat the process



https://colab.research.google.com/drive/1V7U9ybkcj8zut64pIX2ru0e8l3CE9X1C?usp=sharing