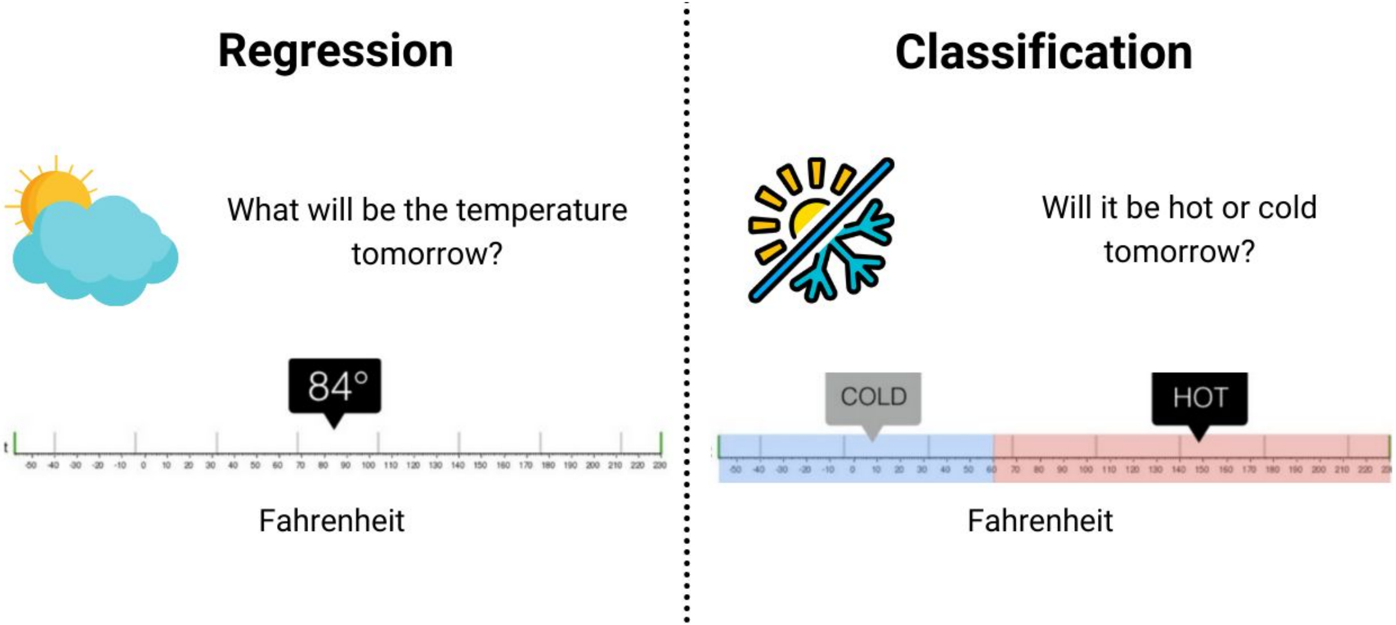
CSE4261: Neural Network and Deep Learning

Lecture: 21.05.2025

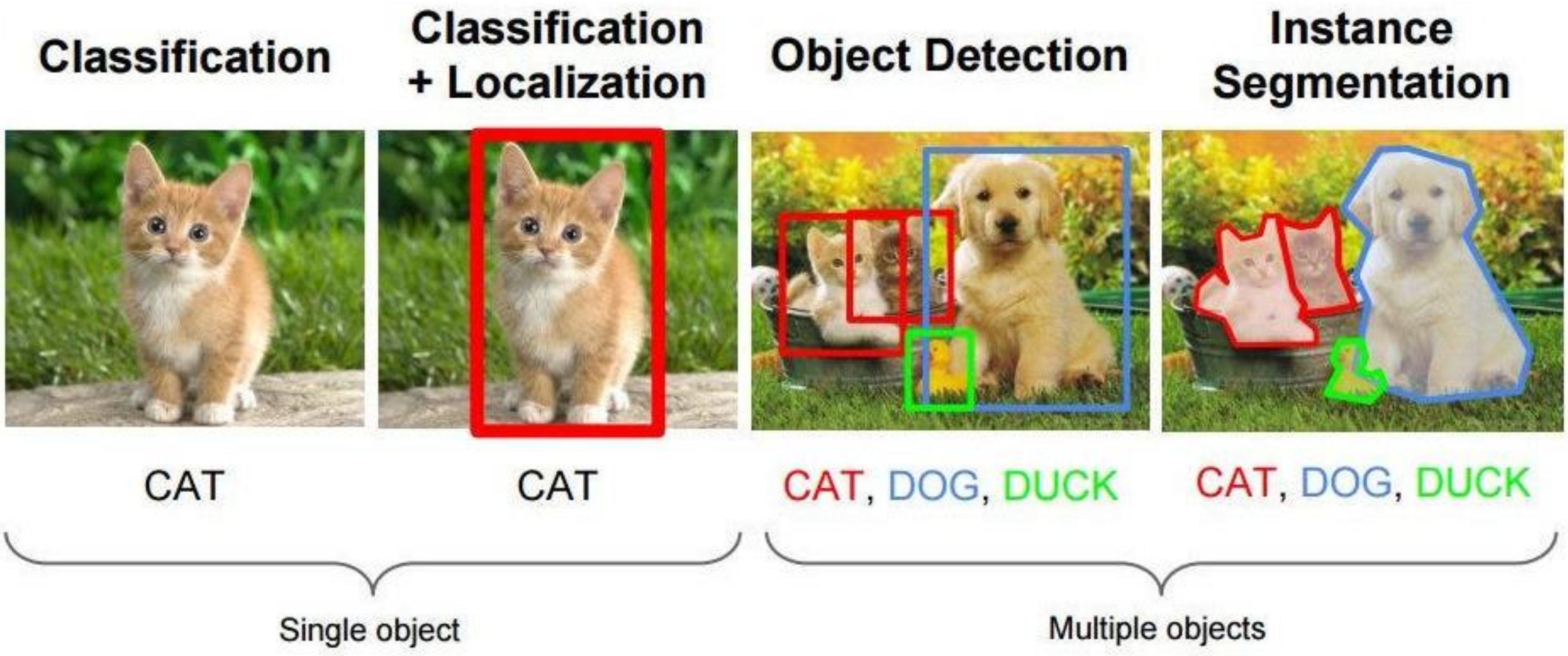
Sangeeta Biswas, Ph.D.
Associate Professor,
University of Rajshahi, Rajshahi-6205, Bangladesh

Regression Vs. Classification

Regression is a problem of predicting a continuous numeric value (e.g., a price, a temperature, a score) based on input data.

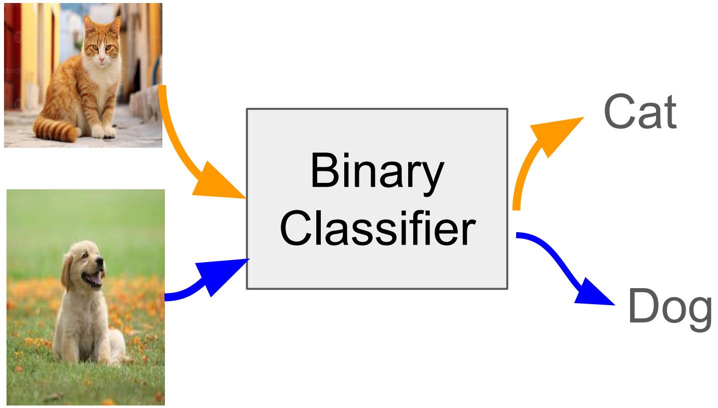


Computer Vision Tasks

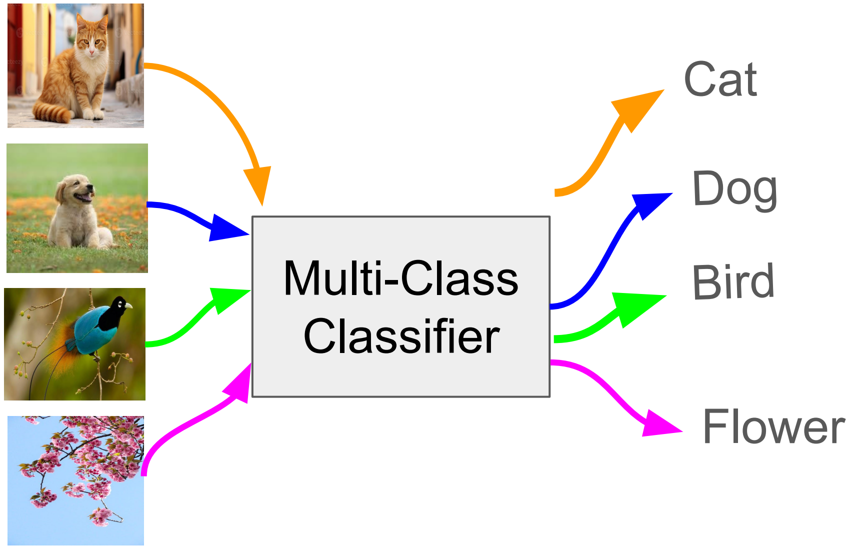


Classification

- Assigning objects to some pre-existing classes / categories / labels / groups.



Number of classes = 2



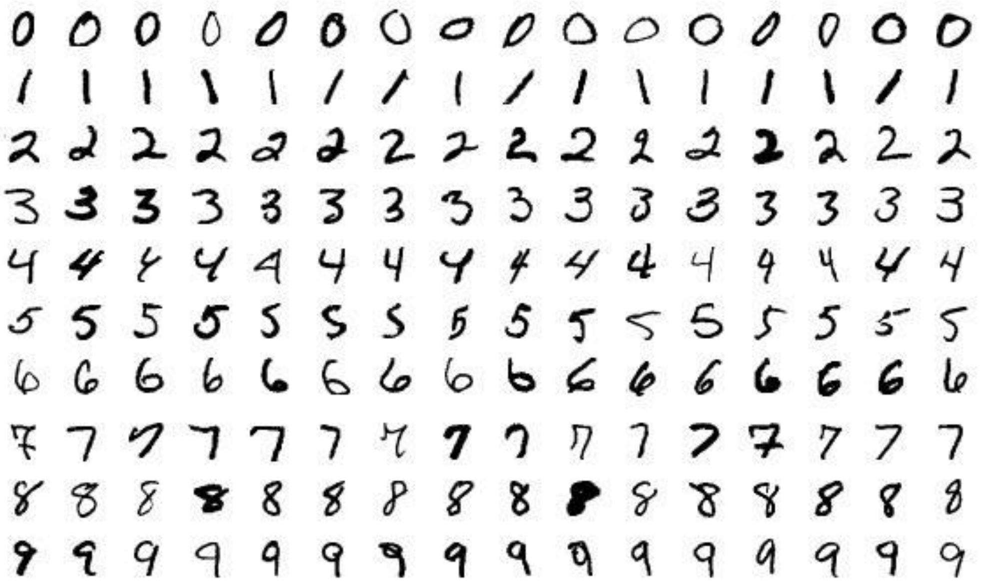
Number of classes = 4

What will We Use

- Linux Environment
- Tensorflow, Keras, PyTorch
- Python
- OpenCV
- Other Python Libraries such as matplotlib, scikit, pandas

MNIST Digit Dataset

- 70000 images 28 x 28
  - 60000 for training
  - 10000 for testing
- 10 classes
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 9



MNIST Fashion Dataset

- 70000 images 28 x 28
  - 60000 for training
  - 10000 for testing
- 10 classes
  - 0: T-shirt/top
  - 1: Trouser
  - 2: Pullover
  - 3: Dress
  - 4: Coat
  - 5: Sandal
  - 6: Shirt
  - 7: Sneaker
  - 8: Bag
  - 9: Ankle boot

Other Datasets for Classification

- CIFAR 10 dataset & CIFAR 100 dataset
  - https://www.cs.toronto.edu/~kriz/cifar.html
  - 60000, 32x32 colour images in 10 classes, with 6000 images per class
  - 50000 training images and 10000 test images
- CIFAR 100 dataset
  - https://www.cs.toronto.edu/~kriz/cifar.html
  - 100 classes containing 600 images each
  - 500 training images and 100 testing images per class
- ImageNet 1000 dataset
  - https://www.image-net.org/
  - 1000 object classes
  - 1281167 training images, 50000 validation images and 100000 test images.

Pre-trained Classifiers Provided by Keras API Team

- Models were trained by ImageNet 1K dataset for 1000 classes

https://keras.io/api/applications/

- Most of the time, we depend on these pre-trained models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3

Code for Loading MNIST Fashion Dataset

```
#-- Load data
from tensorflow.keras.datasets import fashion_mnist
(trainX, trainY), (testX, testY) = fashion_mnist.load_data()
print(trainX.shape, trainX.dtype, trainY.shape, trainY.dtype)

#--- Cross check
plt.imshow(trainX[0])
plt.title(trainY[0])
plt.show()
plt.close()
```

Code for Loading Pre-trained Model

```
#--- Load the pre-trained model, VGG16, with head
from tensorflow.keras.applications import vgg16
vgg16_model = vgg16.VGG16()

#--- Display architecture
vgg16_model.summary(show_trainable = True)

#--- Load the pre-trained model, VGG16, without head
vgg16_model = vgg16.VGG16(include_top = False)
vgg16_model.summary(show_trainable = True)
```

Code for Training, Predicting and Evaluating

```
#--- Train a model
model.compile(loss_function, metric_list)
model.fit(trainX, trainY, epochs, batch_size, validation)

#--- Predict by a model
predictY = np.argmax(model.predict(testY))

#--- Evaluate by model
model.compile(loss_function, metric_list)
loss, metric1, metric2, … = model.evaluate(testX, testY)
```

Fully Connected Neural Network (FCNN)

FCNN consists of a series of fully connected layers

It is also known as multi-layer perceptron (MLP).

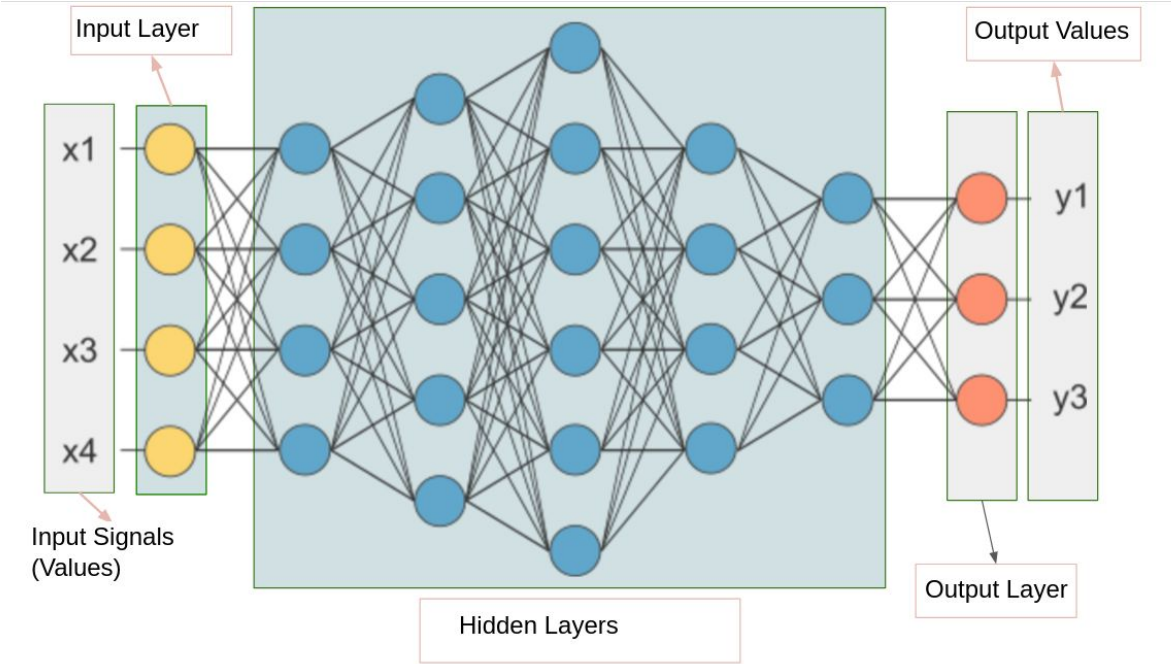


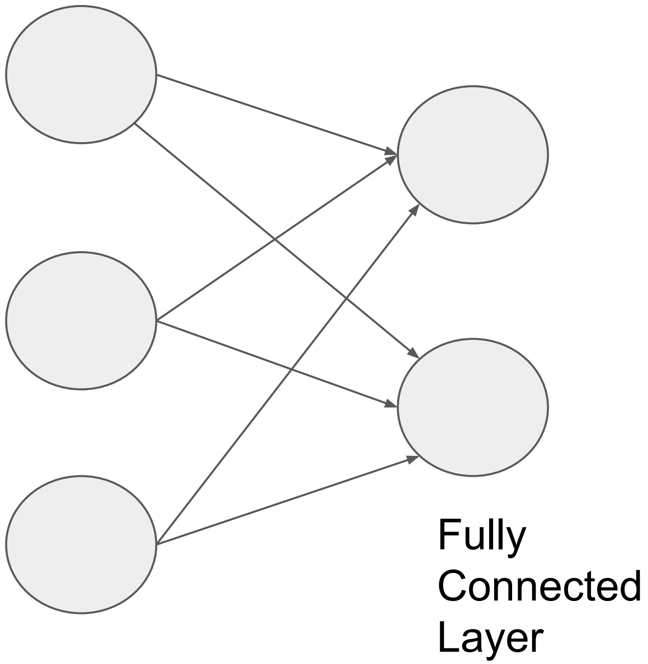
Image Source: Google Search Engine

Fully Connected Layer

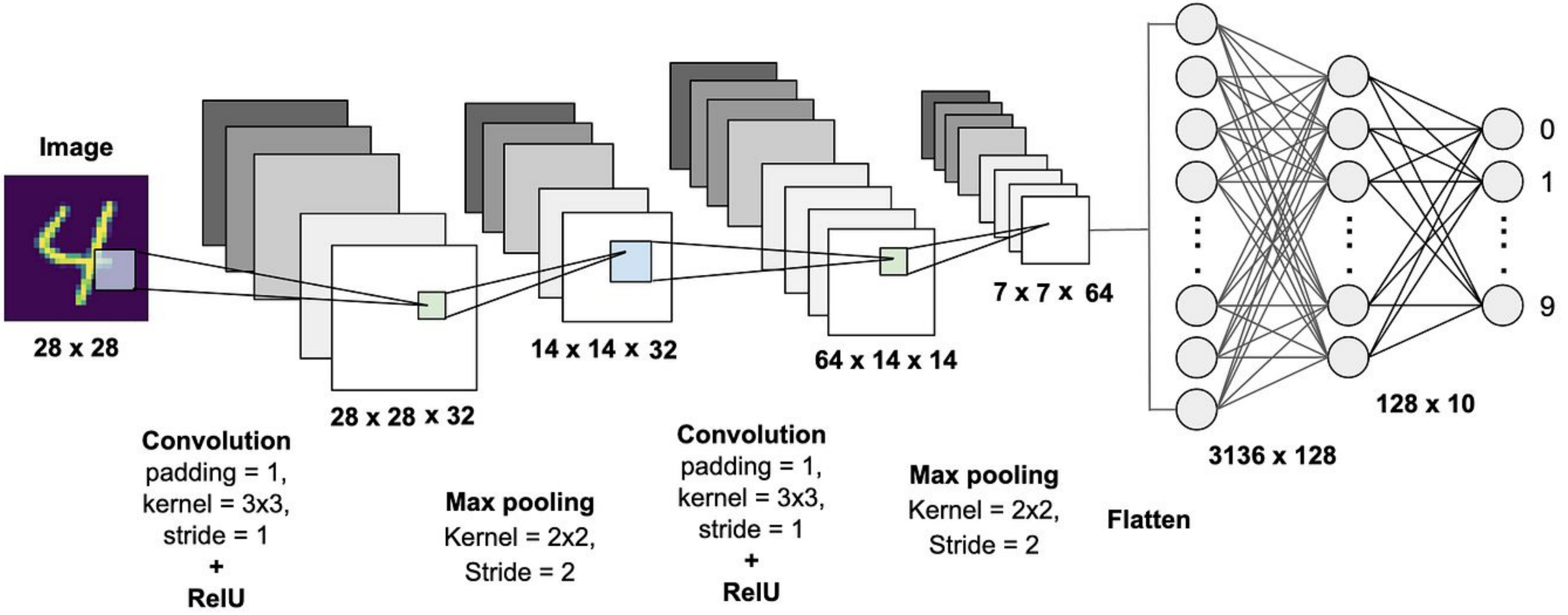
A fully connected layer connects every neuron in one layer to every neuron in its previous layer.

An image need to be turned into a vector before feeding into an fully connected layer.

Flatten() is used in Tensorflow.keras

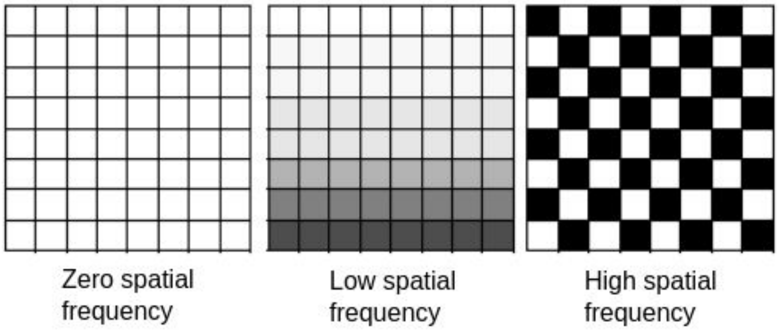


CNN Classifier



Convolution in Image Processing

- In image processing,
  - Convolution is used to modify the spatial frequency characteristics of an image.



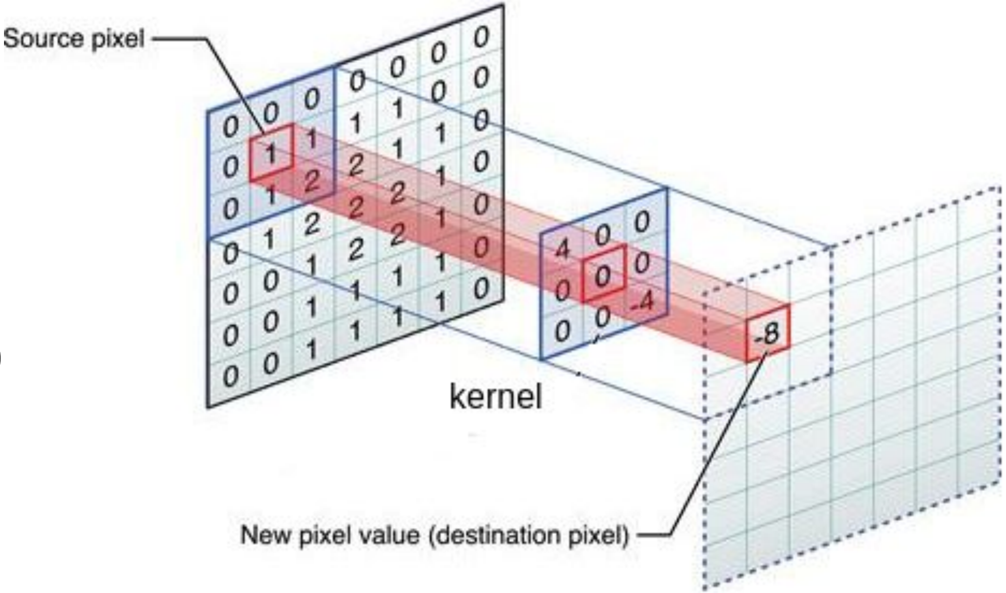
  - It is a filter effect on images
  - It is the process of adding each element of the image to its local neighbors, weighted by the kernel/filter.

Convolution for Single-Channel Input & Single-Channel Output

Convolution layer adds each element of a feature map to its local neighbors, weighted by the kernel.

Source pixel: 1

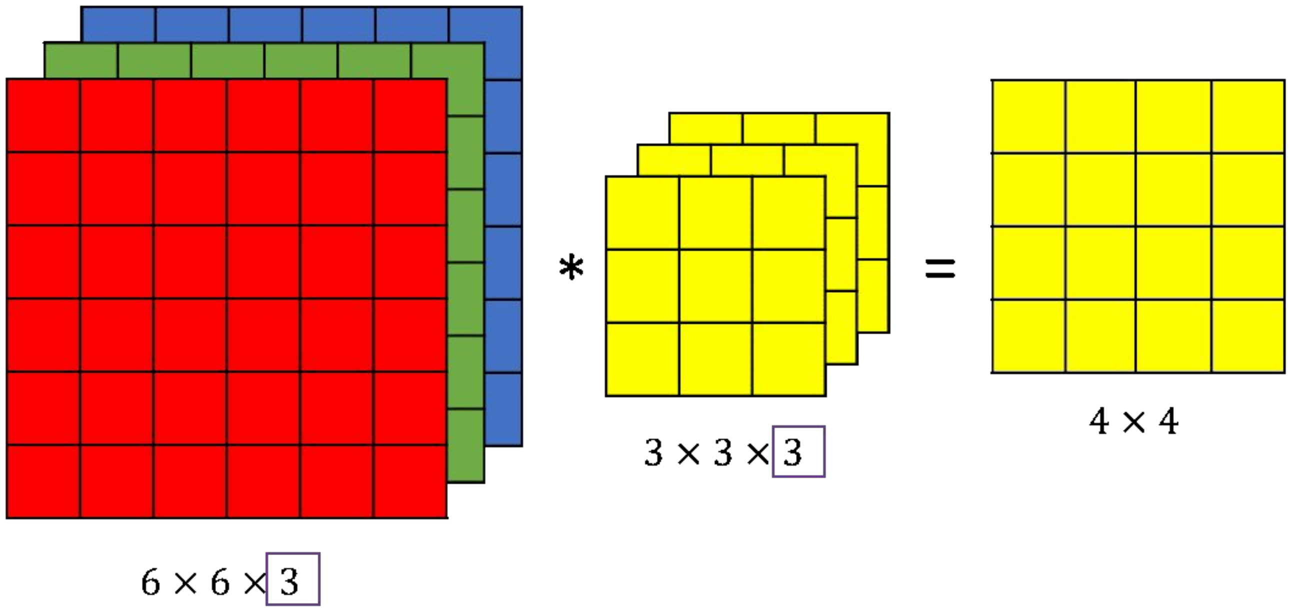
Destination pixel: 4x0 + 0x0 + 0x0 + 0x0 + 1x0 + 2x0 + 0x0 + 1x0 + (-4)x2 = -8



Total number of multiplication for one pixel = kernel_height x kernel_width

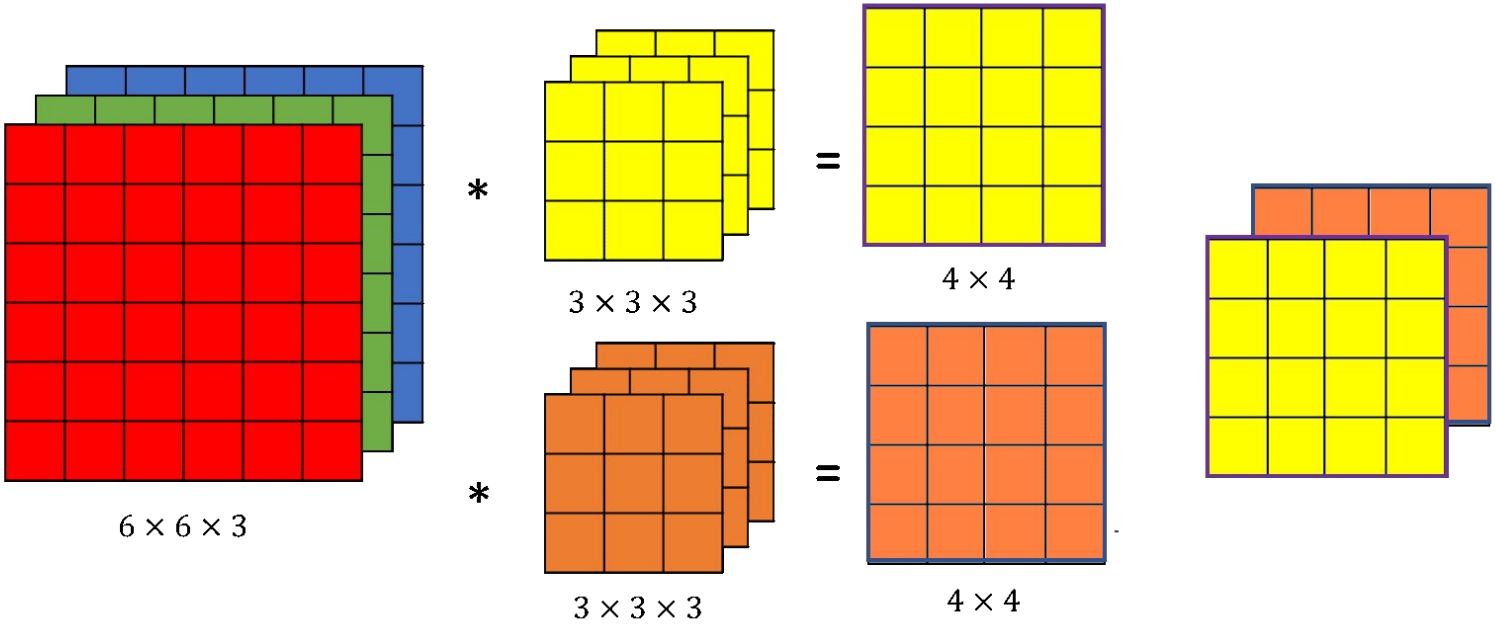
Convolution for Multi-Channels Input & Single-Channel Output

- Number of Kernel's Channels = Number of Input Feature Map's Channels



Convolution for Multi-Channels Input & Multi-Channels Output

- Number of Kernels = Number of Output Feature Map’s Channels



Padding

Padding preserves the spatial size of the input so that the output after convolution remains the same size or desired size.