UNIT 2
CONVOLUTIONAL NEURAL NETWORK

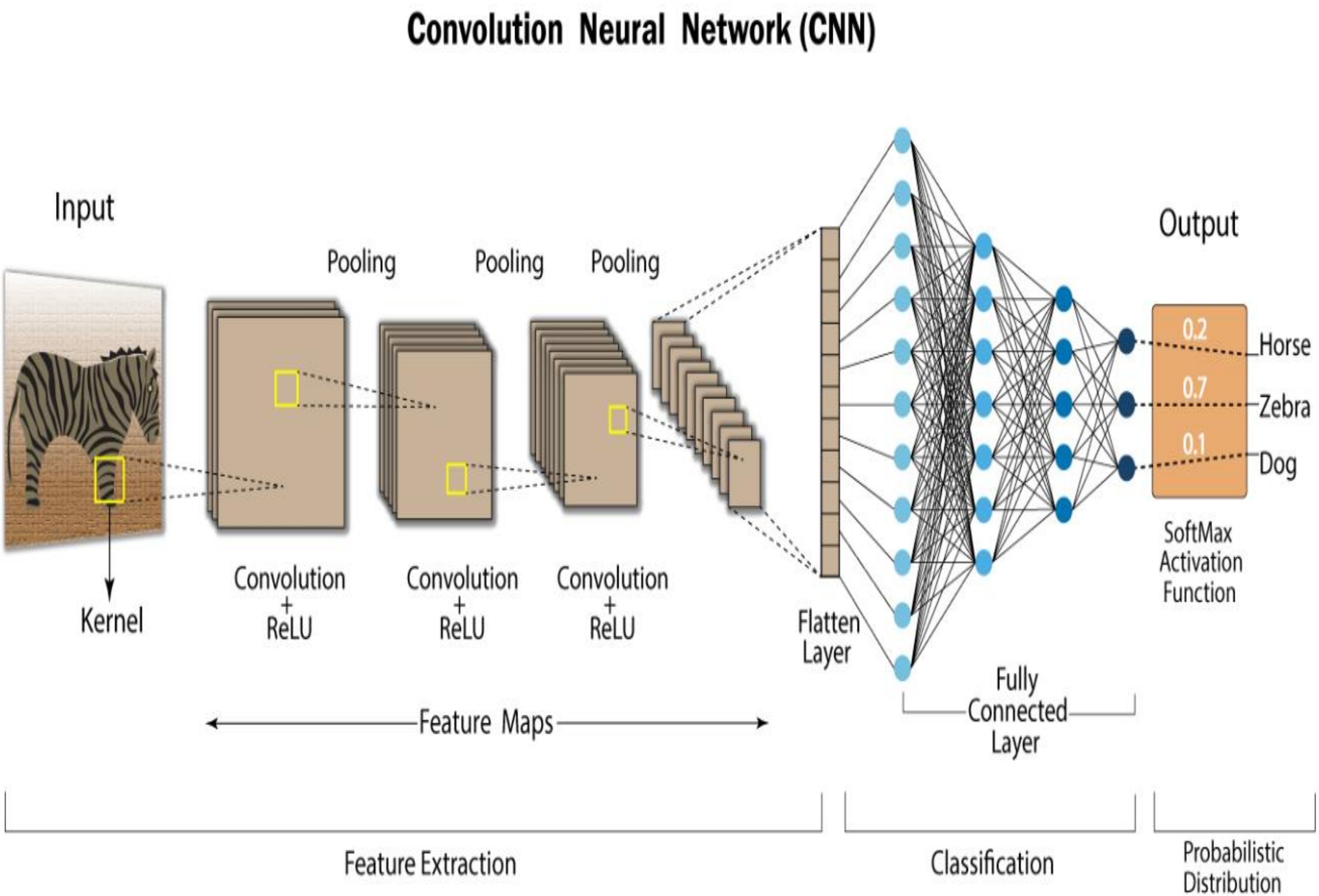
Introduction

- A convolutional neural network (CNN) -class of artificial neural network applied to analyze visual imagery.
- CNNs use a mathematical operation, **convolution** in place of general matrix multiplication at least one of their layers.
- CNNs specifically designed to process pixel data and used in image recognition and processing.
- A convolutional neural network is a feed-forward neural network, often with up to 20 or 30 layers.

Introduction

- CNN,s contain many convolutional layers stacked on top of each other.
- Each one capable of recognizing more sophisticated shapes.
- Three or four convolutional layers possible to recognize handwritten digits.
- With 25 layers it is possible to distinguish human faces.

- ***CNN Applications:***
-
- Image & Video Recognition.
- Recommender Systems.
- Image Classification, Image Segmentation.
- Medical Image Analysis.
- Natural Language Processing.
- Brain–Computer Interfaces.
- Financial Time Series.


Convolution Neural Network (CNN)
Input
Kernel
Pooling
Pooling
Pooling
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Feature Maps
Flatten Layer
Fully Connected Layer
Output
0.2
Horse
0.7
Zebra
0.1
Dog
SoftMax Activation Function
Feature Extraction
Classification
Probabilistic Distribution

Convolution Operation Works

- **Input Data:**
  - The input data can be a single-channel image (grayscale) or multi-channel image (color), where each channel corresponds to a different feature (e.g., Red, Green, and Blue channels in a color image).
- **Filters (Kernels):**
  - Filters are small matrices of learnable weights.
  - For example, a common size for a filter might be 3x3, 5x5, or 7x7.
  - Each filter is designed to detect a specific feature, such as edges, textures, or more complex patterns.

Convolution Operation Works

- The number of filters used in a convolutional layer determines the number of output channels, also known as feature maps.
- **Convolution Process:**
  - The convolution operation involves sliding (or convolving) the filter across the input data.
  - At each position, a dot product is computed between the filter's weights and the corresponding values in the input data.
  - This process produces a single output value for each position, resulting in a feature map.

Convolution Operation Works

- The mathematical expression for the convolution operation at a specific location (i,j) can be written as: $(F * I)(i,j) = \sum_{m=1}^{M} \sum_{n=1}^{N} F(m,n) \cdot I(i+m, j+n)$
- F is the filter matrix (of size M×N).
- I is the input data.
- (i,j)represents the top-left corner of the region of the input where the filter is applied.

Convolution Operation - Example

- As convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one function is modified by another.
- Suppose a tour operator and offering a tour that takes 3 days.
- Guests can start the tour on any day.
- On day 1 of their tour, people take two meals at their hotel, and operator need to provide them with one meal for the trip.
- On day 2, it is 2 meals.
- On day 3 they take a full day trip, so operator need to get them 3 meals.

- Let’s say you have 10 people start on day one, 8 people on day 2, and 5 people on day 3, 4 people on day 4.
- How to keep track of the number of meals need to prepare each day?

Day 1: 10 x 1 = 10

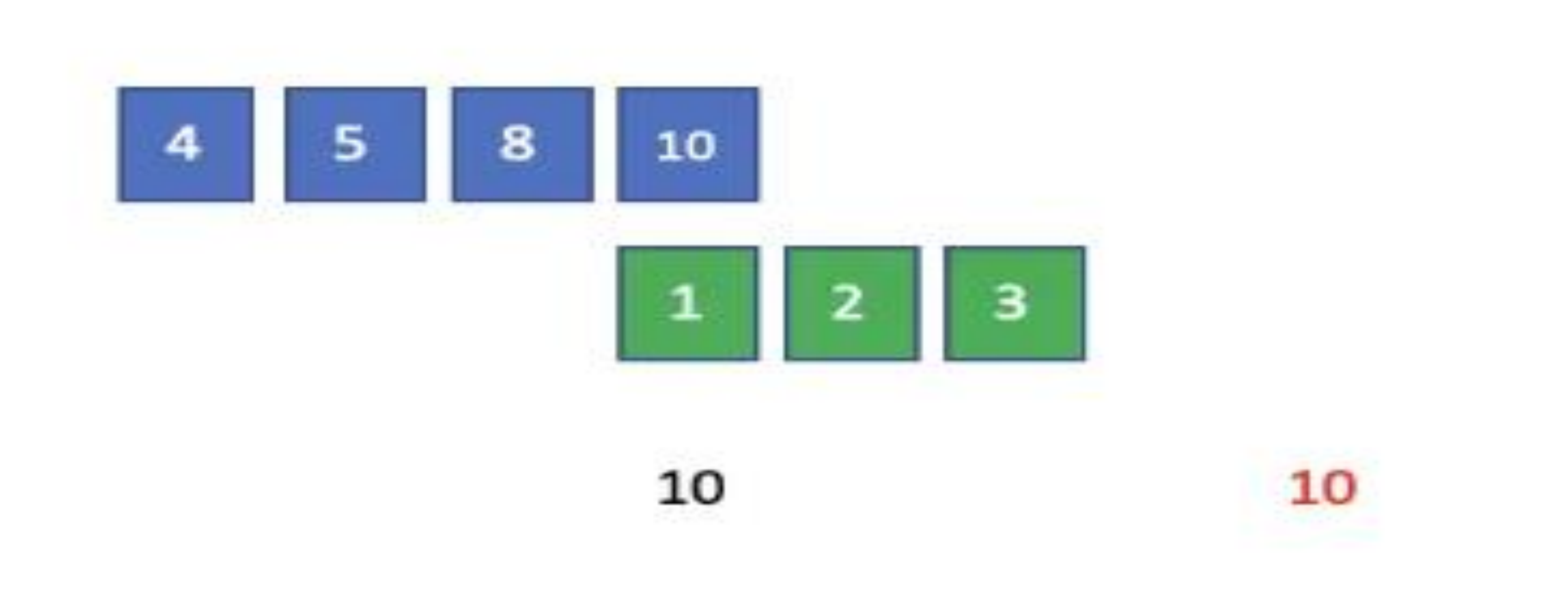
Day 2: 10x2 + 8x1 = 28

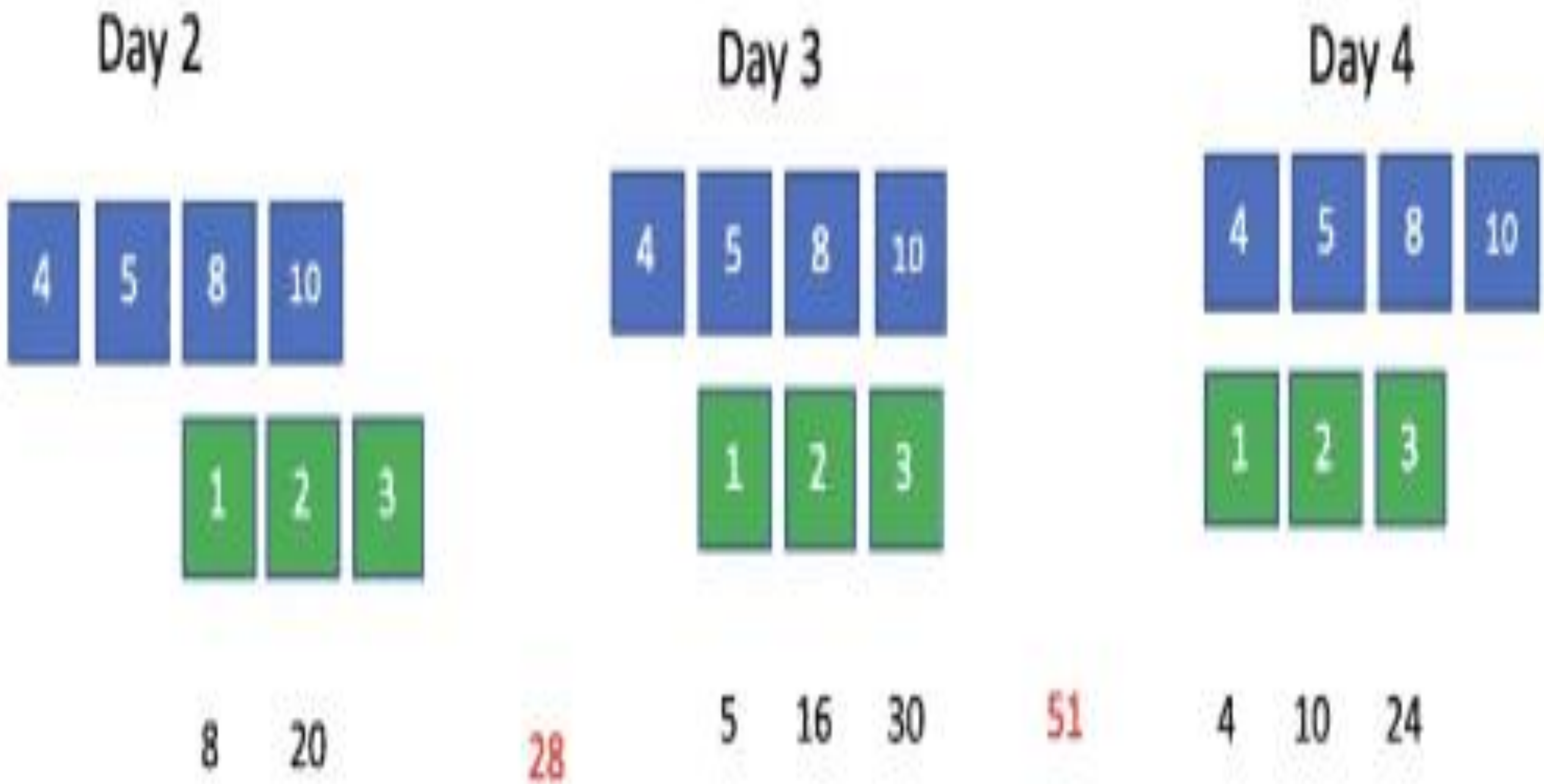
Day 3: 10x3 + 8x2 + 5x1 = 51

Day 4: 8x3 + 5x2 + 4x1 = 38

- **Calculate the Convolution**
- Represent the number of people arriving per day in a grid and the number of meals per day in another grid.

- **1D Convolution**
- Move the first position in the green grid over the last position in the blue grid.
- Multiply the blue entry by the green entry right below.





Day 2
4
5
8
10
1
2
3
8
20
28
Day 3
4
5
8
10
1
2
3
5
16
30
51
Day 4
4
5
8
10
1
2
3
4
10
24

- **Mathematical Definition of the Convolution**
- Need to translate the grids into mathematical functions.
- Tour plan has three days which denote as **d**.
- The function that calculates the number of meals for a given day **f**.
- Number of meals is a function f of d.
- Function is equivalent to the green grid or the kernel.

$$f(d)$$

- Define another function g that tracks the number of people participating in the tour that day.
- Function is equivalent to the blue grid.

$$g(d)$$

- To calculate the convolution, reversed the blue grid.
- Means need to define g as a function of negative d.

$$g(-d)$$

- Introduce variable **t**,
- A running count of the days ,how many people arrive each day.
- To find the correct number of people arriving that day?
- Input the current day **t** into **g** of **negative d**.

$$g(-d+t)$$

- Multiplying the two functions gives total number of meals for one day **d** of the tour.

$$\sum_{d}^{D} f(d)\, g(-d+t)$$

In mathematical notation, write the convolution of two functions or signals..

$f*g$

Parameter Sharing

- In CNNs, parameter sharing refers to the practice of using the same set of weights (filters or kernels) across different parts of the input data.
- This is a fundamental characteristic of the convolutional layers within CNNs.
- The primary purpose is to detect specific features, such as edges or textures, consistently across different regions of an input image or data.

Parameter Sharing Work

- **Convolutional Layer:**
  - A convolutional layer consists of several filters (also known as kernels).
  - Each filter is a small matrix, such as 3x3 or 5x5, with a set of learnable weights.
  - The filter slides (or convolves) over the input data, computing the dot product between the weights of the filter and the values of the input at each spatial location.
  - As the same filter is applied to all parts of the input, the parameters (weights) of the filter are shared across different spatial locations.

Parameter Sharing Work

- **Feature Maps:**
  - The output of the convolutional operation is a feature map, which highlights the presence of specific features detected by the filter in different regions of the input.
  - Since the same filter is used across the entire input, the feature map reflects the activation of that particular feature at every spatial location.

Benefits of Parameter Sharing

- **Reduced Number of Parameters:**
- Convolutional layers with shared weights significantly reduce the number of parameters, making the network more efficient and less prone to overfitting.
- For example, a 3x3 filter applied to a 256x256 image with 3 channels has only 27 parameters, regardless of the image size.

Benefits of Parameter Sharing

- **Translation Invariance:**
- Parameter sharing allows the network to detect features, such as edges or textures, regardless of their position in the image.
- This translation invariance is crucial for image recognition tasks, where the same object can appear in different parts of an image.
- **Efficient Training:**
- Fewer parameters lead to faster training and reduced computational requirements.

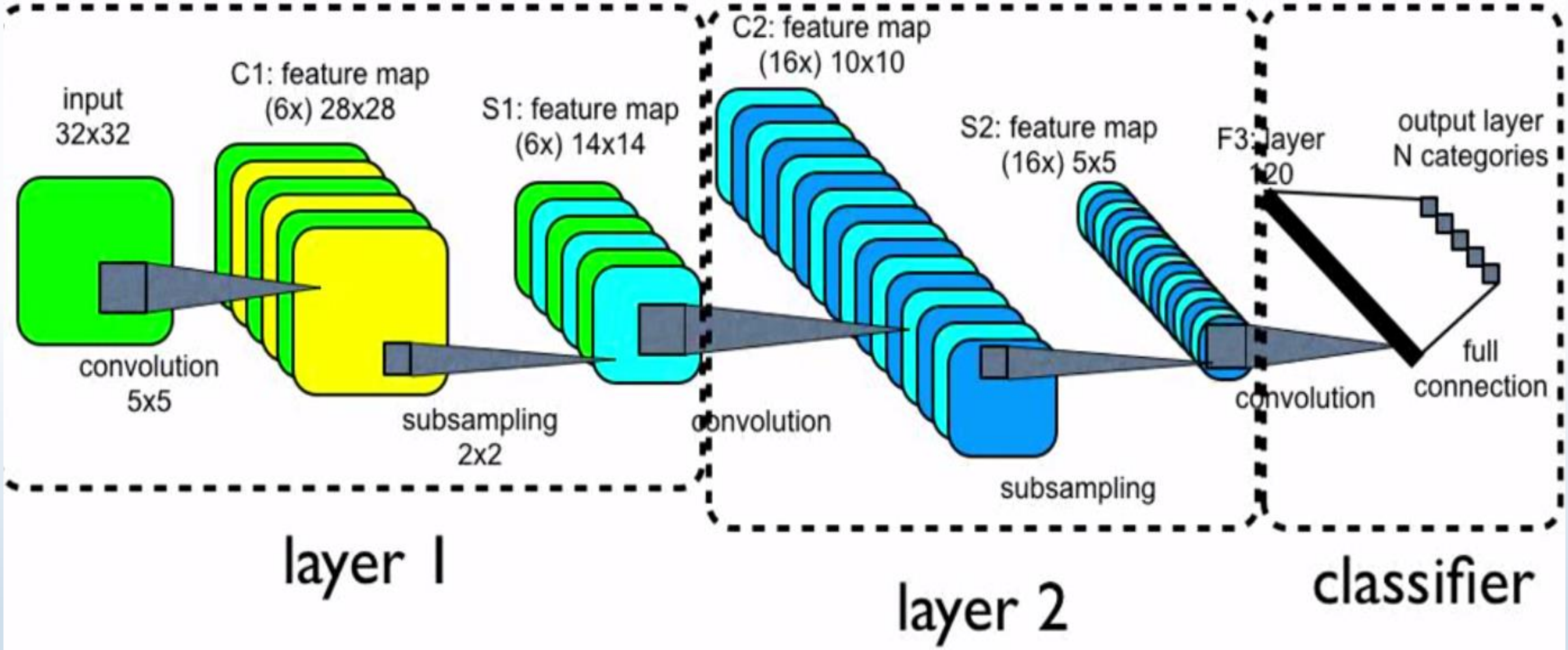
Benefits of Parameter Sharing

- **Improved Generalization:**
- By limiting the number of parameters, parameter sharing acts as a form of regularization.
- Helping prevent the model from overfitting to the training data.
- This improves the model's ability to generalize to new, unseen data.

Pooling

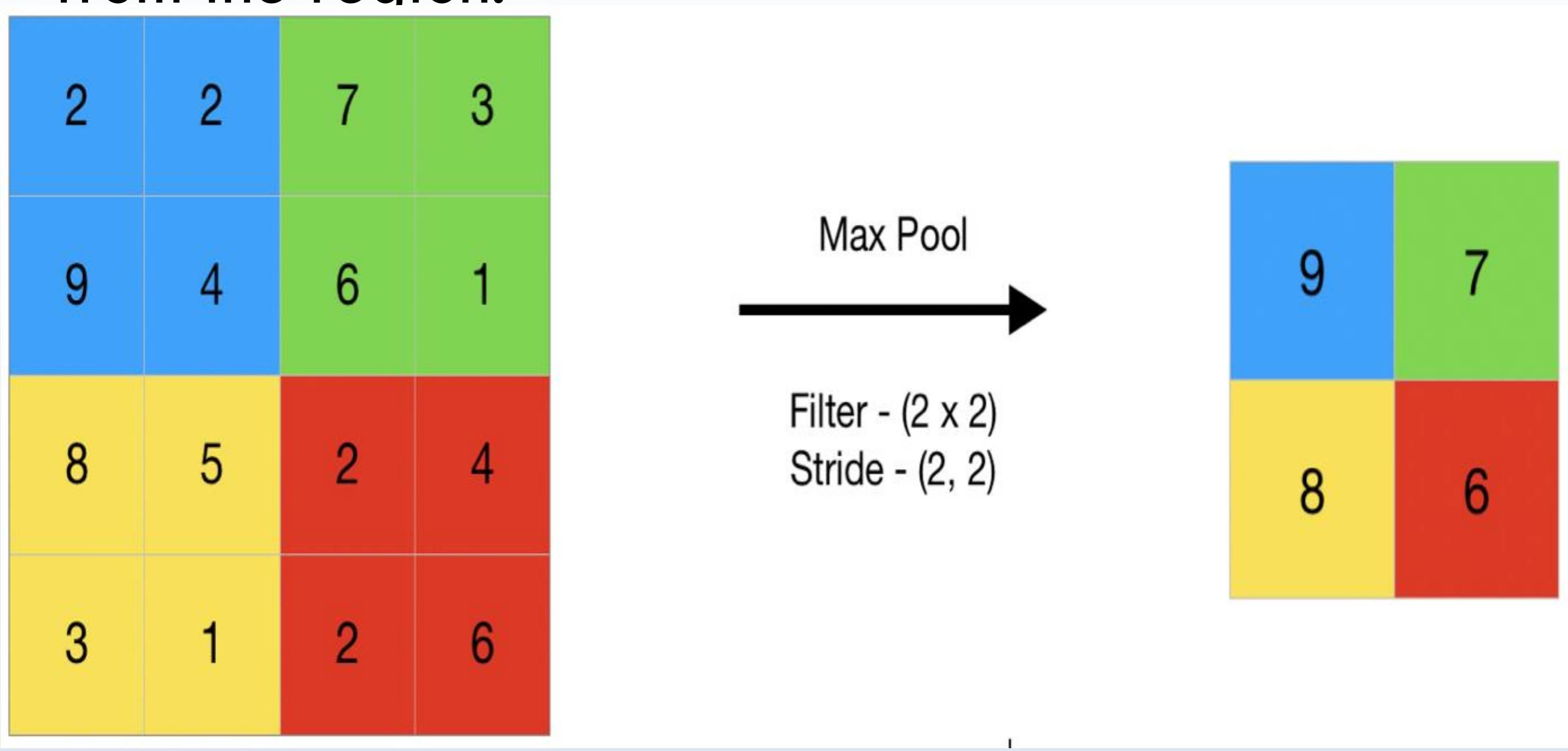
- Pooling layer is added after convolutional layers.
- Used to reduce the dimensions (width and height) of the feature maps.
- Preserving the depth (number of channels).
- Pooling operation involves :
- Sliding a two-dimensional filter over each channel of feature map and summarizing the features lying within the region covered by the filter.
- Feature map having dimensions $\mathbf{n_h \times n_w \times n_c}$

Convolutional Neural Networks

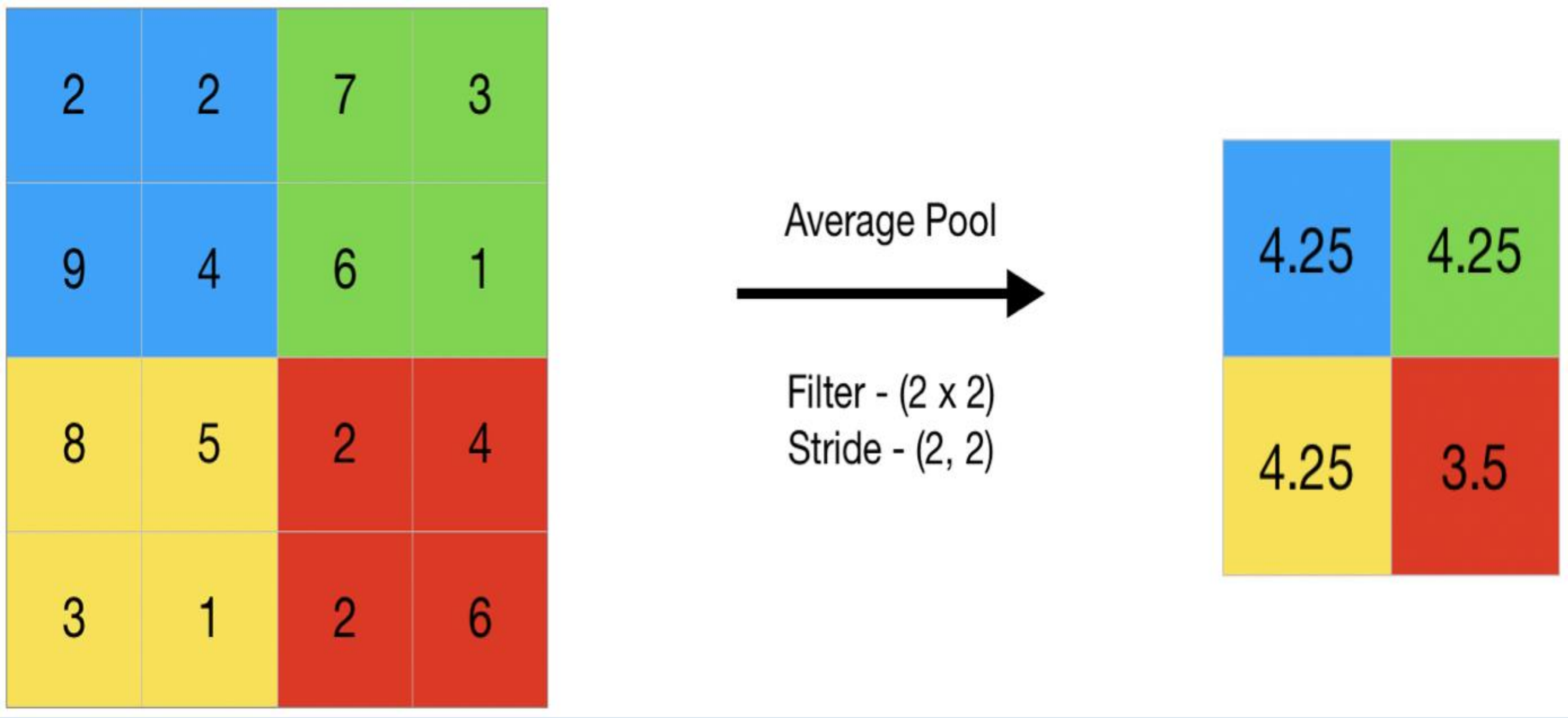


Types of Pooling Layers

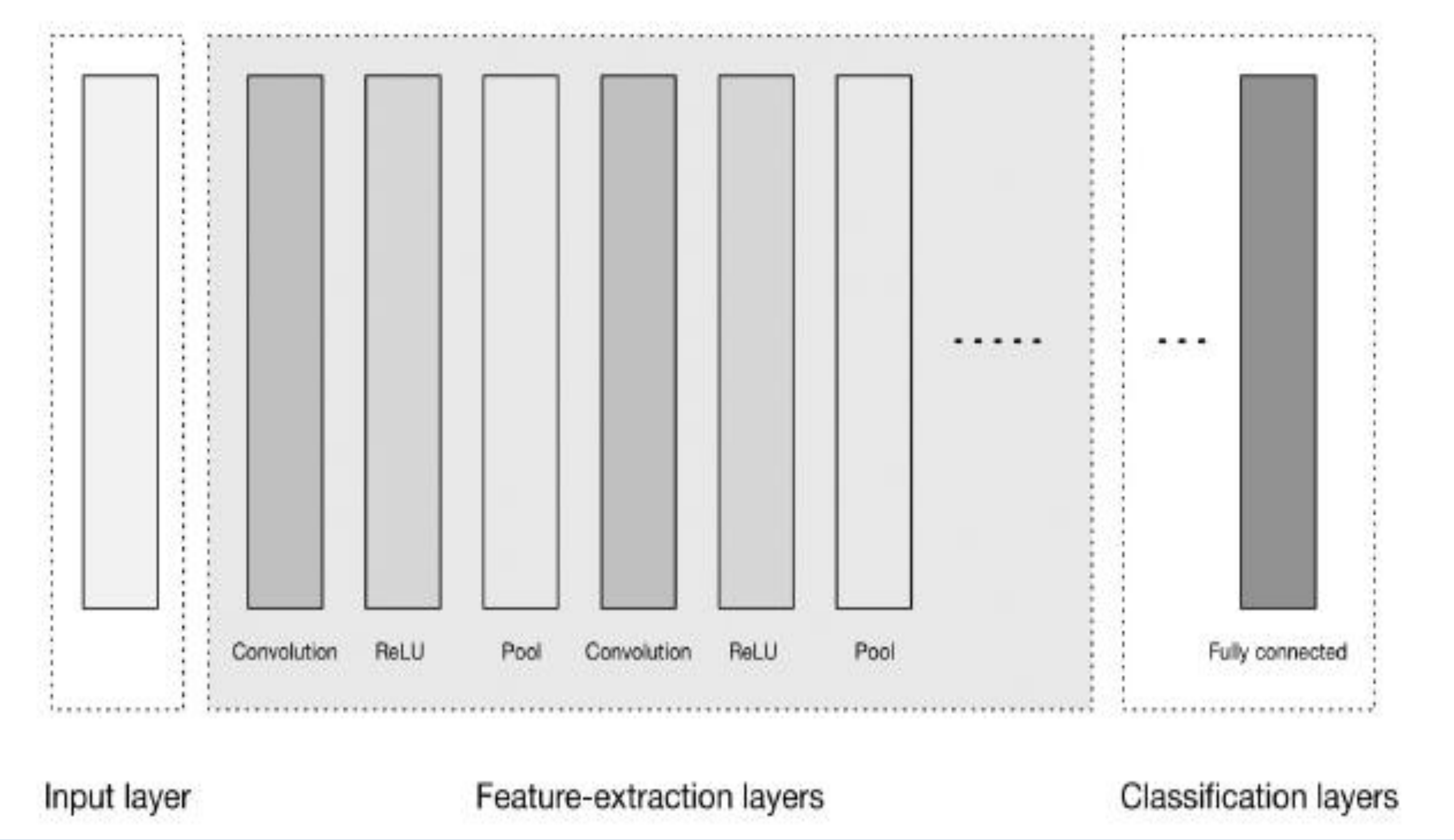
- **Max Pooling**
- Pooling operation that selects the maximum element from the region.



- **Average Pooling**
- Computes the average of the elements present in the region of feature map.



CNN Architecture

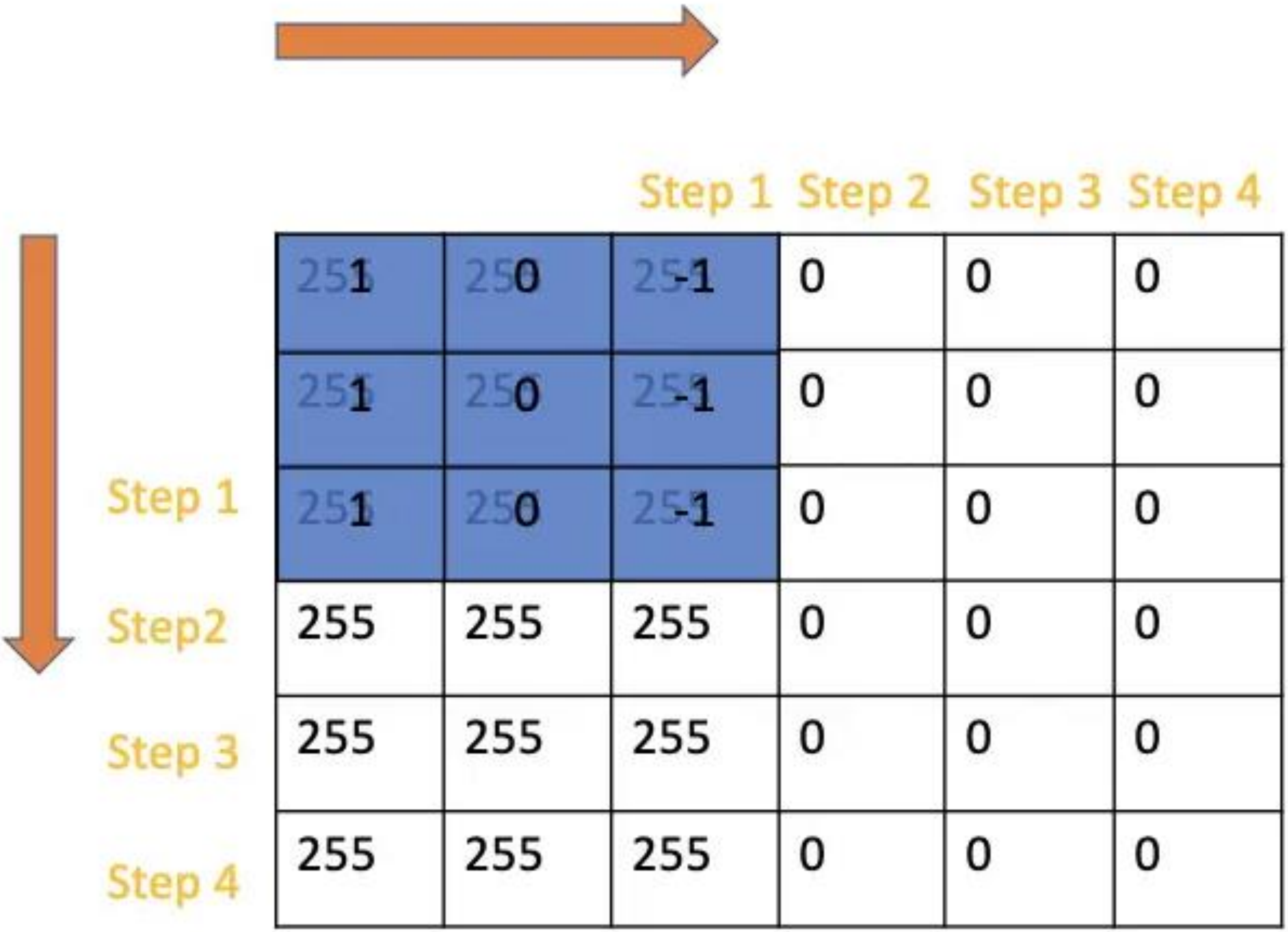


- Three major groups:
- 1. Input layer
- 2. Feature-extraction (learning) layers
- 3. Classification layers

- **Input layer**
- Accepts three-dimensional input .
- Generally in the form spatially of the size (width × height) of the image and has a depth representing the color channels (RGB color channels).
- **Feature-extraction layers**
- General repeating pattern of the sequence:
- Convolution layer
- Rectified Linear Unit (ReLU) activation function as a layer
- Pooling layer
- Layers find a number of features in the images.

- **<u>Classification layers</u>**
- One or more fully connected layers to take the higher-order features.
- Output of the layers produces a two dimensional
- $[b \times N]$
- b is the number of examples in the mini-batch
- N is the number of classes interested in scoring.

Padding and Stride


Step 1
Step 2
Step 3
Step 4
Step 1
Step2
Step 3
Step 4
1 0 -1
1 0 -1
1 0 -1
255 255 255 0 0 0
255 255 255 0 0 0
255 255 255 0 0 0

Padding and Stride

- **Padding describes the addition of empty pixels around the edges of an image.**
- **Proper use of padding ensures that important features are captured and the network can learn effectively from the input data.**
- **Need of Padding :**
- In standard convolution operation, the image shrinks by a factor equivalent to the filter size plus one.
- Take an image of width and height 6, and a filter of width and height 3, the image shrinks by

$$6-3+1=4$$

- Resulting image will be 4×4 dimensions instead of 6×6.
- General formula for calculating the shrinkage of the image dimensions m x m based on the kernel size f x f
- $(m\times m)*(f\times f)=(m-f+1)*(m-f+1)$
- **<u>Two problems:</u>**

1. Perform multiple convolution operations, the final image might become vanishingly small.
2. Cannot slide the full filter over the edge pixels.

   Result lose some information at the edges.

Padding Work

- **Pad** the images with additional empty pixels around the edges.

	255	255	255	0	0	0	
	255	255	255	0	0	0	
	255	255	255	0	0	0	
	255	255	255	0	0	0	
	255	255	255	0	0	0	
	255	255	255	0	0	0	

Padding

- Consider an input feature map of size 5x5 and a 3x3 convolutional filter:

$$\text{Input: } \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

- Output after applying 3x3 filter with valid padding: 3x3

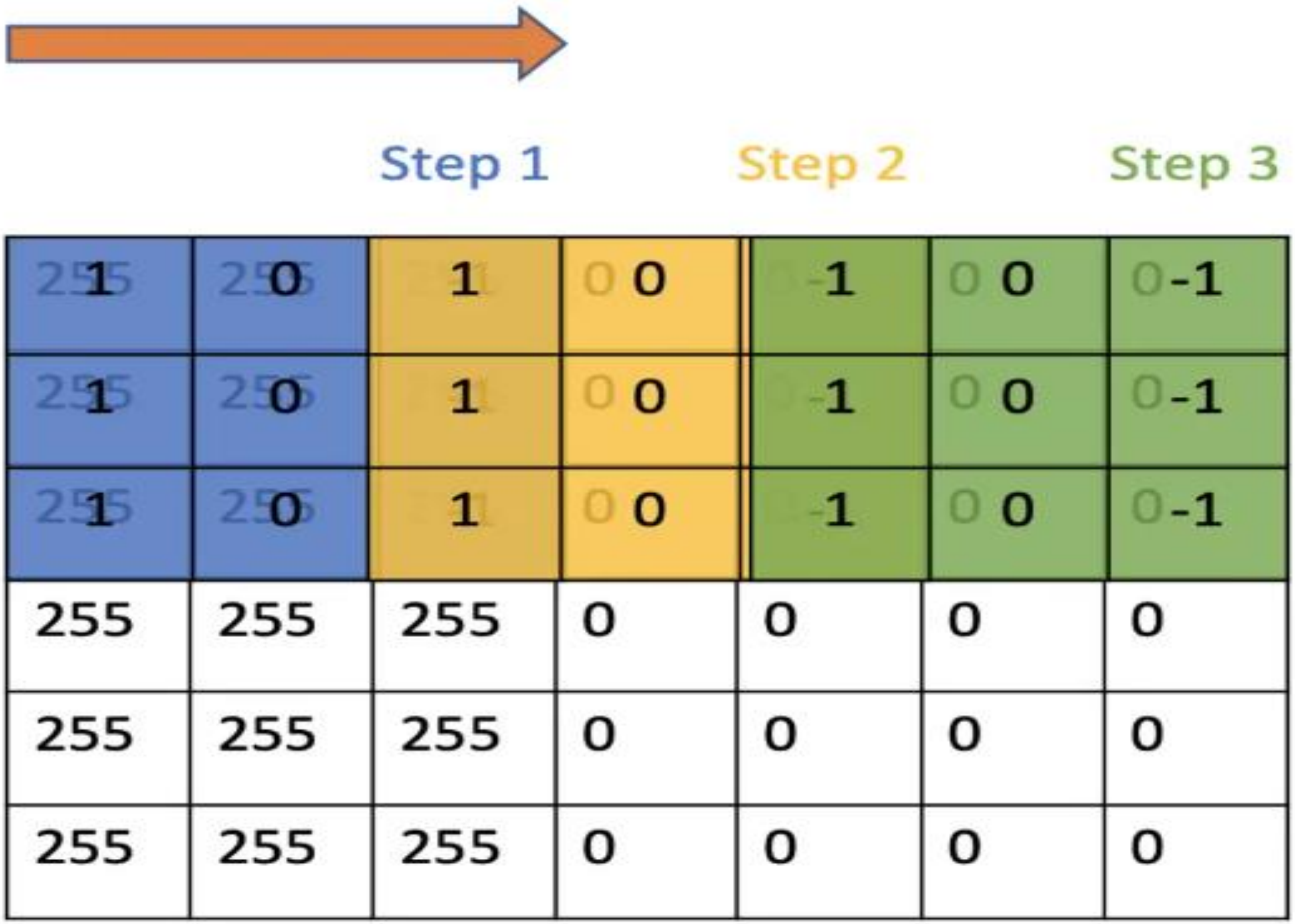
Padding

Padded Input: $$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 & 9 & 10 & 0 \\ 0 & 11 & 12 & 13 & 14 & 15 & 0 \\ 0 & 16 & 17 & 18 & 19 & 20 & 0 \\ 0 & 21 & 22 & 23 & 24 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- Output after applying 3x3 filter with same padding: 5x5

Stride

- **Stride describes the process of increasing the step size by which filter slide over an input image.**
- **Stride of 1**:
  - The filter moves one pixel at a time.
  - Produces the highest possible resolution output, as the filter covers every possible position.
- **Stride Greater than 1**:
  - The filter moves more than one pixel at a time (e.g., 2, 3).
  - Reduces the spatial dimensions of the output feature map more significantly, leading to a coarser feature map.


Step 1
Step 2
Step 3
1 0 1 0 -1 0 -1
1 0 1 0 -1 0 -1
1 0 1 0 -1 0 -1
255 255 255 0 0 0 0
255 255 255 0 0 0 0
255 255 255 0 0 0 0

Example

- Consider a 32x32 image that goes through several layers in a CNN:
- **Initial Image (32x32):**

$$\begin{bmatrix} 1 & 2 & 3 & \dots & 32 \\ 33 & 34 & 35 & \dots & 64 \\ 65 & 66 & 67 & \dots & 96 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 961 & 962 & 963 & \dots & 1024 \end{bmatrix}$$

Example

- **After Convolution (3x3 filter, stride 2, no padding):**
- **Output:** 15x15 feature map (coarser than the original).

$$\begin{bmatrix} 1 & 3 & 5 & \dots & 29 \\ 65 & 67 & 69 & \dots & 93 \\ 129 & 131 & 133 & \dots & 157 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 961 & 963 & 965 & \dots & 989 \end{bmatrix}$$

Example

- **After Pooling (2x2, stride 2):**
- **Output:** 7x7 feature map (even coarser).

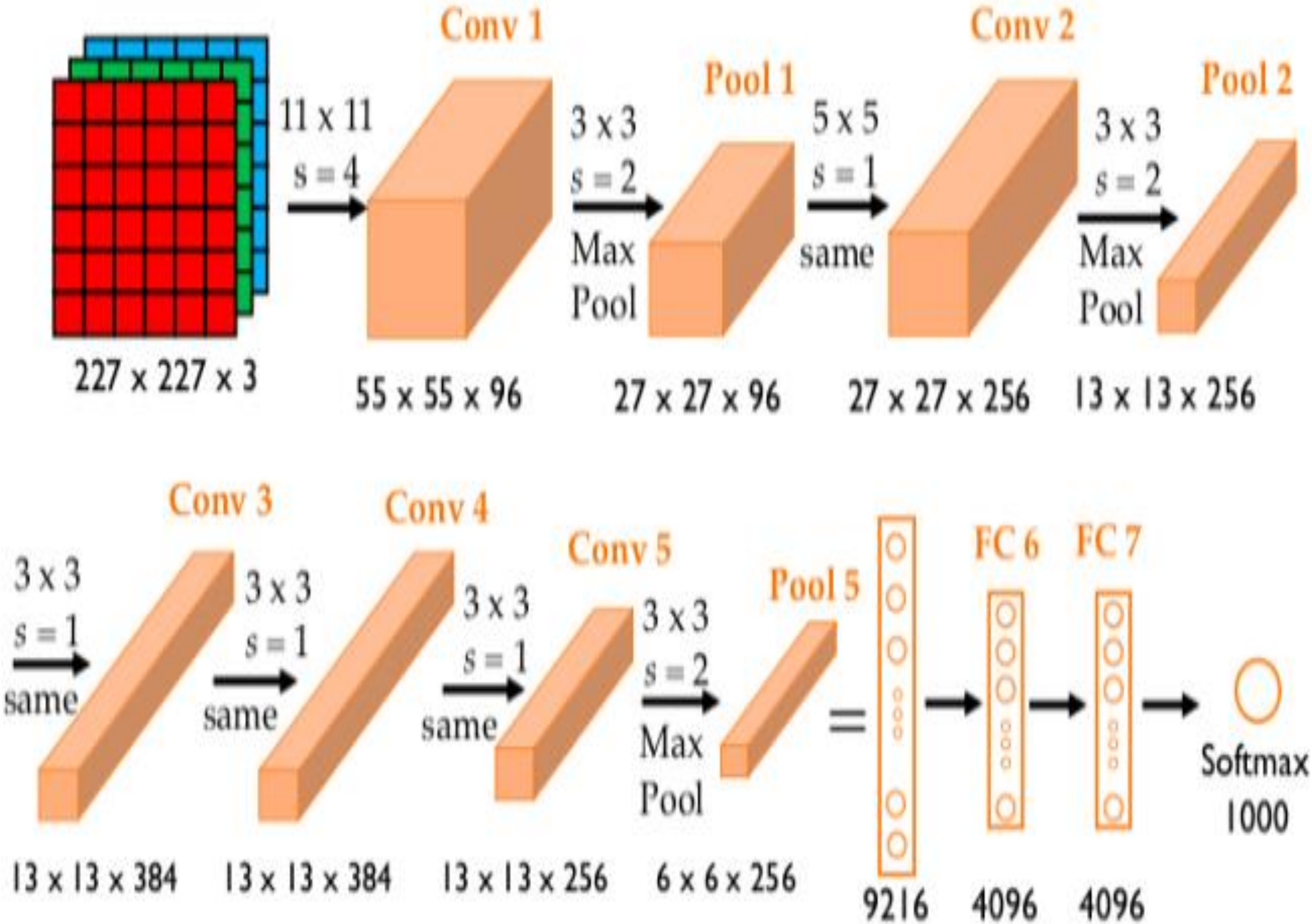
$$\begin{bmatrix} 1 & 5 & 9 & \dots & 29 \\ 129 & 133 & 137 & \dots & 157 \\ 257 & 261 & 265 & \dots & 285 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 961 & 965 & 969 & \dots & 989 \end{bmatrix}$$

AlexNet

- Designed by Alex Krizhevsky.
- Expensive in computation.
- Made feasible - GPUs or Graphical Processing Units, during training.
- First convolutional network which used GPU to boost performance.

- **AlexNet architecture consists of**
- 5 convolutional layers.
- 3 max-pooling layers.
- 2 normalization layers.
- 2 fully connected layers,
- and 1 SoftMax layer.
- Convolutional layer consists: Convolutional filters and a nonlinear activation function ReLU.

- Pooling layers are used to perform max pooling.
- Input size is 224x224x3 .
- Some padding as 227x227x3 .
- Overall 60 million parameters.
- **Model Details**
- ReLU is an activation function.
- Batch size of 128.
- Used Normalization layers.
- SGD Momentum as learning algorithm.


Conv 1
Pool 1
Conv 2
Pool 2
11 x 11
s = 4
3 x 3
s = 2
Max
Pool
5 x 5
s = 1
same
3 x 3
s = 2
Max
Pool
227 x 227 x 3
55 x 55 x 96
27 x 27 x 96
27 x 27 x 256
13 x 13 x 256
Conv 3
Conv 4
Conv 5
Pool 5
FC 6
FC 7
3 x 3
s = 1
same
3 x 3
s = 1
same
3 x 3
s = 1
same
3 x 3
s = 2
Max
Pool
=
Softmax
1000
13 x 13 x 384
13 x 13 x 384
13 x 13 x 256
6 x 6 x 256
9216
4096
4096

Input Layer

- **Input**: The input to the network is an RGB image of size 227x227x3.
- Typically, the images are resized to this dimension before being fed into the network.
- **Preprocessing**: Each image is mean-subtracted (the mean image is computed from the training set) to normalize the input.

First Convolutional Layer (Conv1)

- **Operation:** 96 Filters Of Size 11x11 With A Stride Of 4 And Padding Of 0 Are Applied.
- **Output Size Calculation:**
- Output Size=(227−11)/4+1=55
- Output Is 55x55x96.
- **Activation Function:** ReLU is applied after convolution to introduce non-linearity.

First Max-Pooling Layer (Pool1)

- **Operation:** Max-pooling with a 3x3 filter and a stride of 2.
- **Output Size Calculation:**
- Output size=(55−3)/2+1=27
- Thus, the output is 27x27x96.

- **Local Response Normalization (LRN)**
- Applied to the output of the first pooling layer to normalize the activations.

Second Convolutional Layer (Conv2)

- **Operation:** 256 filters of size 5x5 with a stride of 1 and padding of 2 are applied.
- **Output Size Calculation:**
- Output size=27
- Thus, the output is 27x27x256.
- **Activation Function:** ReLU is applied after convolution.

Second Max-Pooling Layer (Pool2)

- **Operation:** Max-pooling with a 3x3 filter and a stride of 2.
- **Output Size Calculation:**
  Output size $=(27-3)/2+1=13$
- Thus, the output is 13x13x256.

Local Response Normalization (LRN)

- Applied again to the output of the second pooling layer.

Third Convolutional Layer (Conv3)

- **Operation:** 384 filters of size 3x3 with a stride of 1 and padding of 1 are applied.
- **Output Size Calculation:**
- Output size=13
- Thus, the output is 13x13x384.
- **Activation Function:** ReLU is applied after convolution.

Fourth Convolutional Layer (Conv4)

- **Operation:** 384 filters of size 3x3 with a stride of 1 and padding of 1 are applied.
- **Output Size Calculation:**
- Output size=13
- Thus, the output is 13x13x384.
- **Activation Function:** ReLU is applied after convolution.

Fifth Convolutional Layer (Conv5)

- **Operation:** 256 filters of size 3x3 with a stride of 1 and padding of 1 are applied.
- **Output Size Calculation:**
- Output size=13
- Thus, the output is 13x13x256.
- **Activation Function:** ReLU is applied after convolution

Third Max-Pooling Layer (Pool3)

- **Operation:** Max-pooling with a 3x3 filter and a stride of 2.
- **Output Size Calculation:**
  $\text{Output size} = (13-3)/2+1 = 6$
- Thus, the output is 6x6x256.

Flattening

- The 3D tensor is flattened into a 1D vector.
- The size of the vector is: $6 \times 6 \times 256 = 9216$.

Fully Connected Layer

- **First Fully Connected Layer (FC1)**
- **Operation:** Fully connected layer with 4096 neurons.
- **Activation Function:** ReLU.
- **Dropout:** 50% dropout is applied to prevent overfitting.
- **Second Fully Connected Layer (FC2)**
- **Operation:** Fully connected layer with 4096 neurons.
- **Activation Function:** ReLU.
- **Dropout:** 50% dropout is applied to prevent overfitting.

Fully Connected Layer

- **Third Fully Connected Layer (FC3)**
- **Operation:** Fully connected layer with 1000 neurons (one for each class in ImageNet).
- **Activation Function:** Softmax, providing the class probabilities.