Trustworthy Aritificial Intelligence: Neural Networks

Presented by

Dr. Narinder Singh Punn,

Assistant Professor,

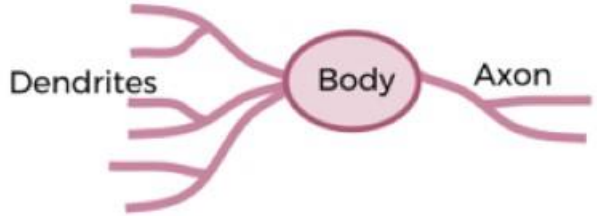
Dept. of CSE,

ABV-IIITM Gwalior

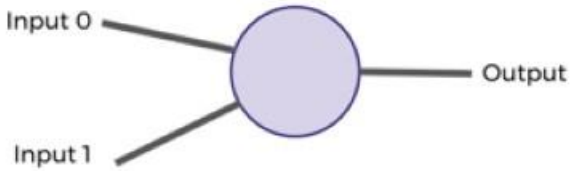
विश्वजीवनामृतं ज्ञानम्

Perceptron

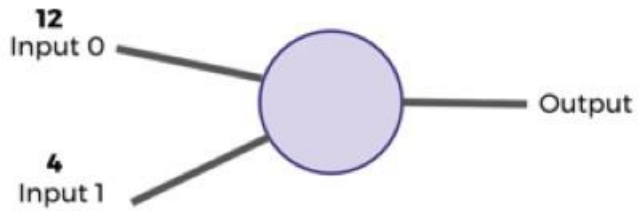
- Biological Neuron
  - Electrical signal gets passed through the dendrites to the body of the cell and then a single output or a single electrical signal is passed via axon to other neurons.

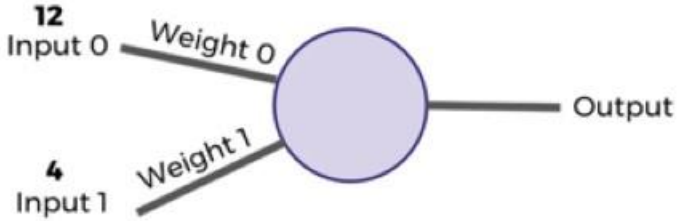


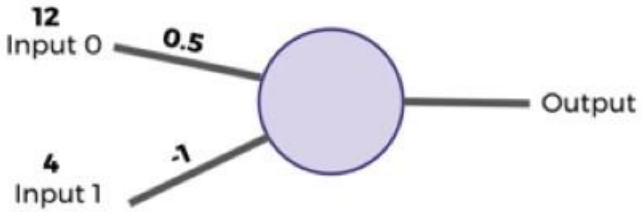
- We attempt to mimic this behavior into the artificial neuron (perceptron) having inputs and outputs.

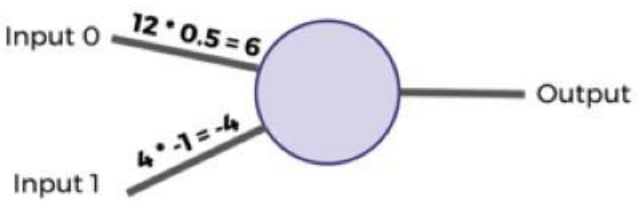


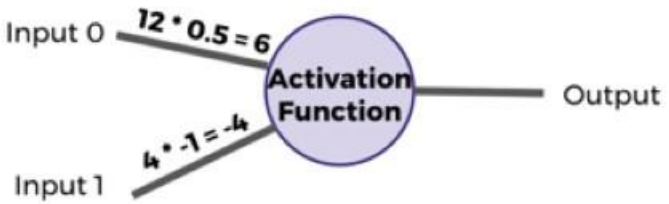
Perceptron

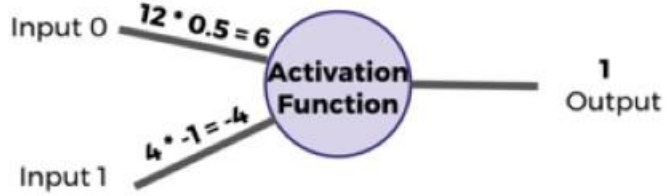

12
Input 0
4
Input 1
Output

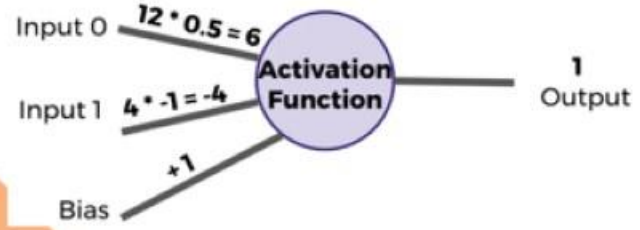

12
Input 0
Weight 0
4
Input 1
Weight 1
Output


12
Input 0
0.5
4
Input 1
-1
Output


Input 0
12 * 0.5 = 6
Input 1
4 * -1 = -4
Output

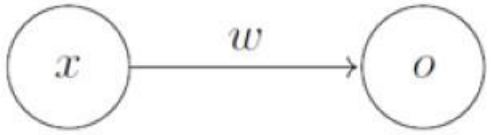

Input 0
12 * 0.5 = 6
Activation Function
Input 1
4 * -1 = -4
Output


Input 0
12 * 0.5 = 6
Activation Function
Input 1
4 * -1 = -4
1
Output

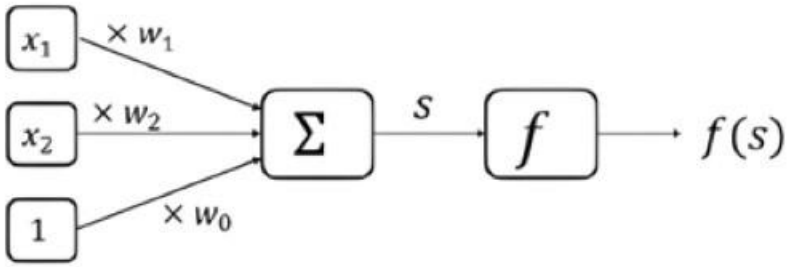

Input 0
12 * 0.5 = 6
Activation Function
Input 1
4 * -1 = -4
Bias
+1
1
Output

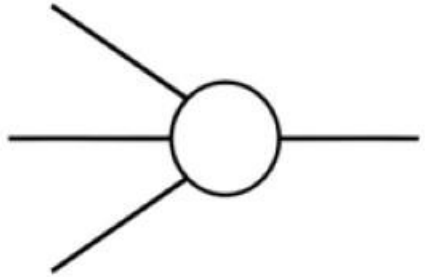
Neural Networks

- In its simplest form, a neural network has one input node and one output node



Perceptron model





Compare this model to logistic regression

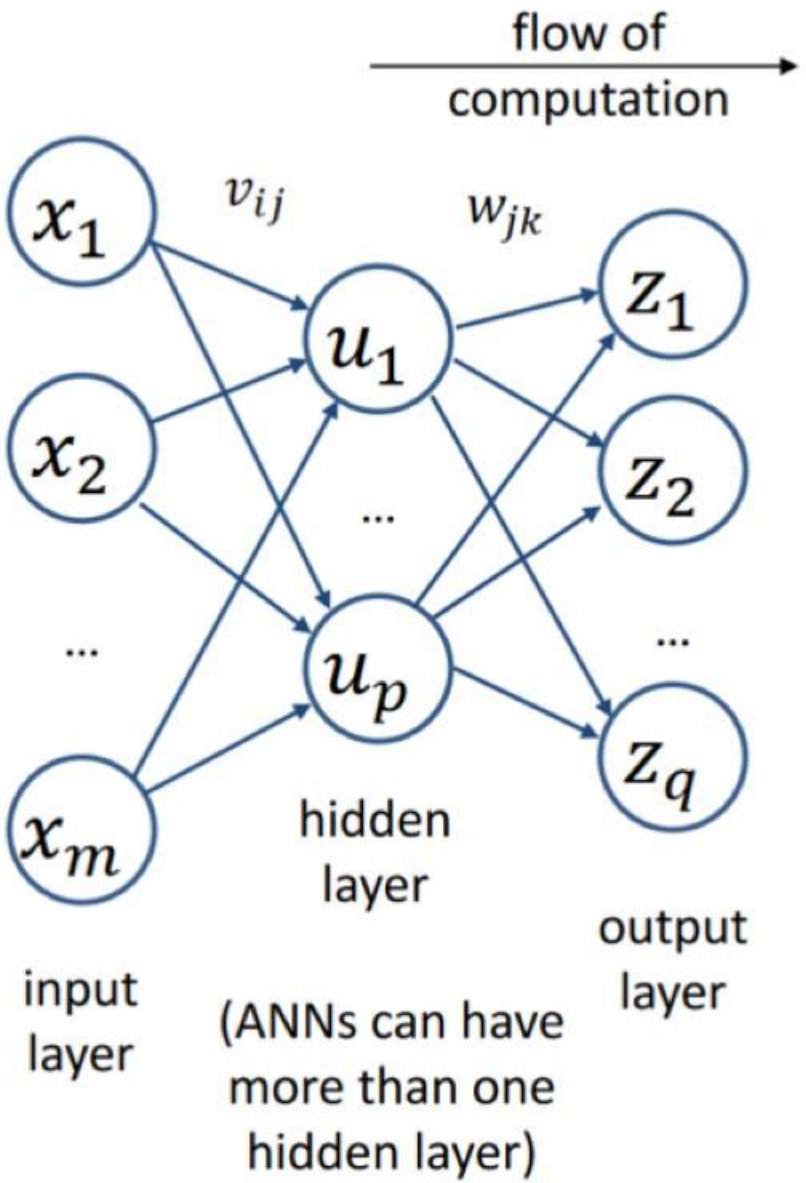
- x_1, x_2 – inputs
- w_1, w_2 – synaptic weights
- w_0 – bias weight
- f – activation function

Artificial Neural Network

x_i are inputs, i.e., attributes

note: here x_i are components of a single training instance $\boldsymbol{x}$

a training dataset is a set of instances



z_i are outputs, i.e., predicted labels

note: ANNs naturally handle multidimensional output

e.g., for handwritten digits recognition, each output node can represent the probability of a digit

Number of Parameters

- Weights from Input to Hidden Layer:

There are m input nodes and p hidden nodes.

Each of the m inputs connects to each of the p hidden nodes, resulting in $m \times p$ weights.

Additionally, each hidden node u_j has a bias, so there are p biases.

Total parameters for this layer: $m \times p + p$.

Number of Parameters

- Weights from Hidden to Output Layer:

There are p hidden nodes and q output nodes.

Each of the p hidden nodes connects to each of the q output nodes, resulting in $p \times q$ weights.

Additionally, each output node z_k has a bias, so there are q biases.

Total parameters for this layer: $p \times q + q$.

Number of Parameters

- Total Parameters:

Combine the parameters from both layers:

$$\text{Total parameters} = (m \times p + p) + (p \times q + q)$$

This can be factored as:

$$\text{Total parameters} = (m \times p + p \times q) + (p + q)$$

Alternatively, grouping the bias terms:

$$\text{Total parameters} = (m \times p) + (p \times q) + p + q$$

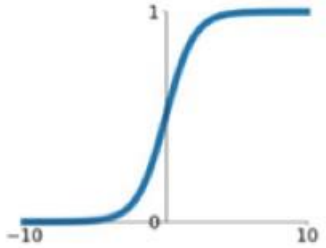
ANN in Supervised Learning

- ANNs can be naturally adapted to various supervised learning setup
  - Univariate regression
  - Multivariate regression
  - Binary classification
  - Multivariate classification

Activation Functions

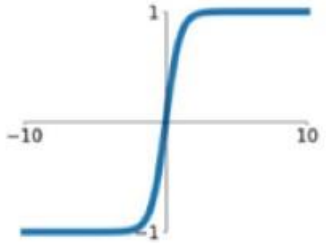
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



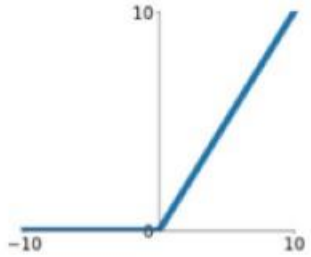
tanh

$$\tanh(x)$$



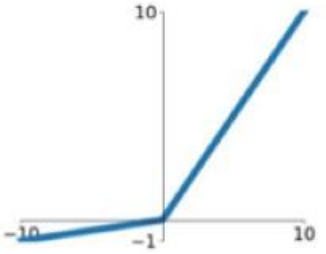
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

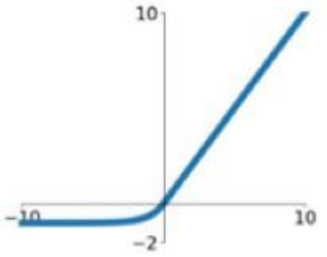


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Stochastic Gradient Descent for ANN

Choose initial guess $\boldsymbol{\theta}^{(0)}$, $k = 0$

Here $\boldsymbol{\theta}$ is a set of all weights form all layers

For i from 1 to T (epochs)

For j from 1 to N (training examples)

Consider example $\{\boldsymbol{x}_j, y_j\}$

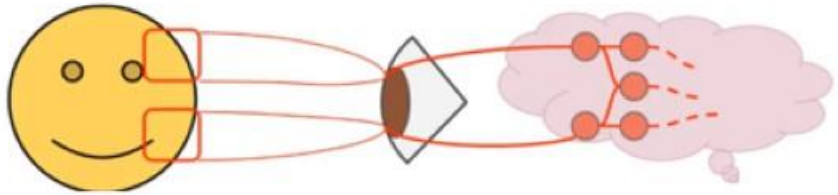
Update: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \boldsymbol{\nabla} L(\boldsymbol{\theta}^{(i)})$

$$L = \frac{1}{2}(z_j - y_j)^2$$

Need to compute partial derivatives $\frac{\partial L}{\partial v_{ij}}$ and $\frac{\partial L}{\partial w_j}$

CNNs

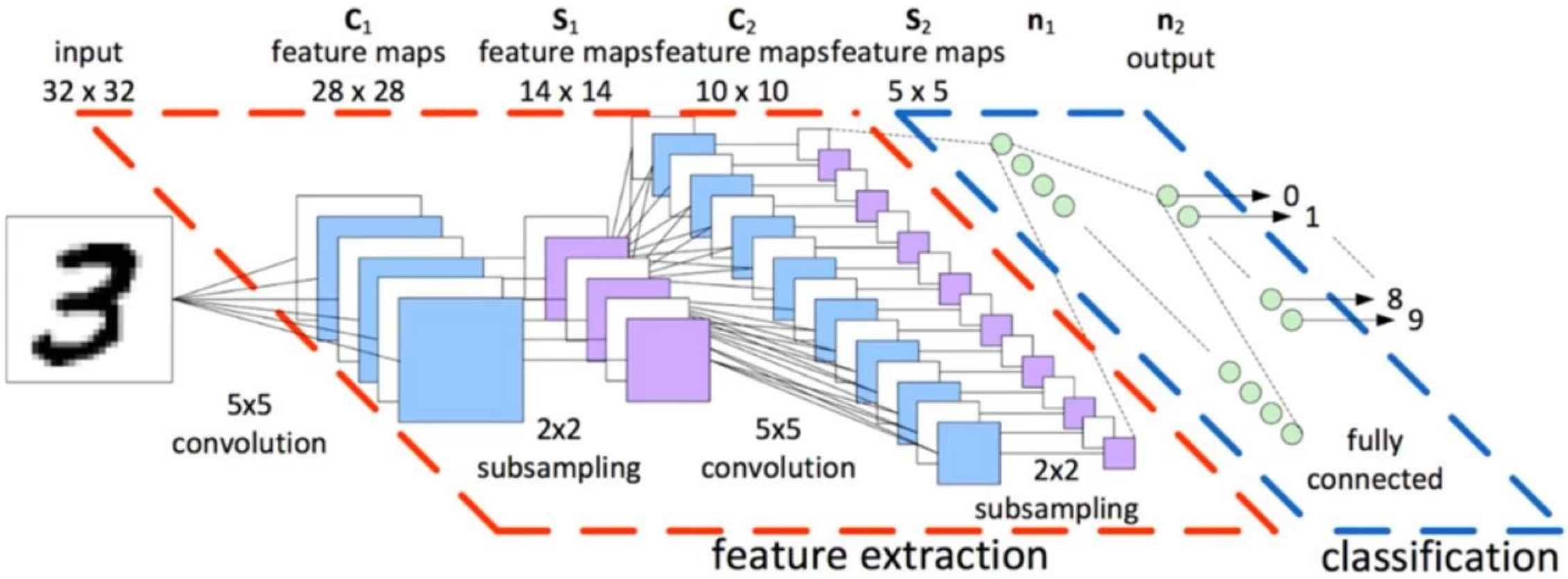
- Just like simple perceptron model. They too have their origin in biological research.
- Neurons in the visual cortex had a small local receptive field.
- These neurons are actually only looking at a local a smaller subsection of the entire image that the person is viewing and then these local subsections can then later on overlap and create a larger image and visual field.



- This idea of these local subsets directly inspired the artificial neural network (ANN) architecture that would become the convolutional neural network (CNN).

CNNs

- Now let's breakdown the various aspects of a CNN here.



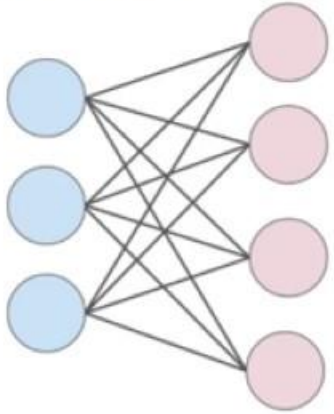
CNNs

- Input Layer
  - Holds raw pixel values of the image.
- Convolution layer
  - Extract the unique features from the input image.
- Pooling layer
  - Reduce the dimensionality (downsampling).
  - Max pooling, average pooling, etc.
- Fully Connected layer
  - Computes the class scores followed by the soft max regression to identify the class of an object.

FNN vs CNN

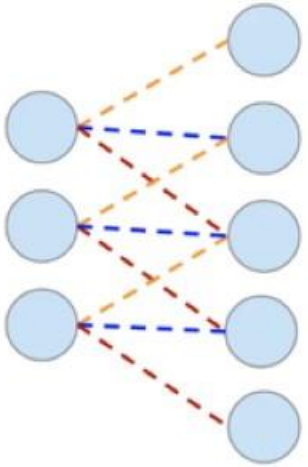
FNN

- In FNN every neuron in one layer is directly connected to every other neuron in the next layer.



CNN

- Each unit is connected to a smaller number of nearby units in the next layer inspired from the idea of the biological visual cortex.



Convolution Operation

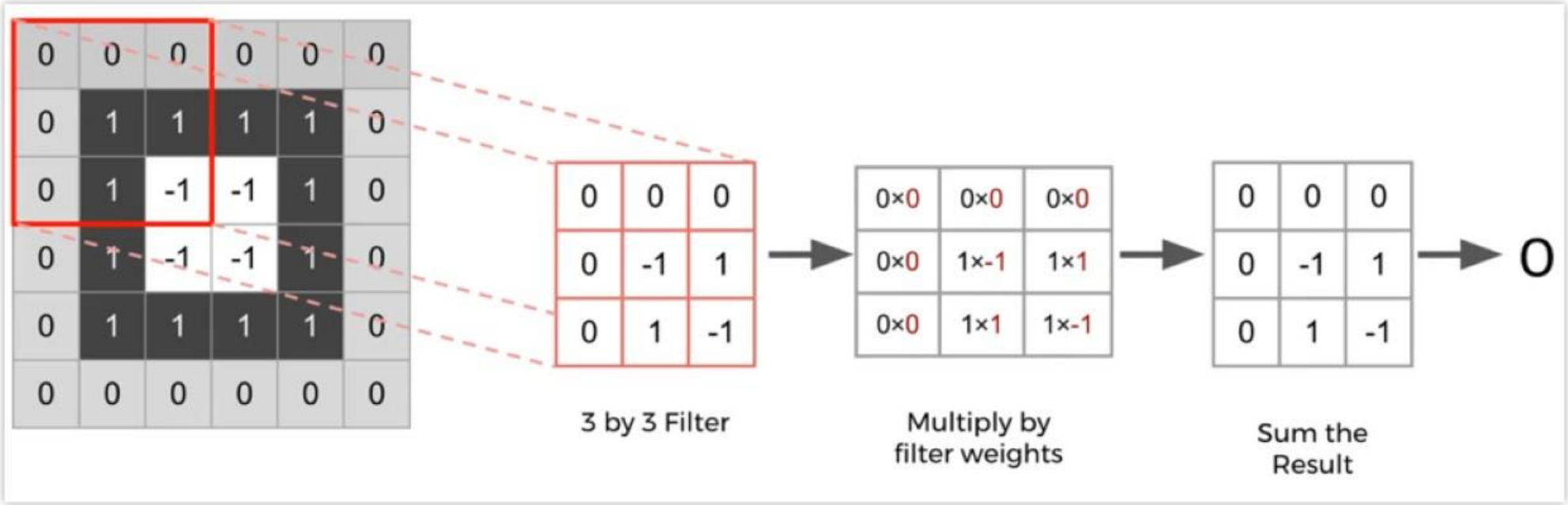
- Key Hyperparameters:
  - Kernel Size (Filter Size):
    - The dimensions of the convolutional filter (e.g., 3x3, 5x5).
    - Determines the receptive field of the convolution operation, affecting the amount of input data considered at each step.
  - Number of Filters (or Kernels):
    - The number of filters applied to the input (e.g., 32, 64).
    - Each filter learns a different feature (e.g., edges, textures), and the output has this many feature maps.

Convolution Operation

- Key Hyperparameters:
  - Stride:
    - The step size with which the kernel slides over the input (e.g., 1, 2).
    - A larger stride reduces the spatial dimensions of the output, while a smaller stride (e.g., 1) preserves more detail.
  - Padding:
    - The number of zeros added around the input borders (e.g., 'valid' for no padding, 'same' to preserve input dimensions).
    - Controls the output size and helps avoid losing information at the edges.

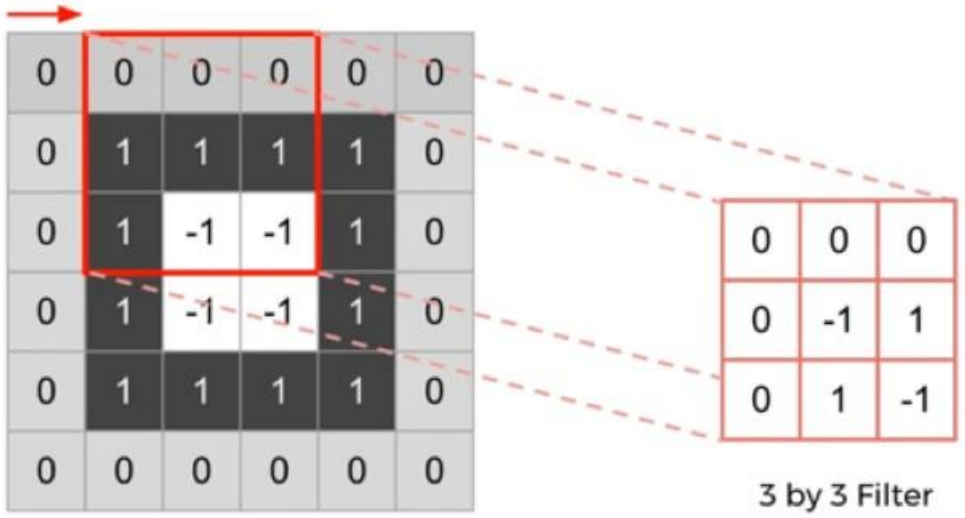
Convolution Operation

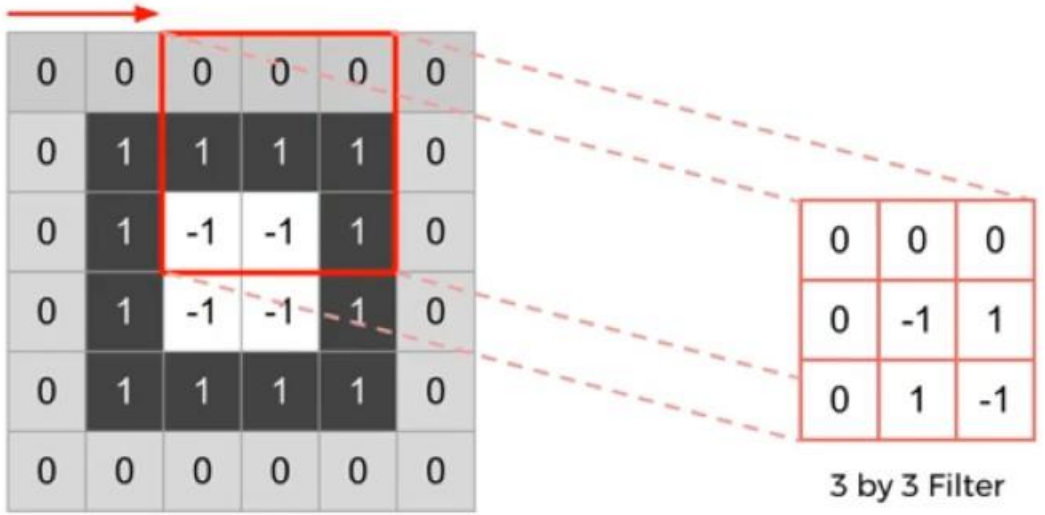
- Example:



Convolution Operation

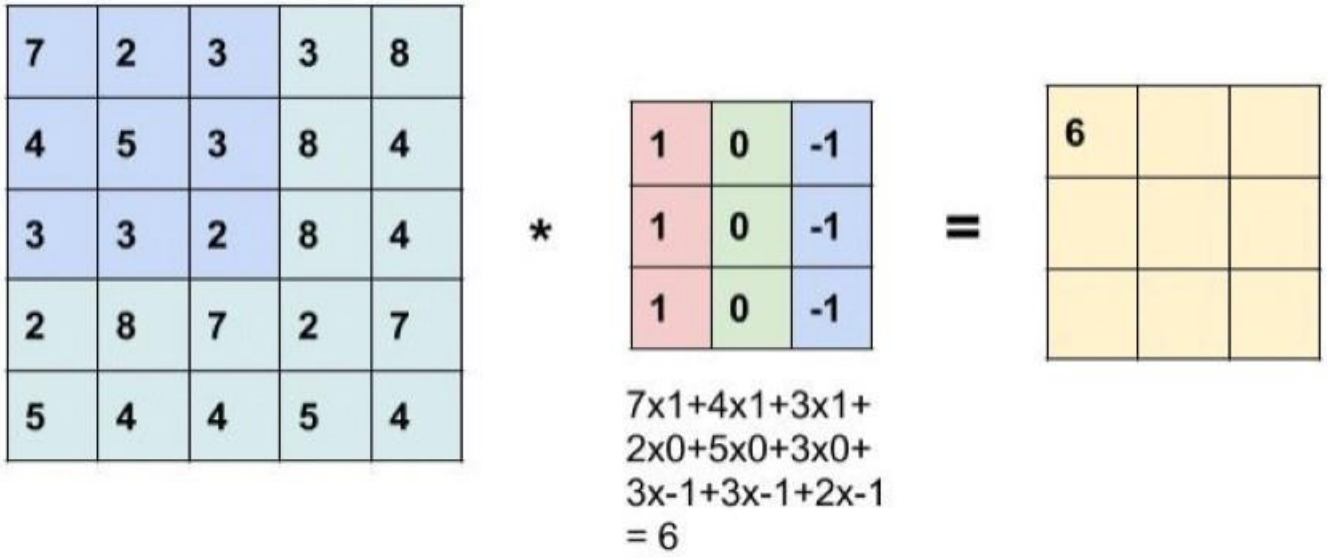
- Example:





Convolution Operation

- Example:



Convolution Operation

- Representation:

Let I be the input image (a matrix of pixel values) and K be the kernel (a smaller matrix of weights). The output of the convolution, a feature map denoted by O, is a matrix where each element $O(i,j)$ is calculated by:

$$O(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) \cdot K(m,n)$$

- In Practice:

$$O(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

Convolution Operation

- Output dimension:

For a 2D convolution:

- **Input dimensions**: $H_{in} \times W_{in}$ (height × width).
- **Kernel size**: $K_h \times K_w$ (height × width).
- **Stride**: $S_h \times S_w$ (vertical × horizontal stride).
- **Padding**: $P_h \times P_w$ (vertical × horizontal padding, e.g., 0 for no padding, "same" or "valid" as options).

The output dimensions $H_{out} \times W_{out}$ are given by:

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \cdot P_h - K_h}{S_h} \right\rfloor + 1$$

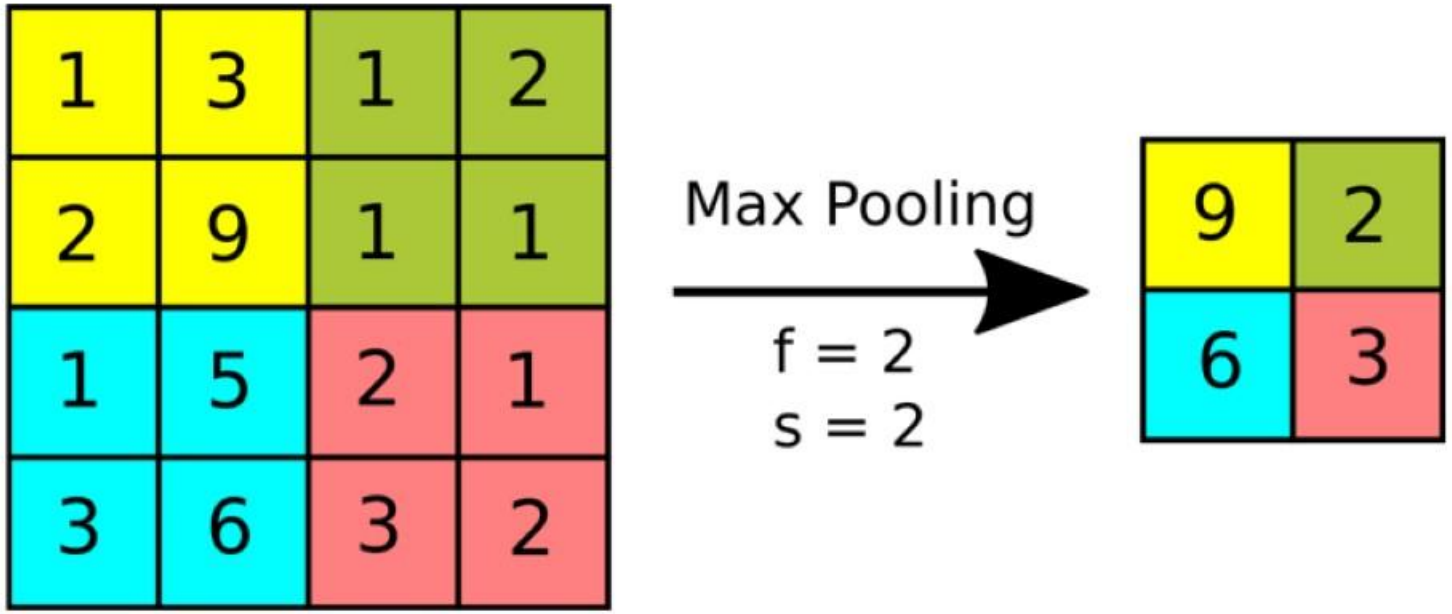
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \cdot P_w - K_w}{S_w} \right\rfloor + 1$$

Pooling Operation

- Used to reduce the spatial dimensions (height and width) of the input feature maps while retaining important information.
- It acts as a form of downsampling.

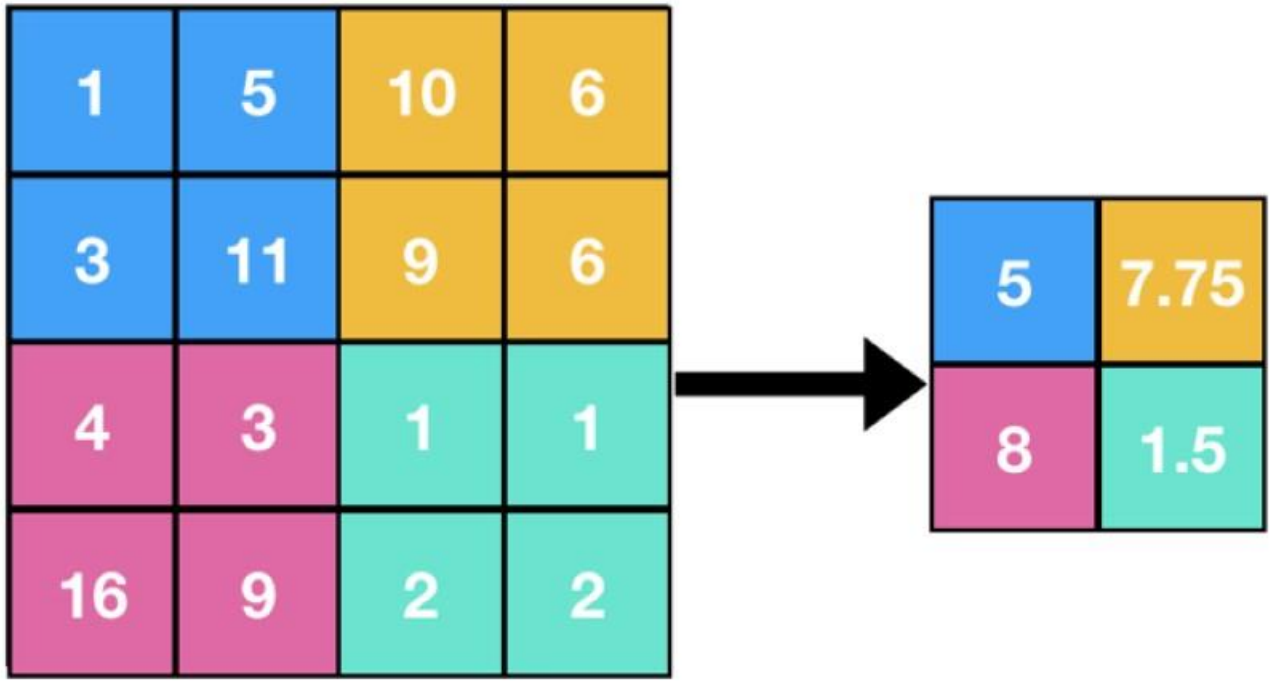
Types of Pooling

- Max Pooling:
  - Takes the maximum value from each region of the feature map covered by the pooling filter.
  - Commonly used because it emphasizes the most prominent features (e.g., edges or textures).



Types of Pooling

- Average Pooling:
  - Computes the average value within each region.
  - Smooths the feature map and is less sensitive to noise but may dilute strong features.



Types of Pooling

- Other Variants:
  - Min Pooling: Takes the minimum value (less common).
  - Global Pooling: Reduces the entire feature map to a single value (e.g., global max or average pooling), often used before the final classification layer.

Pooling Dimensions

For an input feature map of size $H_{in} \times W_{in}$, with a pooling filter of size $F_h \times F_w$ and stride $S_h \times S_w$:

$$H_{out} = \left\lfloor \frac{H_{in} - F_h}{S_h} \right\rfloor + 1$$

$$W_{out} = \left\lfloor \frac{W_{in} - F_w}{S_w} \right\rfloor + 1$$

- Padding is typically 0 (no padding) in pooling layers.

Softmax Regression

- Softmax regression, also known as multinomial logistic regression, is a supervised learning algorithm used for multi-class classification problems.

- It generalizes logistic regression to handle more than two classes by predicting the probabilities of each class, ensuring they sum to 1.

Softmax Regression

For a feature vector $x \in \mathbb{R}^n$, softmax regression computes a score for each class k using weights $w_k \in \mathbb{R}^n$ and bias b_k:

$$z_k = w_k^T x + b_k$$

The probability of class k is given by the softmax function:

$$p(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log(p(y_i = k|x_i))$$

Softmax Regression

- The Iris dataset contains 150 samples of iris flowers, each with 4 features (sepal length, sepal width, petal length, petal width) and 3 classes (Setosa, Versicolor, Virginica).
- Softmax regression can classify a flower into one of these classes.

Suppose we have a sample with features $x = [5.1, 3.5, 1.4, 0.2]$. Assume trained weights and biases for the three classes are:

- Setosa: $w_1 = [0.2, 0.1, -0.5, -0.4], b_1 = 0.3$
- Versicolor: $w_2 = [-0.1, 0.2, 0.3, 0.1], b_2 = -0.2$
- Virginica: $w_3 = [0.1, -0.3, 0.4, 0.5], b_3 = 0.1$

- Find the class for new sample x.

Softmax Regression

1. **Compute scores**:

$$z_1 = (0.2 \cdot 5.1) + (0.1 \cdot 3.5) + (-0.5 \cdot 1.4) + (-0.4 \cdot 0.2) + 0.3 = 1.02 + 0.35 - 0.7 - 0.08 + 0.3 = 0.87$$

$$z_2 = (-0.1 \cdot 5.1) + (0.2 \cdot 3.5) + (0.3 \cdot 1.4) + (0.1 \cdot 0.2) - 0.2 = -0.51 + 0.7 + 0.42 + 0.02 - 0.2 = 0.43$$

$$z_3 = (0.1 \cdot 5.1) + (-0.3 \cdot 3.5) + (0.4 \cdot 1.4) + (0.5 \cdot 0.2) + 0.1 = 0.51 - 1.05 + 0.56 + 0.1 + 0.1 = 0.22$$

2. **Apply softmax**:

$$e^{z_1} = e^{0.87} \approx 2.39, \quad e^{z_2} = e^{0.43} \approx 1.54, \quad e^{z_3} = e^{0.22} \approx 1.25$$

Sum of exponentials:

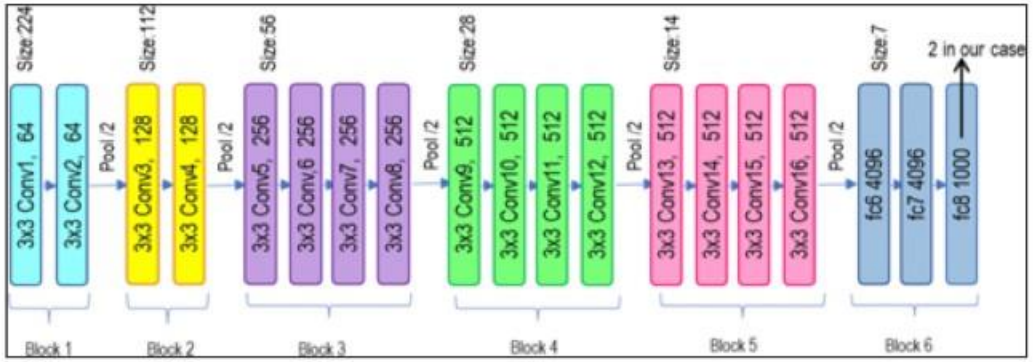
$$2.39 + 1.54 + 1.25 = 5.18$$

Probabilities:

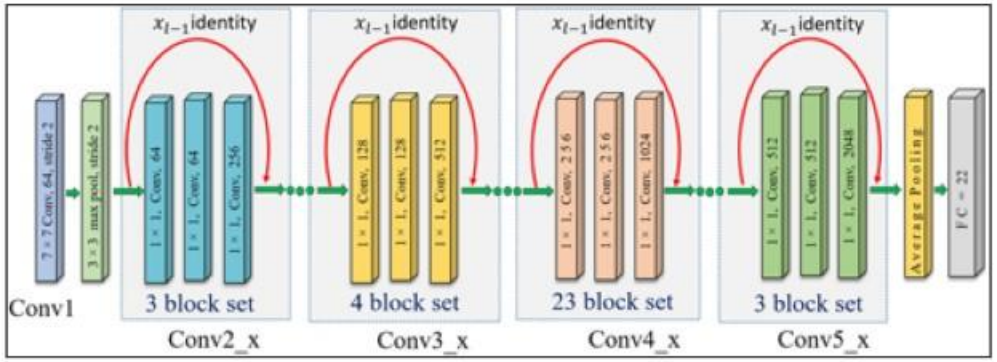
$$p(\text{Setosa}) = \frac{2.39}{5.18} \approx 0.46, \quad p(\text{Versicolor}) = \frac{1.54}{5.18} \approx 0.30, \quad p(\text{Virginica}) = \frac{1.25}{5.18} \approx 0.24$$

3. **Prediction**: The model predicts the class with the highest probability, Setosa (0.46).

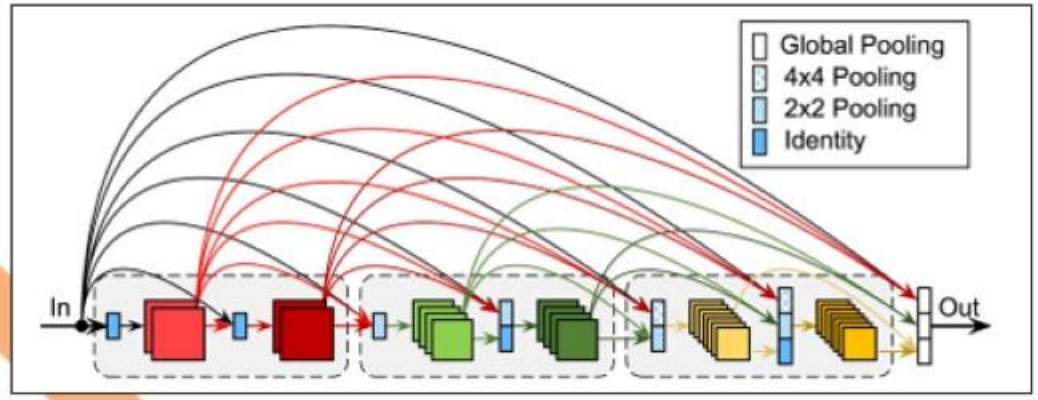
Popular CNN Models



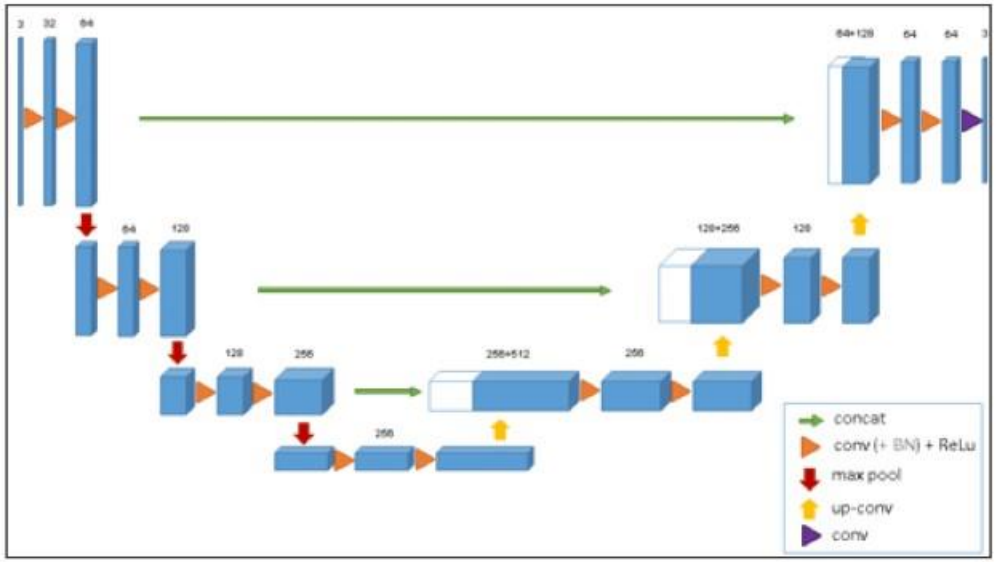
VGG-19



ResNet



DenseNet



U-Net

THANK YOU