Course Content/Syllabus

Unit 2: Convolutional Neural Network

CNN- From Fully Connected layers to Convolutions, Convolution for images, Padding and Stride, Pooling, Convolution Architectures - Alexnet, NiN, VGGNet, ResNet, DenseNet, Transfer Learning, Case Study: Image Recognition Using CNN.

Outline

- Foundations of Convolutional Neural Networks
- Fully Connected Layers to Convolution
- CNN Operations
- Architecture
- Simple Convolution Network
- Deep Convolutional Models
  - ResNet
  - NiN
  - VGG Net
  - AlexNet
  - InceptionNet
  - DenseNet

Foundations of Convolutional Neural Networks

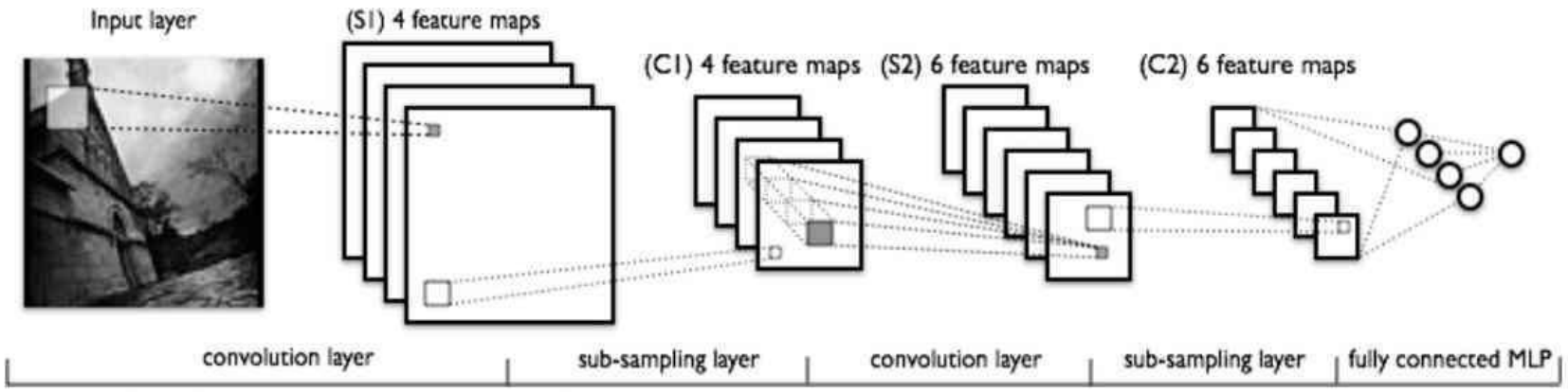
❑ What is Convolutional Neural Network(CNN)?

- ❑ A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks.
- ❑ It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.
- ❑ The architecture of CNNs is inspired by the visual processing in the human brain, and they are well-suited for capturing hierarchical patterns and spatial dependencies within images.

What is a Convolution Neural Network?

ConvNets were initially developed in the neural network image processing community where they achieved break-through results in recognising an object from a pre-defined category.

A Convolutional Neural Network typically involves two operations, which can be thought of as feature extractors: **convolution** and **pooling**.



The Convolutional Neural Network architecture applied to image classification.

(Image adapted from **http://deeplearning.net/**)

The output of this sequence of operations is then typically sent as input to a fully connected layer(s) which is usually a multi-layer perceptron neural network (MLP).

Neural Network and Deep Learning

What is a Convolution Neural Network?

For this image, suppose a classification task is to be performed.

Whether the image contains a bird or not?

This is a classic use case of CNN by which we perform image classification

Why don't we use neural networks for the same?

What is a Convolution Neural Network?

Why not Neural Networks for image classification task?

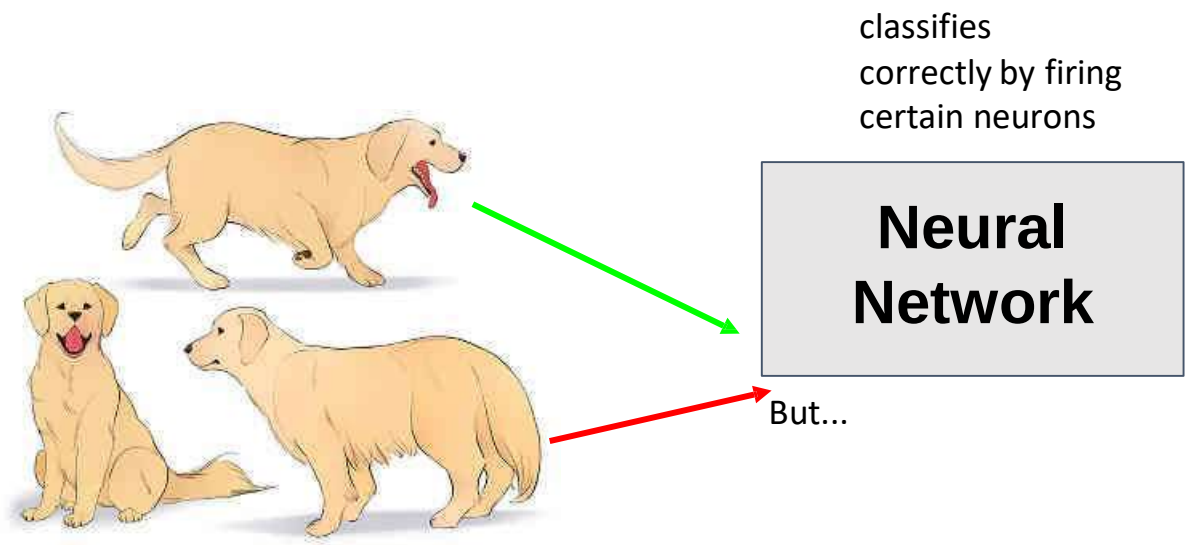
Reason 1: Images are Big

- Images used for computer vision are 224 x 224 or larger and they include **3 colour channels** so 224 x 224 x 3 = 150,528 input features.

- As this propagates into further layers over a **million weights** are required in first layer alone. This would make it nearly impossible to train.

Reason 2: Positions can change

- If a network is trained to detect dogs, you'd want it to be able to a detect a dog regardless of where it appears in the image.



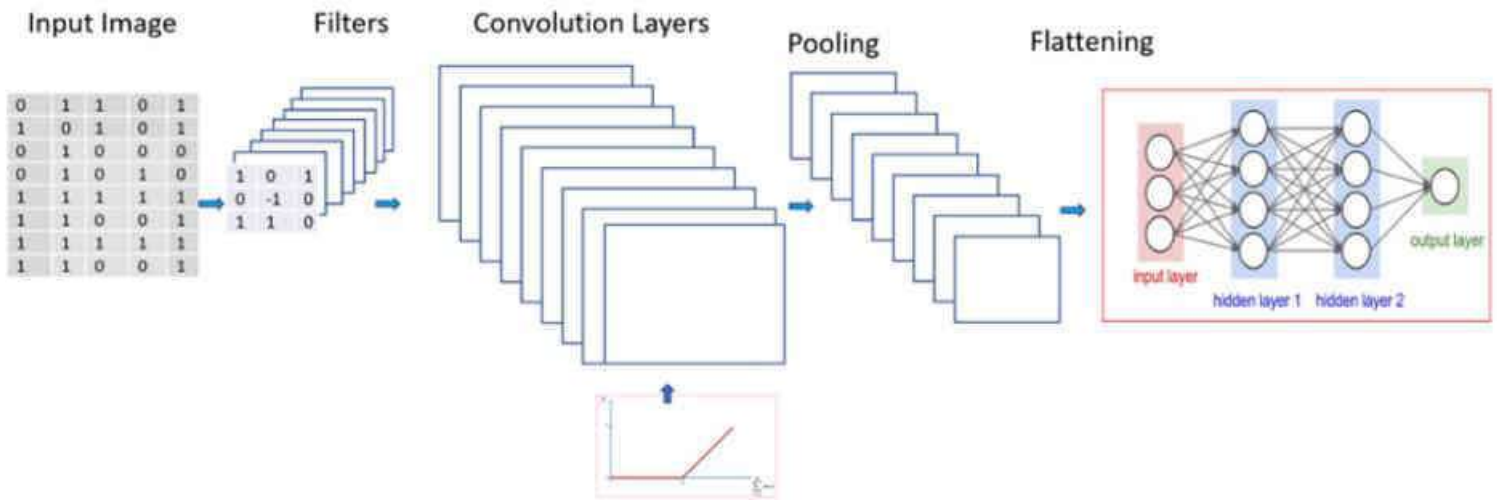
But...
Sending input the same entity in different angles or positions, it **would not activate the same neurons**, so the network would react completely differently!

In such cases CNN can help us solve problems

What is a Convolution Neural Network?

What are Convolutional Neural Networks?

They're basically just neural networks that use Convolutional layers(Conv layers), which are based on the mathematical **operation of convolution**. Convolution layers consist of a set of filters.

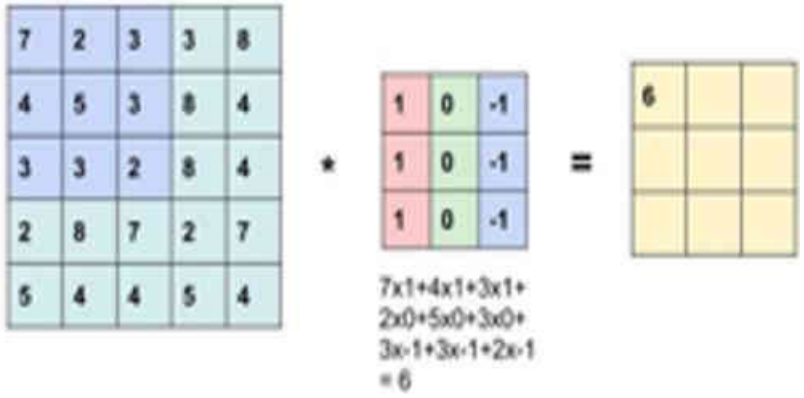


What is a Convolution Neural Network?

- Here element wise multiplication is happening
- If the **Blue matrix is an instance** we seem to get a smaller version of some abstracted version in the yellow box
- Is it **feature Reduction? ?**

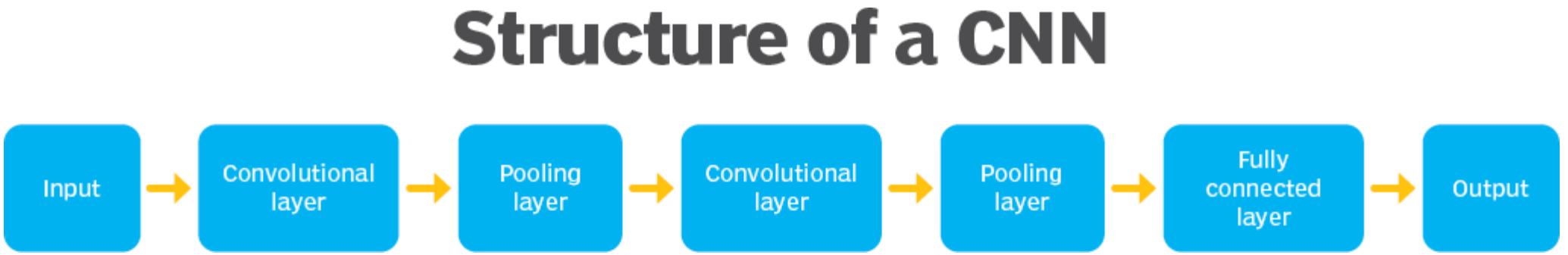
  Nope

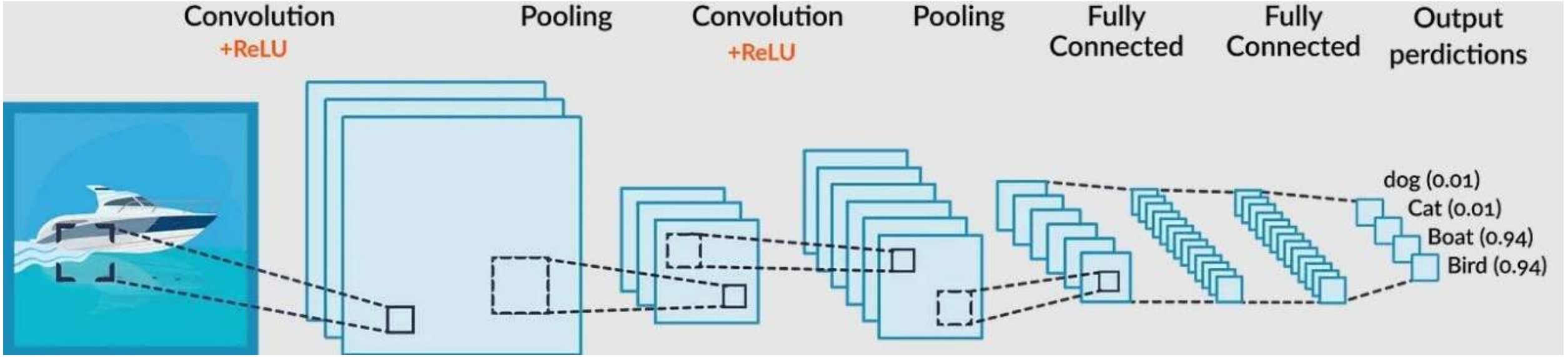
- We can see that some neighbourhood information captured.



Foundations of Convolutional Neural Networks

- **Key components of a Convolutional Neural Network include:**





Foundations of Convolutional Neural Networks

❑ **Key components of a Convolutional Neural Network include:**

1. **Convolutional Layers:** These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.
2. **Pooling Layers:** Pooling layers downsample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. Max pooling is a common pooling operation, selecting the maximum value from a group of neighboring pixels.
3. **Activation Functions:** Non-linear activation functions, such as Rectified Linear Unit (ReLU), introduce non-linearity to the model, allowing it to learn more complex relationships in the data.
4. **Fully Connected Layers:** These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.

Foundations of Convolutional Neural Networks

- CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features that are associated with specific objects or classes.
- Proven to be highly effective in image-related tasks, achieving state-of-the-art performance in various computer vision applications.
- Their ability to automatically learn hierarchical representations of features makes them well-suited for tasks where the spatial relationships and patterns in the data are crucial for accurate predictions.
- CNNs are widely used in areas such as image classification, object detection, facial recognition, and medical image analysis.
- The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes.
- The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information.
- The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

Foundations of Convolutional Neural Networks

❑ Convolutional Neural Network Design

- ❑ The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.
- ❑ It is the sequential design that give permission to CNN to learn hierarchical attributes.
- ❑ In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.
- ❑ The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Foundations of Convolutional Neural Networks

- **Convolutional Neural Network Training** :CNNs are trained using a supervised learning approach.
- This means that the CNN is given a set of labeled training images. The CNN then learns to map the input images to their correct labels.
- The training process for a CNN involves the following steps:

1. **Data Preparation:** The training images are preprocessed to ensure that they are all in the same format and size.
2. **Loss Function:** A loss function is used to measure how well the CNN is performing on the training data. The loss function is typically calculated by taking the difference between the predicted labels and the actual labels of the training images.
3. **Optimizer:** An optimizer is used to update the weights of the CNN in order to minimize the loss function.
4. **Backpropagation:** Backpropagation is a technique used to calculate the gradients of the loss function with respect to the weights of the CNN. The gradients are then used to update the weights of the CNN using the optimizer.

Foundations of Convolutional Neural Networks

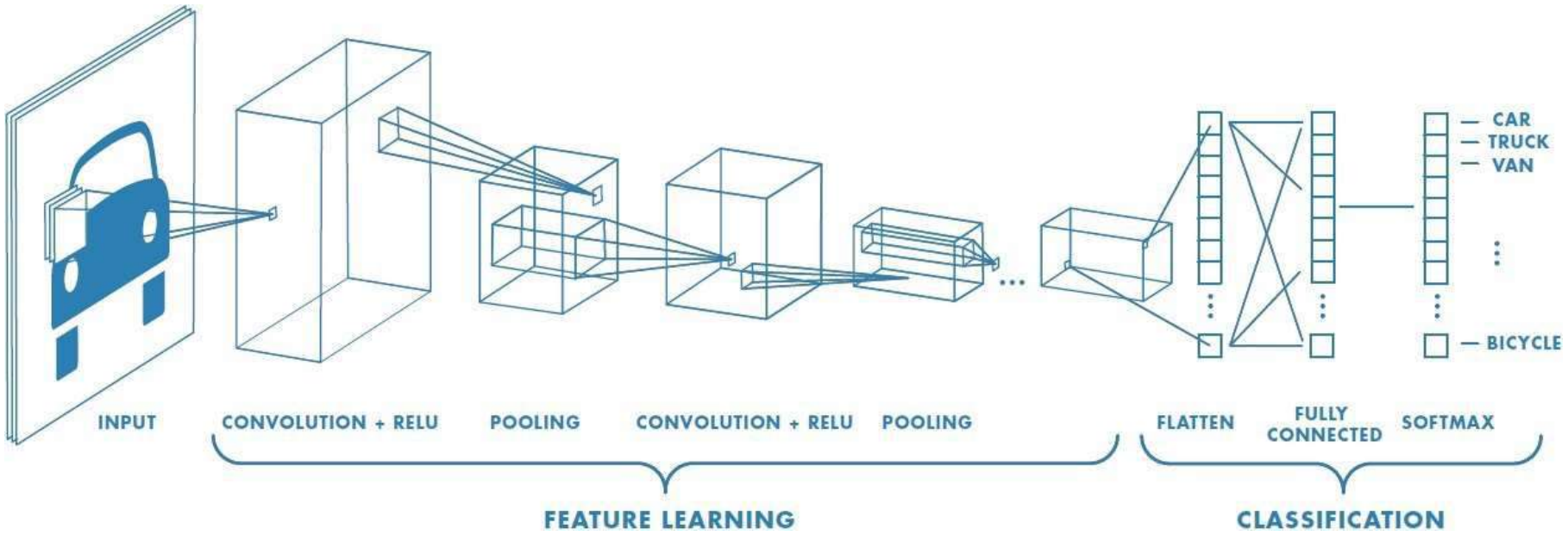
- **CNN Evaluation:** After training, CNN can be evaluated on a held-out test set.
- A collection of pictures that the CNN has not seen during training makes up the test set.
- How well the CNN performs on the test set is a good predictor of how well it will function on actual data.
- The efficiency of a CNN on picture categorization tasks can be evaluated using a variety of criteria.
- Among the most popular metrics are:

1. **Accuracy:** Accuracy is the percentage of test images that the CNN correctly classifies.
2. **Precision:** Precision is the percentage of test images that the CNN predicts as a particular class and that are actually of that class.
3. **Recall:** Recall is the percentage of test images that are of a particular class and that the CNN predicts as that class.
4. **F1 Score:** The F1 Score is a harmonic mean of precision and recall. It is a good metric for evaluating the performance of a CNN on classes that are imbalanced.

Convolutional Neural Network :

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

General Architecture :



CNN Operations

- Convolutional Neural Network is the type of neural network which is used for analysis of visual inputs.
- Some popular applications of CNN includes image classification, image recognition, Natural language processing etc.
- Now, there are certain steps/operations that are involved CNN, these can be categorized as follows:

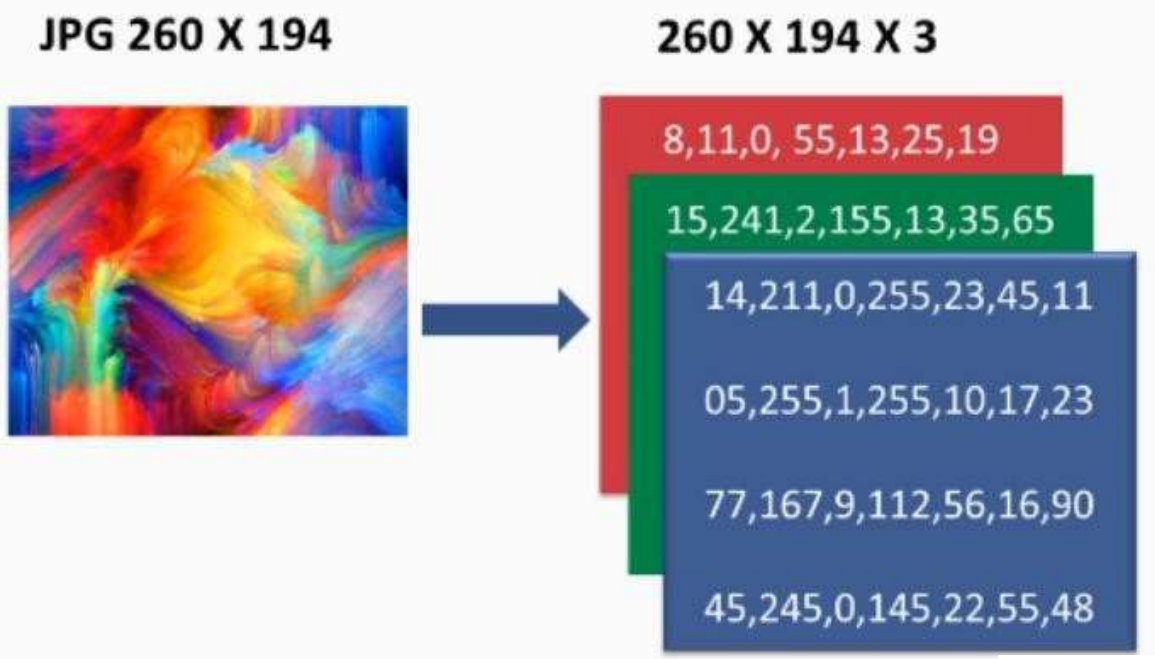
1. **Convolution operation**
2. **Pooling**
3. **Padding**
4. **Stride**
5. **Flattening**
6. **Fully connected layers**

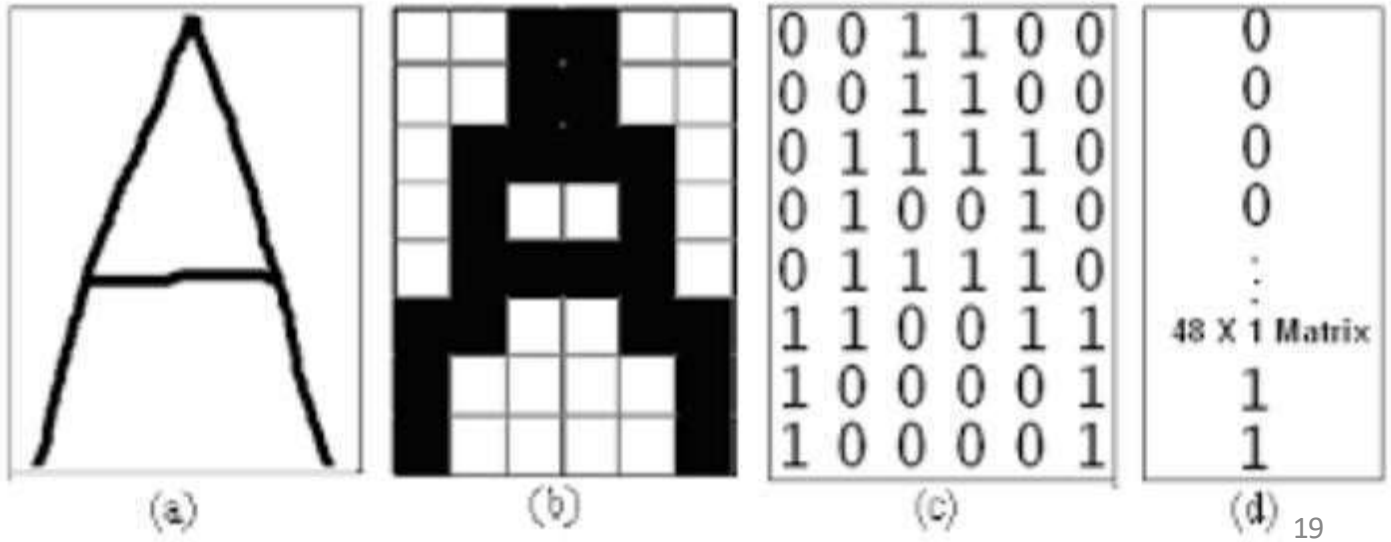
CNN Operations

- **CONVOLUTION OPERATION:** Convolution operations is the first and one of the most important step in the functioning of a CNN.
- Convolution operation focuses on extracting/preserving important features from the input (image etc).
- To understand this operation, let us consider image as input to our CNN. Now when image is given as input, they are in the form of matrices of pixels.
- If the image is grayscale, then the image is considered as a matrix, each value in the matrix ranges from 0-255.
- We can even normalize these values lets say in range 0-1. 0 represents white and 1 represents black.
- If the image is colored, then three matrices representing the RGB colors with each value in range 0-255.
- The same can be seen in the images in the next page:

CNN Operations

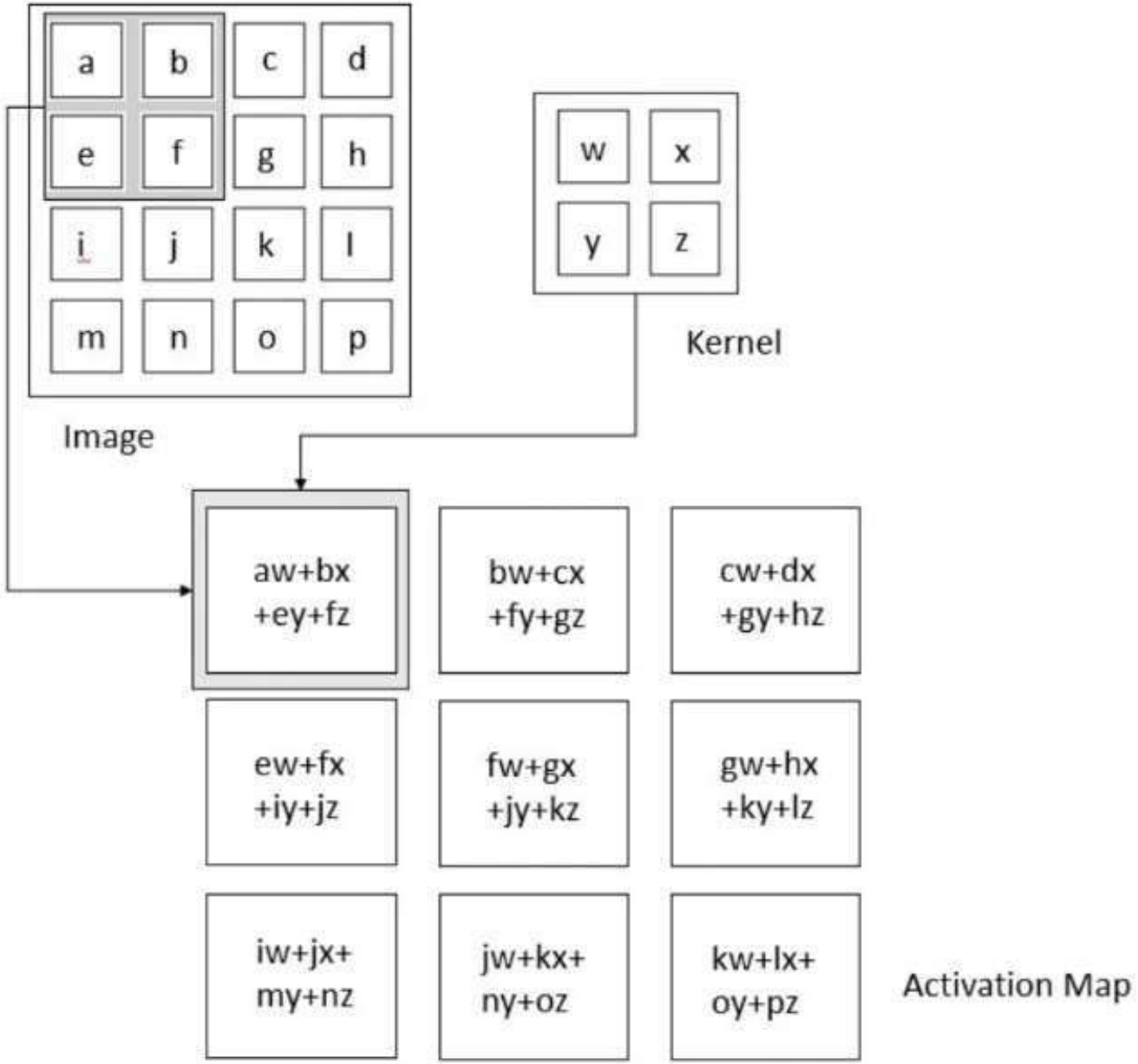
❑ CONVOLUTION OPERATION:





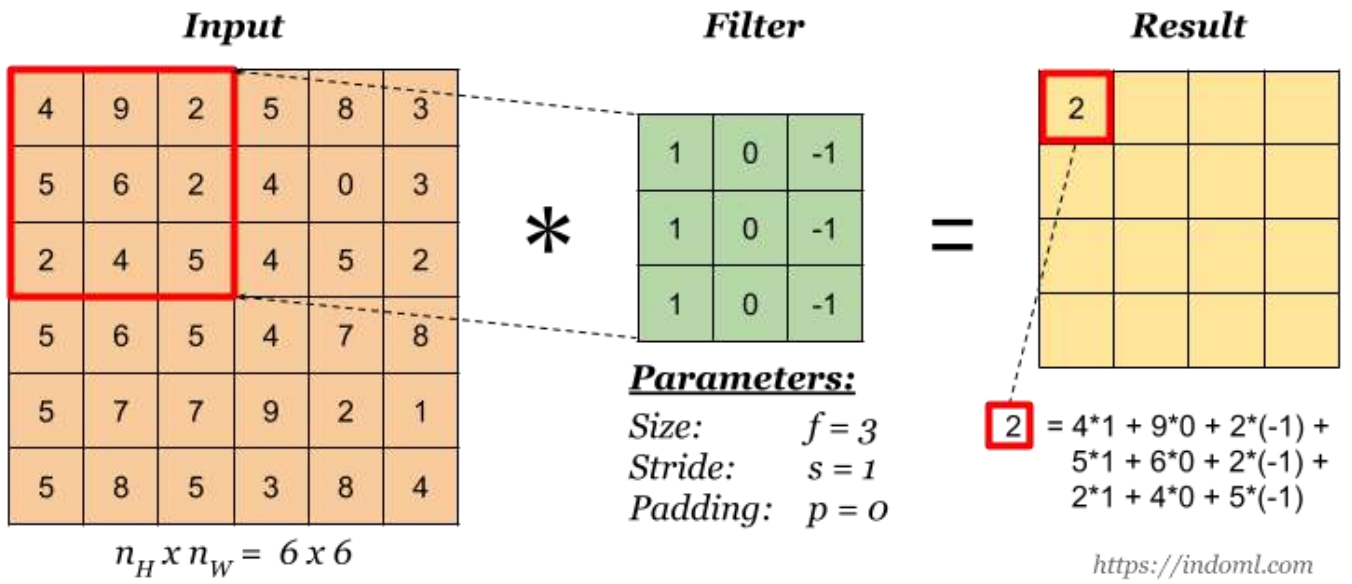
CNN Architecture

❑ Convolutional Neural Network Architecture: Convolution Layer



CNN Operations

- **CONVOLUTION OPERATION:** Consider the image given below:
- Here, input of size 6x6 is given and kernel of size 3x3 is used.
- The feature map obtained in of size 4x4.
- To increase non-linearity in the image, Rectifier function can be applied to the feature map.
- Finally, after the convolution step is completed and feature map is obtained, this map is given as input to the pooling layer.



❑ Convolutional Neural Network Architecture: Convolution Layer

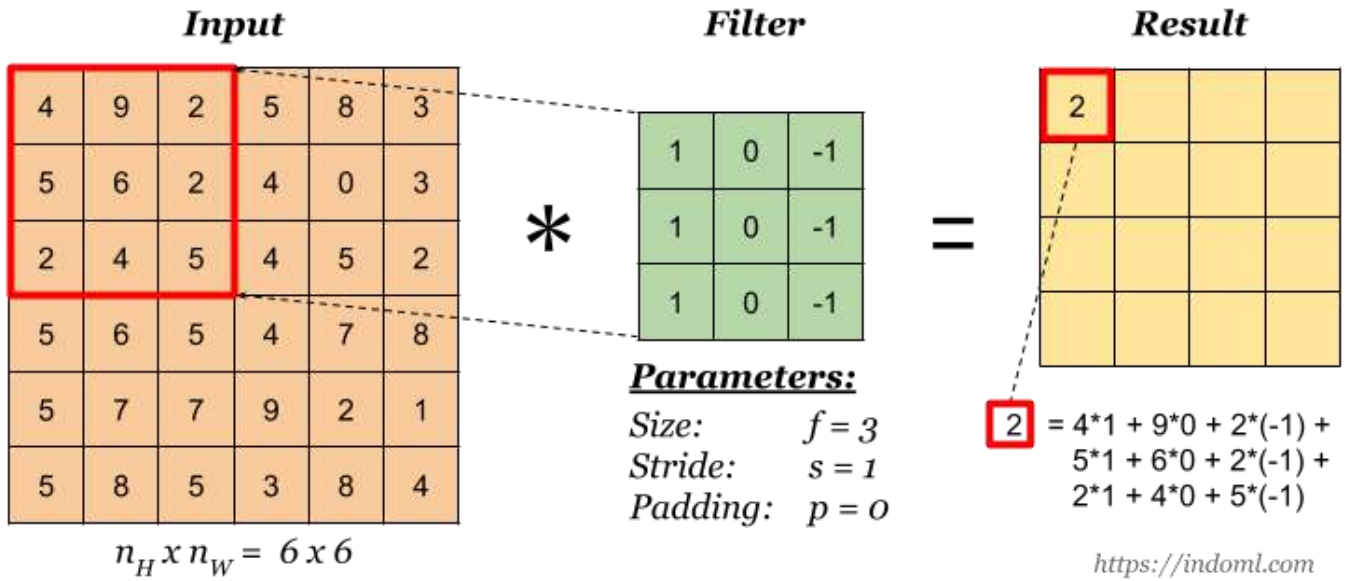
- ❑ During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region.
- ❑ This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image.
- ❑ The sliding size of the kernel is called a stride.
- ❑ If we have an input of size **W x H x D** and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

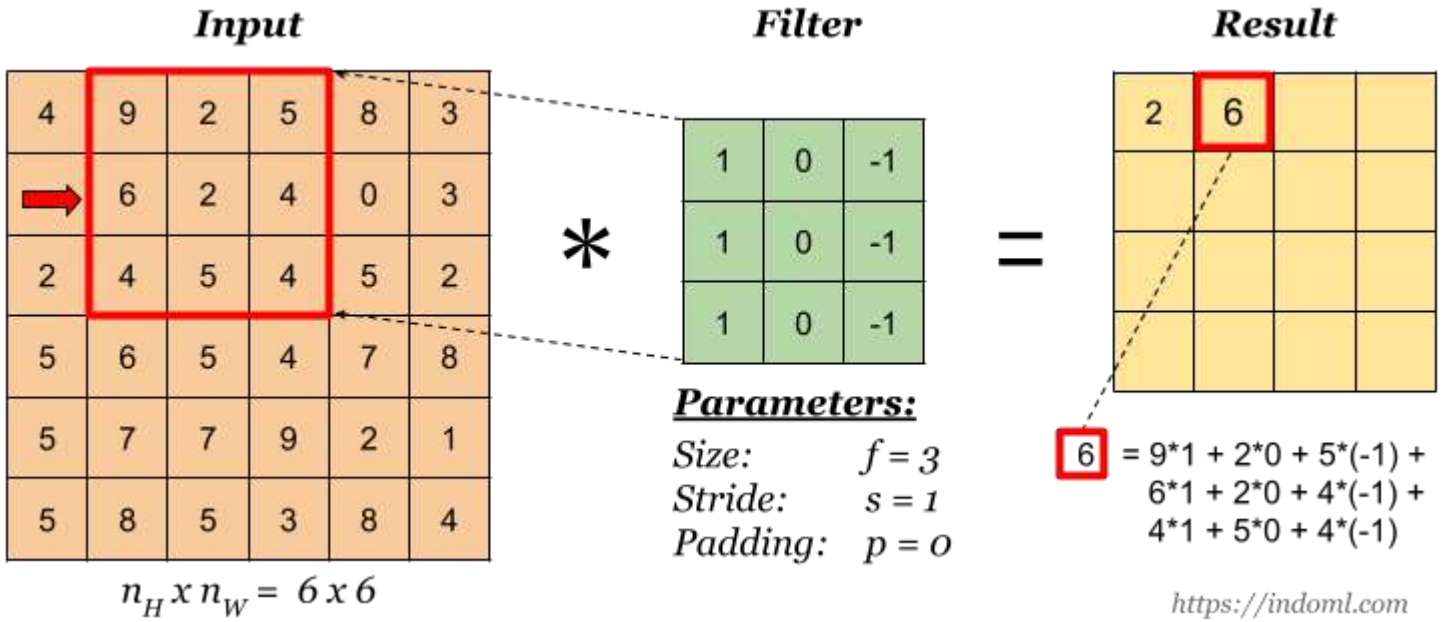
Convolution Layer — The Kernel

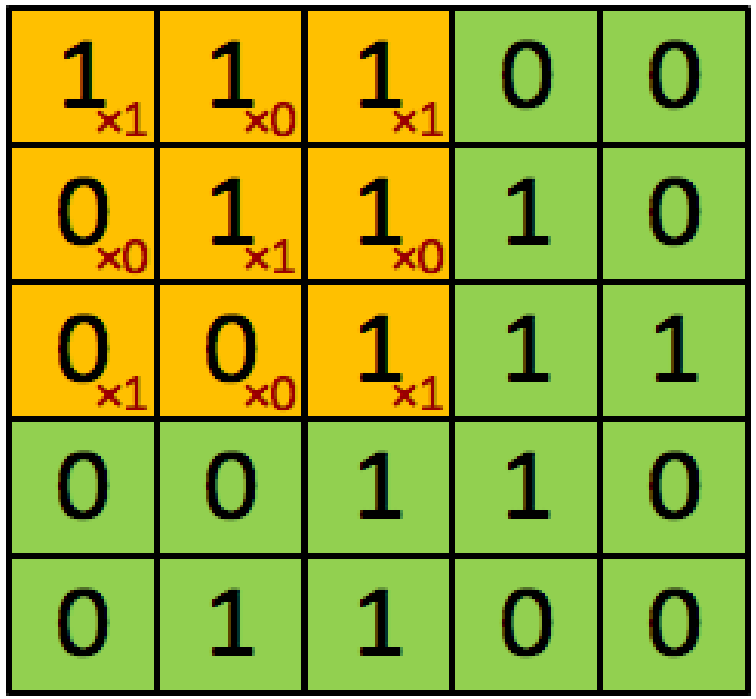
Basic Convolution Operation

Step 1: overlay the filter to the input, perform element wise multiplication, and add the result.

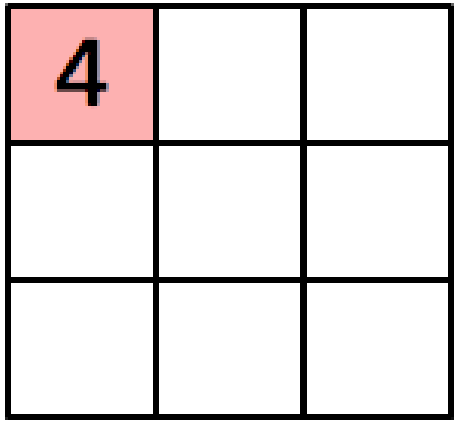


Step 2: move the overlay right one position (or according to the **stride** setting), and do the same calculation above to get the next result. And so on.





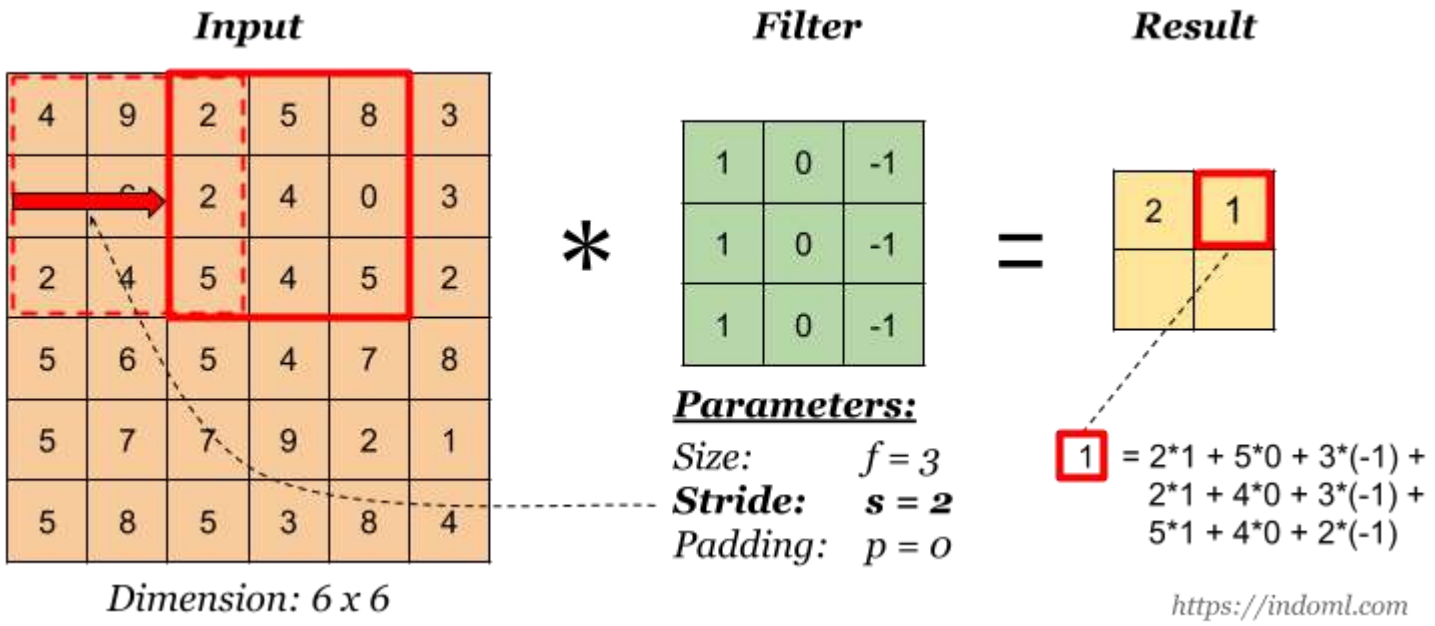
Image



Convolved Feature

Stride

Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



CNN Operations

❑ **Padding:** CNN has offered a lot of promising results but there are some issues that comes while applying convolution layers. There are two significant problems:

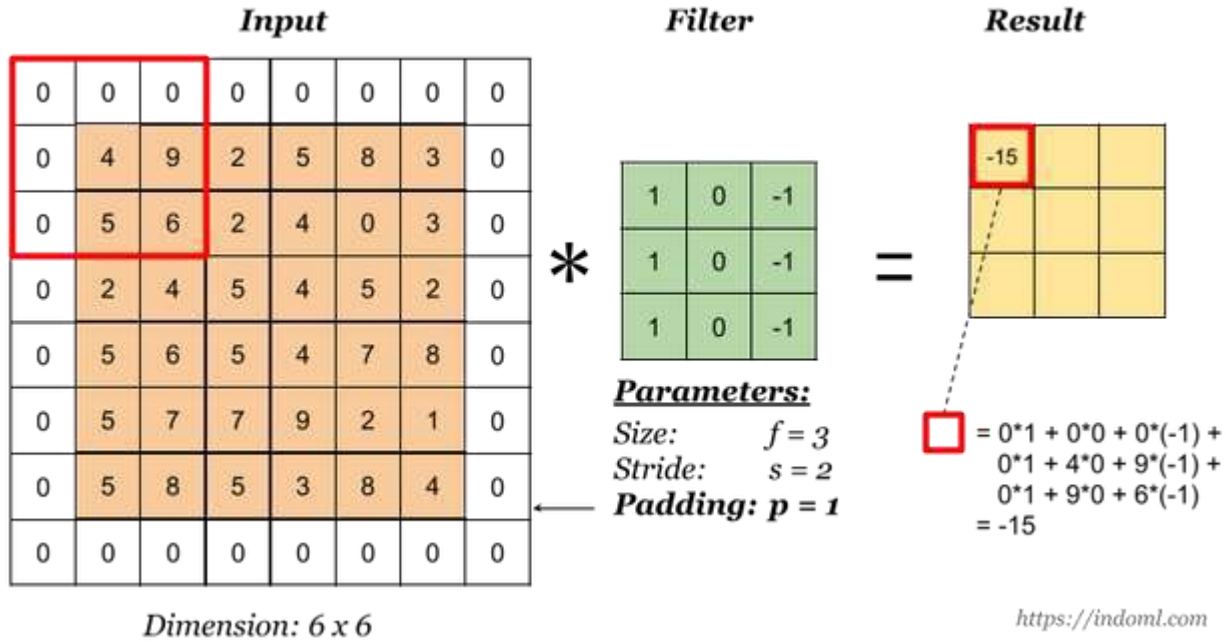
1. When we apply convolution operation, based on the size of image and filter, the size of the resultant processed image reduces according to the following rule:
    - ❑ Let image size: nxn
    - ❑ Let filer size: mxm
    - ❑ Then, resultant image size: (n-m+1)x(n-m+1)
2. Another problem comes with the pixel values in the edges of the image matrix. The values on the edges of the matrix do not contribute as much as the values in the middle. Like if some pixel value resides in the corner i.e. (0,0) position, then it will be considered only once in the convolution operation, while values in the middle will be considered multiple times.

   To overcome these important problems, padding is the solution. In padding, we layer of values like adding layer(s) of zeros around the feature matrix, as shown in the below image.

Padding

It allows us to use a CONV layer without necessarily shrinking the height and width of the volumes.

It helps us keep more of the information at the border of an image. Without padding, very few values at the next layer would be affected by pixels as the edges of an image.



CNN Operations

❑ POOLING:

- ❑ This step helps to maintain spatial invariance or even deal with other kind of distortions in the process.
- ❑ This means that if we are trying to perform image recognition or something like checking if the image contains a dog, there is a possibility that the image might not be straight (maybe tilted), or texture difference is there, the object size in the image is small etc.
- ❑ So, this should not let our model to provide incorrect output.
- ❑ This is what pooling is all about. There can be different types of pooling like min pooling, max pooling etc.
- ❑ It helps to preserve the essential features.
- ❑ What we obtain is called pooled feature map. Here the size is reduced, features are preserved and distortions are dealt with.

CNN Operations

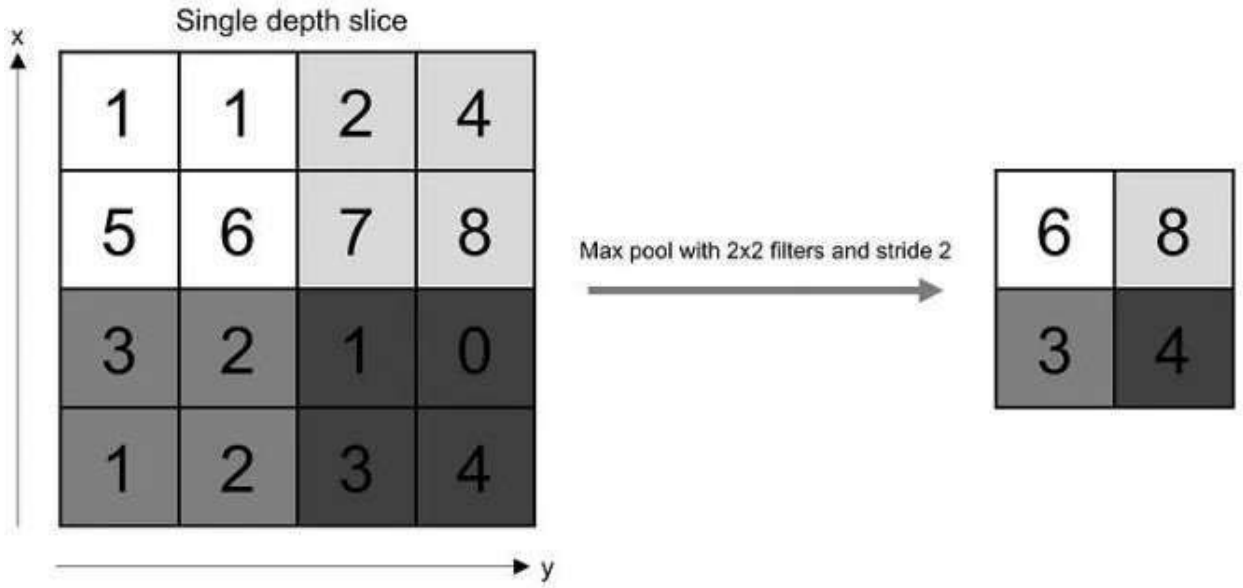
❑ POOLING:

❑ Mentioned below are some types if pooling that are used:

1. **Max Pooling**: In max pooling, the maximum value is taken from the group of values of patch feature map.
2. **Minimum Pooing**: In this type of pooling, the minimum value is taken from the patch in feature map.
3. **Average Pooling**: Here, the average of values is taken.
4. **Adaptive Pooling**: In this type of pooling, we only need to define the output size we need for the feature map. Parameters such as stride etc are automatically calculated.

❑ Convolutional Neural Network Architecture: Pooling Layer

❑ If we have an activation map of size *W* x *H* x *D*, a pooling kernel of spatial size *F*, and stride *S*, then the size of output volume can be determined by the following formula:



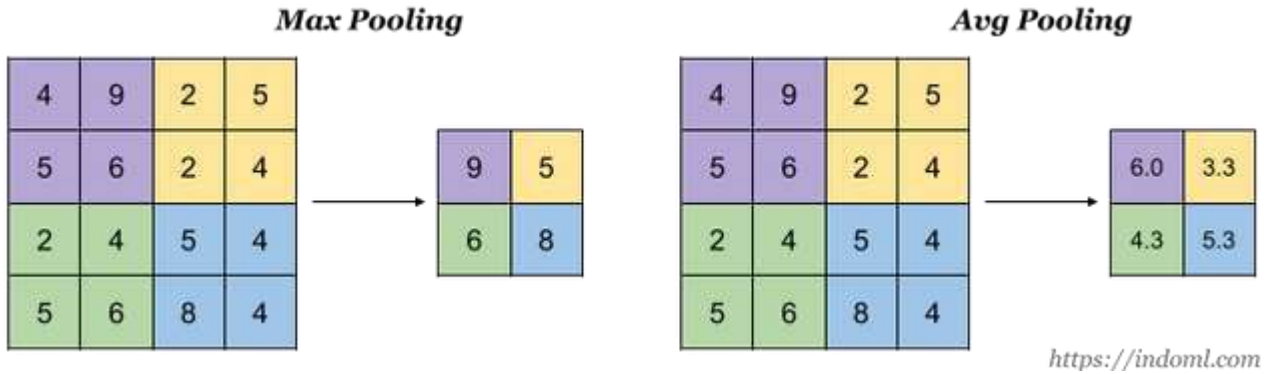
$$W_{out} = \frac{W - F}{S} + 1$$

❑ This will yield an output volume of size *Wout* x *Hout* x *D*. In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

Pooling Layer

Pooling layer is used to reduce the size of the representations and to speed up calculations, as well as to make some of the features it detects a bit more robust.

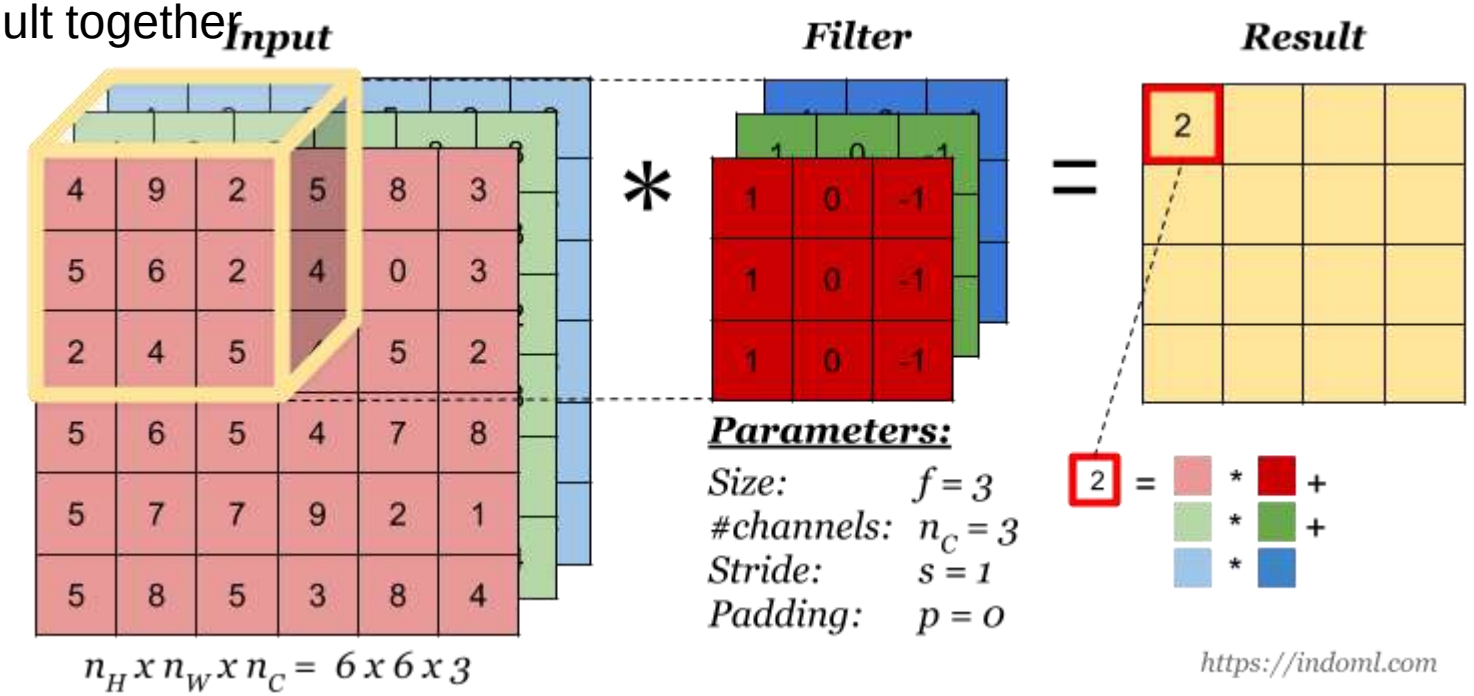
Sample types of pooling are **max pooling** and **avg pooling**, but these days max pooling is more common.

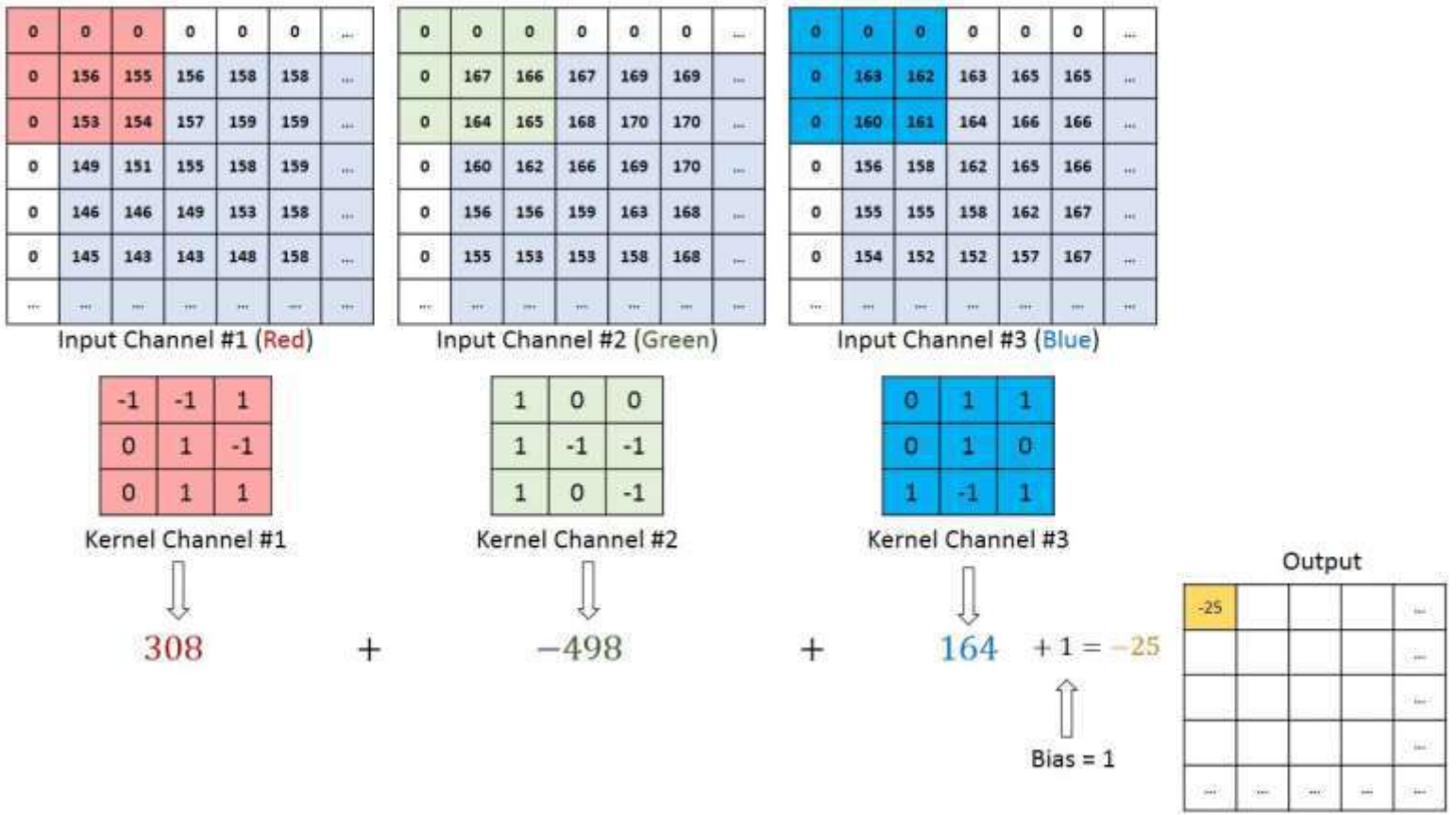


Convolution Operation on Volume

When the input has more than one channels (e.g. an RGB image), the filter should have matching number of channels.

To calculate one output cell, perform convolution on each matching channel, then add the result together.

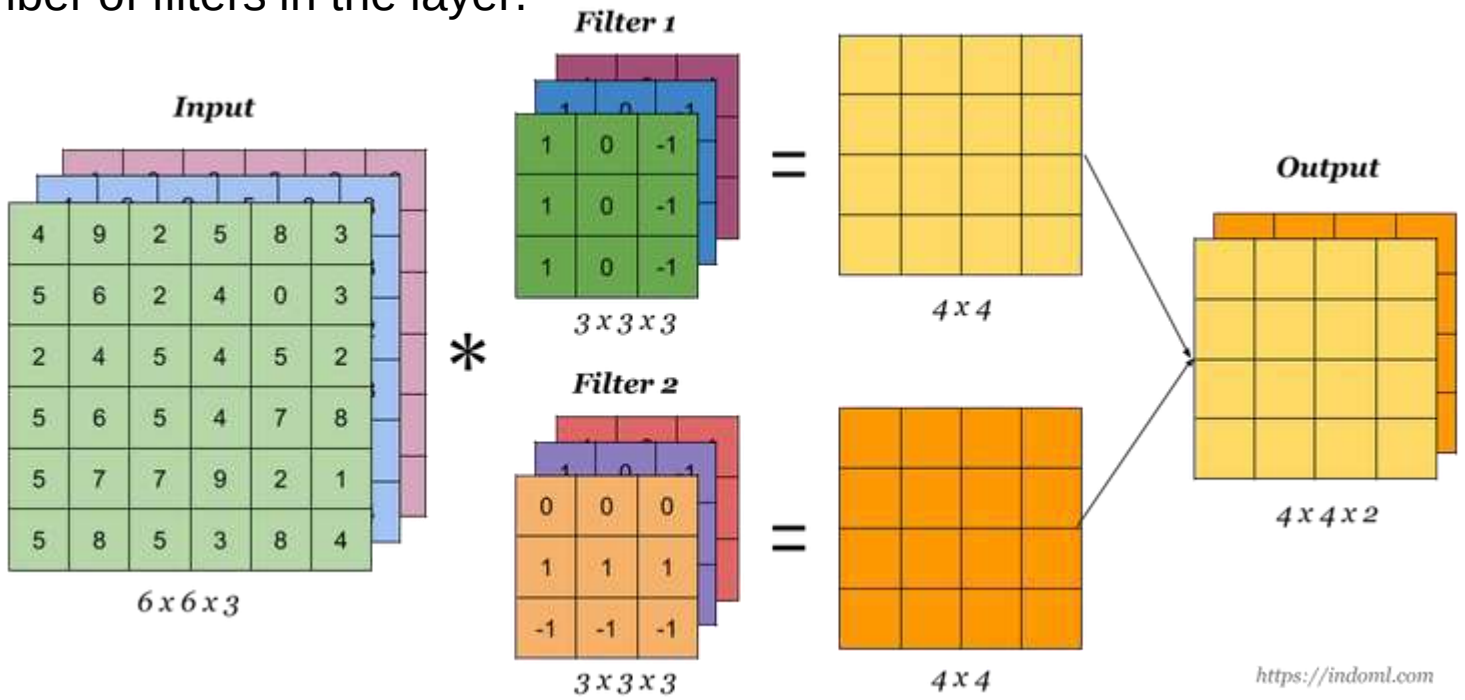



0 0 0 0 0 0 ...
0 156 155 156 158 158 ...
0 153 154 157 159 159 ...
0 149 151 155 158 159 ...
0 146 146 149 153 158 ...
0 145 143 143 148 158 ...
Input Channel #1 (Red)
0 0 0 0 0 0 ...
0 167 166 167 169 169 ...
0 164 165 168 170 170 ...
0 160 162 166 169 170 ...
0 156 156 159 163 168 ...
0 155 153 153 158 168 ...
Input Channel #2 (Green)
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 ...
Input Channel #3 (Blue)
-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
308
+
−498
+
164
+ 1 = −25
Bias = 1
Output
-25

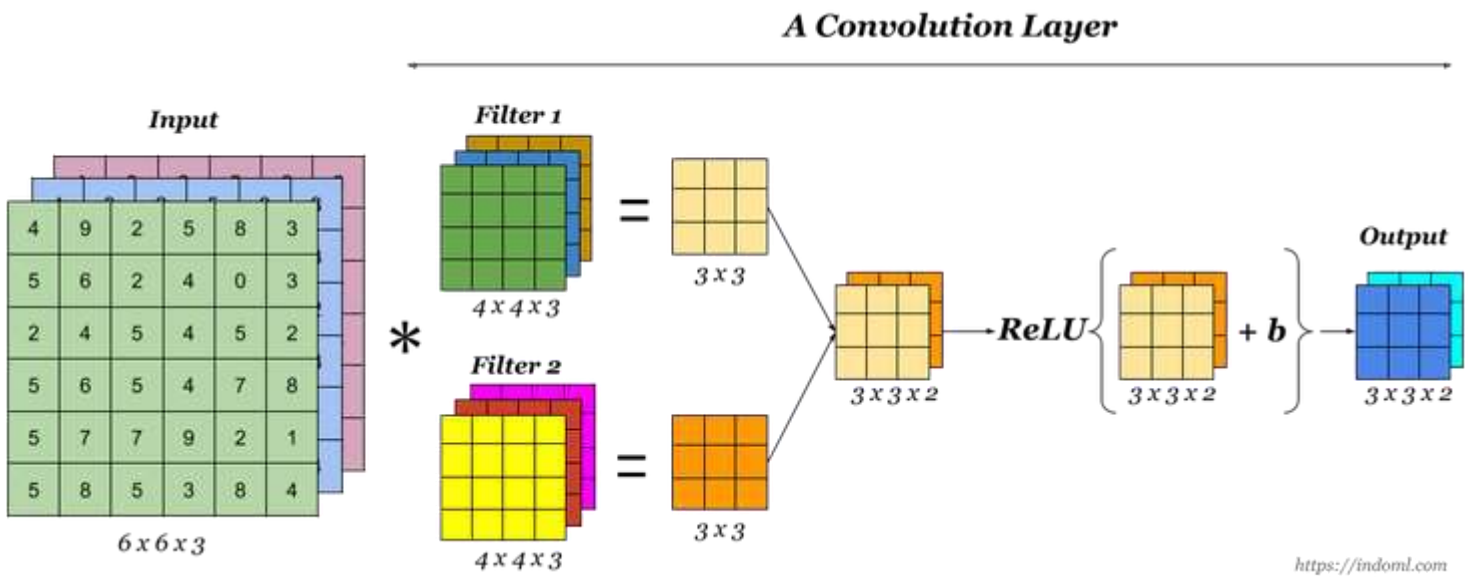
Convolution Operation with Multiple Filters

Multiple filters can be used in a convolution layer to detect multiple features.

The output of the layer then will have the same number of channels as the number of filters in the layer.



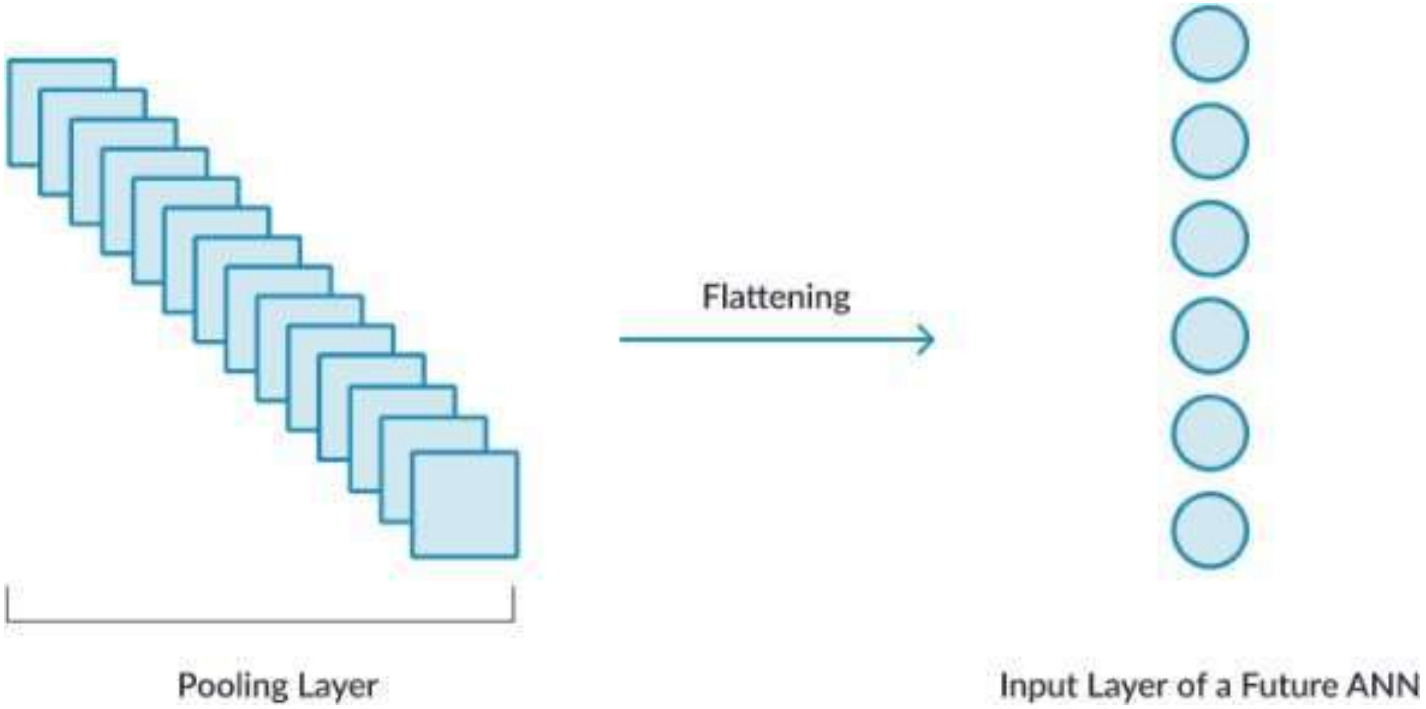
One Convolution Layer



CNN Operations

❑ Flattening

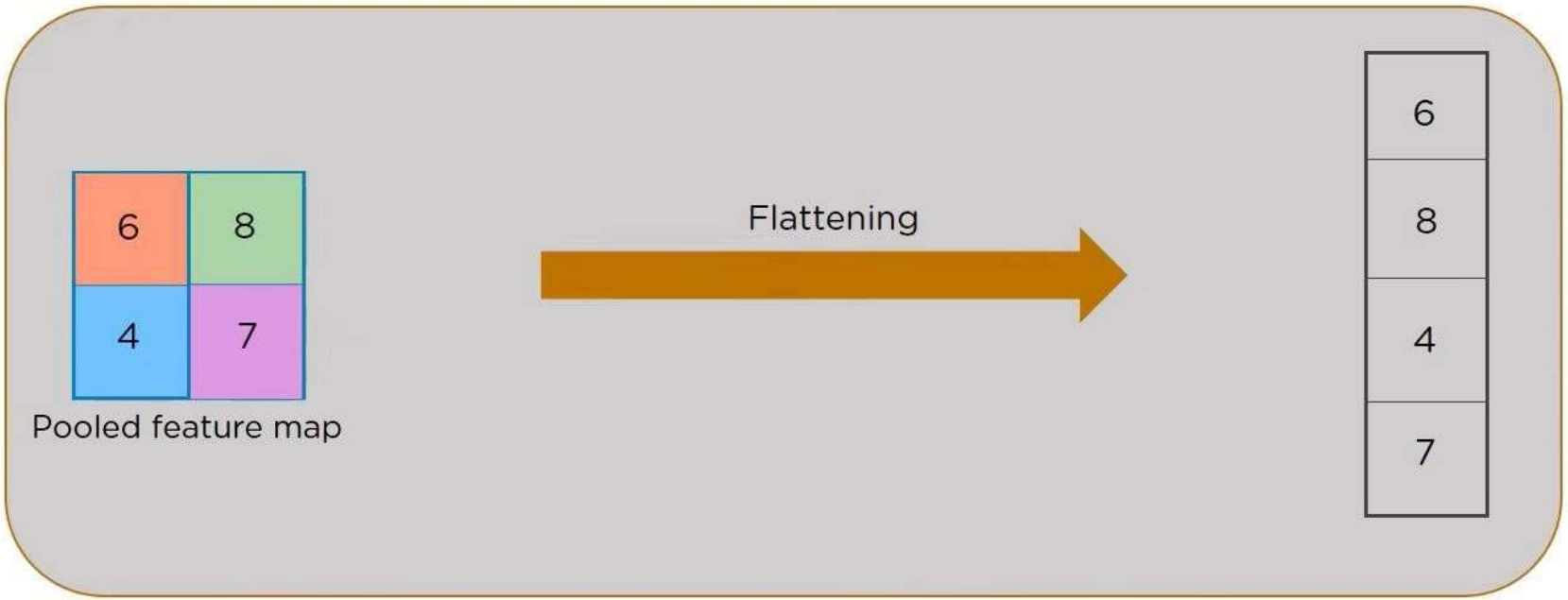
- ❑ This is a step that is used in CNN but not always.
- ❑ Based on the upcoming layers in the CNN, this step is involved.
- ❑ What happens here is that the pooled feature map (i.e. the matrix) is converted into a vector.
- ❑ And this vector plays the role of input layer in the upcoming neural networks.



CNN Operations

❑Flattening

Flattening is the process of converting all the resultant 2 dimensional arrays from pooled feature map into a single long continuous linear vector.



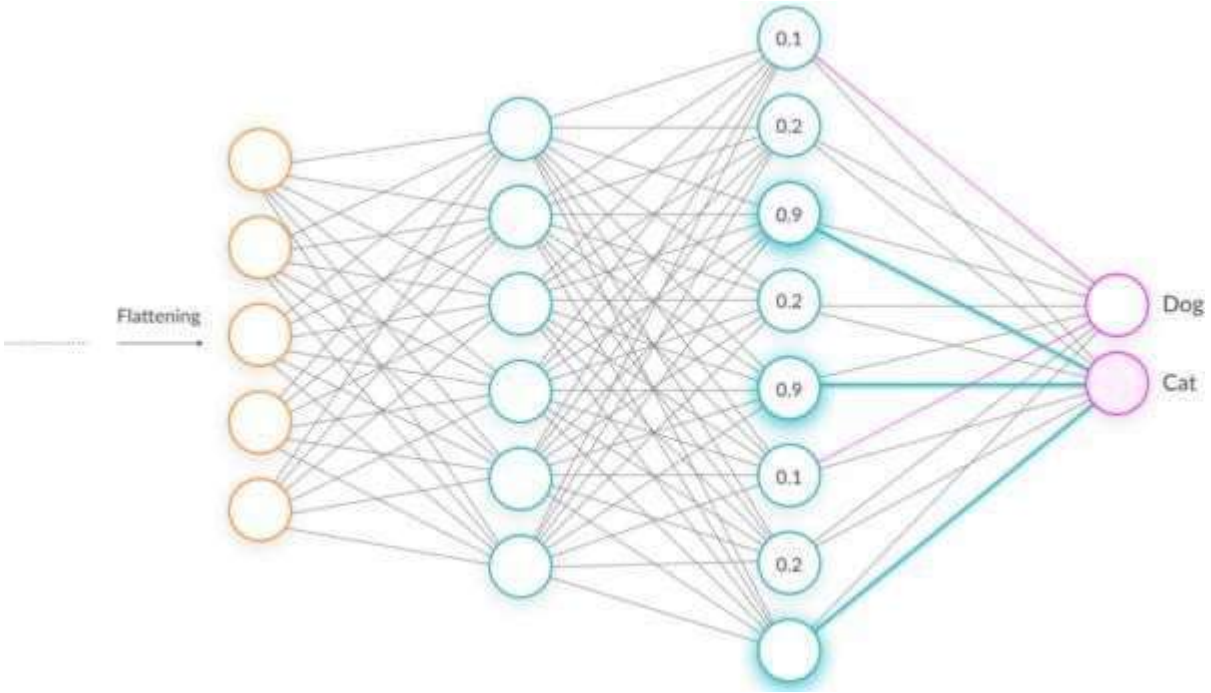
CNN Operations

❑ Fully Connected Layers

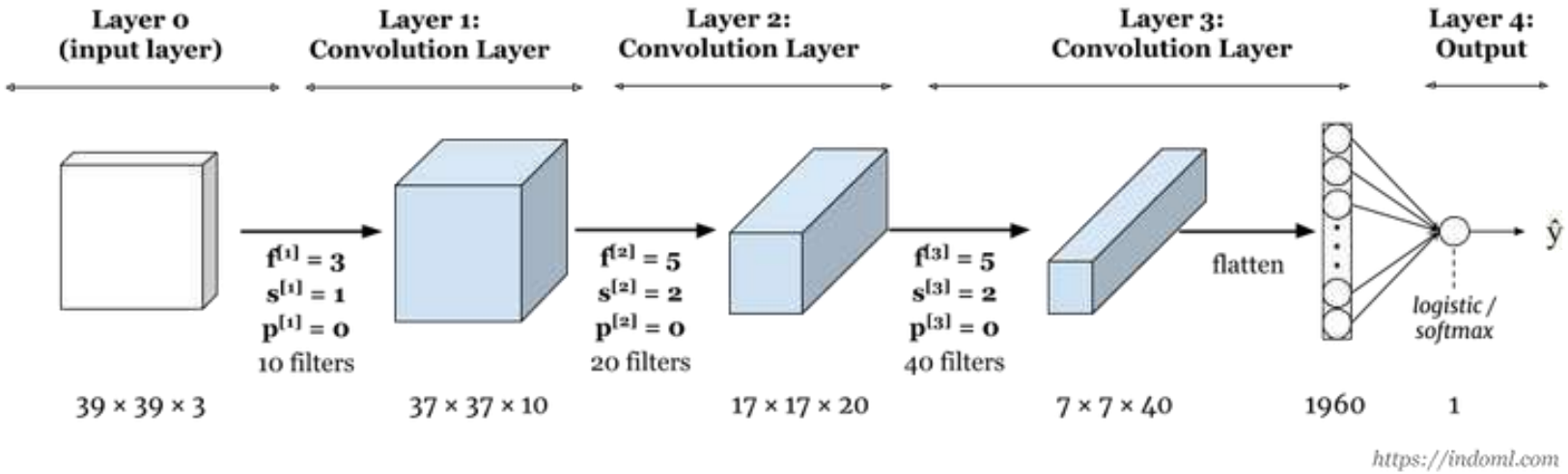
❑ After several layers of convolution and pooling operations are completed, now the final output is given to the fully connected layer.

❑ This is basically a neural network in which each neuron is connected to every other neuron in the previous layer.

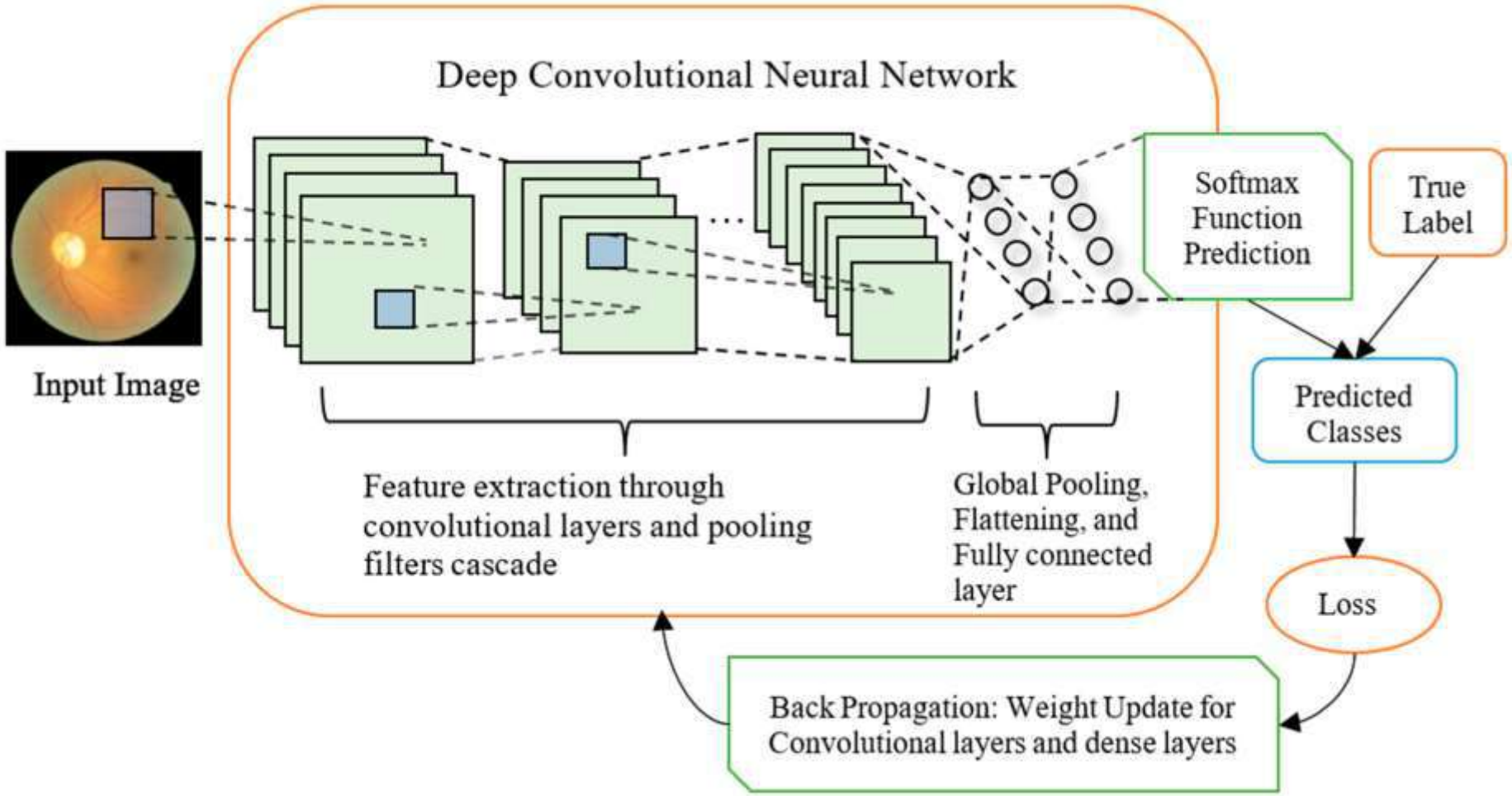
❑ All the recognition and classification parts are done by this neural network.



Sample Complete Network



Back propagation to Update Weight :



CNN Architecture

- **Designing a Convolutional Neural Network:**
- Now that we understand the various components, we can build a convolutional neural network.
- We will be using Fashion-MNIST, which is a dataset of Zalando‟s article images consisting of a training set of 60,000 examples and a test set of 10,000 examples.
- Each example is a 28x28 grayscale image, associated with a label from 10 classes.
- Our convolutional neural network has architecture as follows:

[INPUT]

→[CONV 1] → [BATCH NORM] → [ReLU] → [POOL 1]

→[CONV 2] → [BATCH NORM] → [ReLU] → [POOL 2]

→[FC LAYER] → [RESULT]

- For both conv layers, we will use kernel of spatial size 5 x 5 with stride size 1 and padding of 2.
- For both pooling layers, we will use max pool operation with kernel size 2, stride 2, and zero padding.

CNN Architecture

❑Designing a Convolutional Neural Network:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

CONV 1
Input Size (W_1 x H_1 x D_1) = 28 x 28 x 1
• Requires four hyperparameter: ○ Number of kernels, k = 16 ○ Spatial extend of each one, F = 5 ○ Stride Size, S = 1 ○ Amount of zero padding, P = 2
• Outputting volume of W_2 x H_2 x D_2 ○ W_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ H_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ D_2 = k
Output of Conv 1 (W_2 x H_2 x D_2) = 28 x 28 x 16

CNN Architecture

❑Designing a Convolutional Neural Network:

$$W_{out} = \frac{W - F}{S} + 1$$

POOL 1
Input Size (W_2 x H_2 x D_2) = 28 x 28 x 16
• Requires two hyperparameter: ○ Spatial extend of each one, F = 2 ○ Stride Size, S = 2
• Outputting volume of W_3 x H_3 x D_2 ○ W_3 = (28 – 2) / 2 + 1 = 14 ○ H_3 = (28 – 2) / 2 + 1 = 14
Output of Pool 1 (W_3 x H_3 x D_2) = 14 x 14 x 16

CNN Architecture

❑Designing a Convolutional Neural Network:

CONV 2
Input Size (W_3 x H_3 x D_2) = 14 x 14 x 16
• Requires four hyperparameter: o Number of kernels, k = 32 o Spatial extend of each one, F = 5 o Stride Size, S = 1 o Amount of zero padding, P = 2
• Outputting volume of W_4 x H_4 x D_3 o W_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o H_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o D_3 = k
Output of Conv 2 (W_4 x H_4 x D_3) = 14 x 14 x 32

CNN Architecture

❑Designing a Convolutional Neural Network:

POOL 2
Input Size (W_4 x H_4 x D_3) = 14 x 14 x 32
• Requires two hyperparameter: ○ Spatial extend of each one, F = 2 ○ Stride Size, S = 2
• Outputting volume of W_5 x H_5 x D_3 ○ W_5 = (14 – 2) / 2 + 1 = 7 ○ H_5 = (14 – 2) / 2 + 1 = 7
Output of Pool 2 (W_5 x H_5 x D_3) = 7 x 7 x 32

CNN Architecture

❑Designing a Convolutional Neural Network:

Convolutional Neural Networks (CNNs) use **convolution layers** to detect patterns by applying filters to input data, capturing spatial hierarchies. **Pooling layers** downsample feature maps, reducing dimensions while retaining essential information.

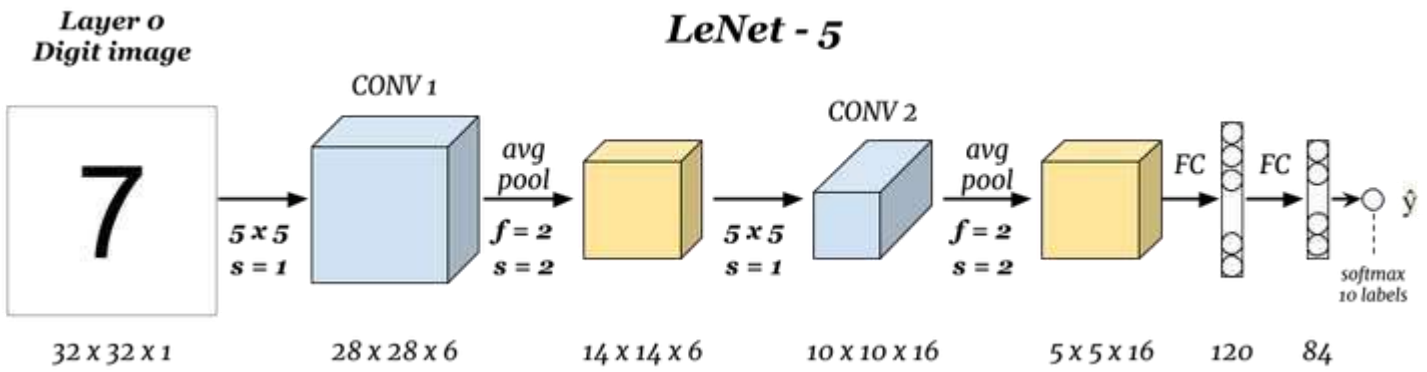
FC Layer
Input Size (W_5 x H_5 x D_3) = 7 x 7 x 32
Output Size (Number of Classes) = 10

Classic Network: LeNet – 5

- LeNet-5 is one of the earliest convolutional neural networks (CNNs), developed by **Yann LeCun** and his colleagues in 1989.
- It was primarily designed for **handwritten digit recognition** and played a significant role in the advancement of deep learning.
- The architecture of LeNet-5 is the foundation for modern CNNs used in image processing tasks.
- LeNet-5 consists of **7 layers** (including convolutional, pooling, and fully connected layers) with approximately **60,000** trainable parameters.
- It follows the **feature extraction and classification** approach.

Well Known Architectures

Classic Network: LeNet – 5



Document Recognition paper by Y. Lecun, L. Bottou, Y. Bengio and P. Haffner (1998)

Number of parameters: ~ 60 thousands.

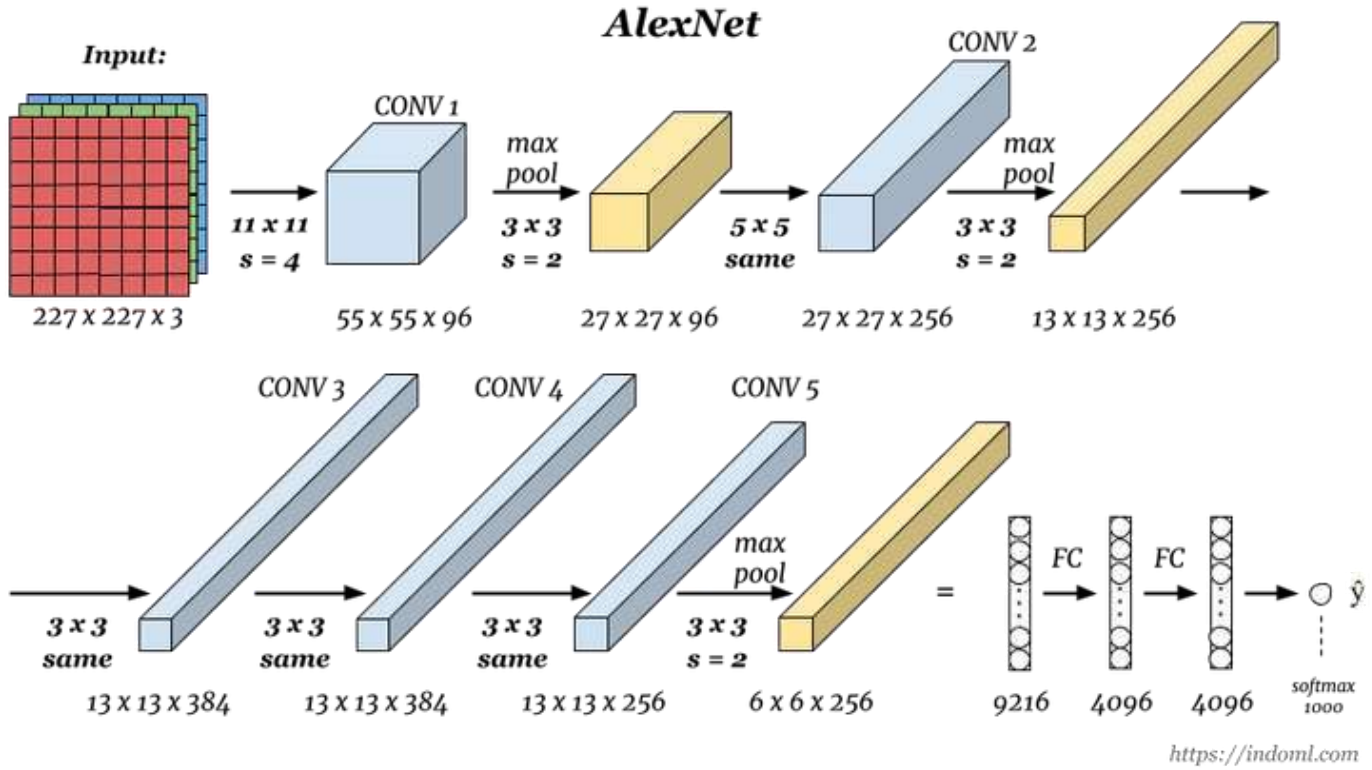
AlexNet

- AlexNet is a deep convolutional neural network (CNN) architecture that **revolutionized deep learning** by winning the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012**
- AlexNet is a deeper and wider version of LeNet-5, utilizing ReLU activations, dropout, and multiple GPUs to improve performance.
- AlexNet consists of 8 layers: 5 convolutional layers, 3 fully connected (FC) layers, followed by a Softmax classifier
- Each convolution layer consists of a convolution filter and a non-linear activation function called "ReLU".
- The pooling layers are used to perform the max-pooling function and the input size is fixed due to the presence of fully connected layers.
- The input size is mentioned at most of the places as 224x224x3 but due to some padding which happens it works out to be 227x227x3.
- Above all this AlexNet has over 60 million parameters.

AlexNet

- **Key Features:**
- „ReLU" is used as an activation function rather than „tanh"
- Batch size of 128
- SGD Momentum is used as a learning algorithm
- Data Augmentation is been carried out like flipping, jittering, cropping, colour normalization, etc.
- AlexNet was trained on a GTX 580 GPU with only 3 GB of memory which couldn"t fit the entire network.
- So the network was split across 2 GPUs, with half of the neurons(feature maps) on each GPU.

Classic Network: AlexNet



AlexNet is another classic CNN architecture from *ImageNet Classification with Deep Convolutional Neural Networks* paper by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever (2012).
Number of parameters: ~ 60 millions.

Classic Network: AlexNet

Feature	Details
Year	2012
Architecture	8 layers (5 Convolutional + 3 Fully Connected)
Activation	ReLU
Regularization	Dropout (50%)
Optimizer	SGD with Momentum
Batch Size	128
Pooling	Overlapping Max Pooling
GPU Usage	Yes (Split into 2 GPUs)
Dataset	ImageNet (1.2M images, 1000 classes)
Accuracy	80.2% (Top-5)

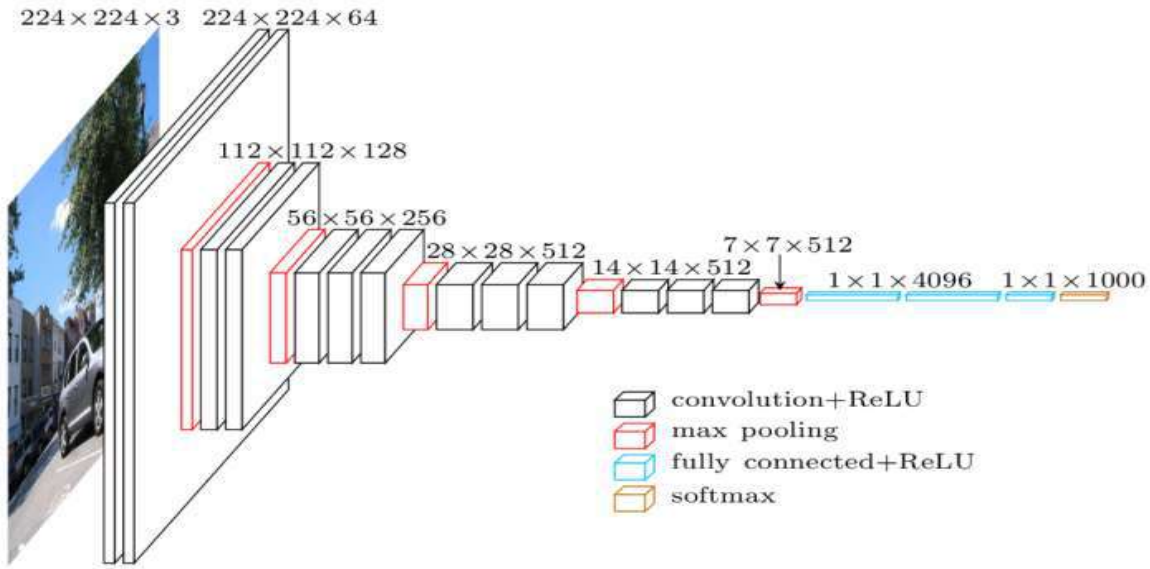
VGG Net

- VGG Net (Visual Geometry Group Network) is a deep convolutional neural network (CNN) architecture developed by the Visual Geometry Group at the University of Oxford (2014).
- The key innovation of VGG Net is its use of small 3×3 convolutional filters stacked together, which increases depth while maintaining computational efficiency.
- VGG Net comes in several variants, mainly **VGG-11, VGG-13, VGG-16, and VGG-19**, where the numbers denote the total layers in the network. The most widely used versions are **VGG-16** and **VGG-19**.
- VGG-16: 16 weight layers (13 convolutional layers + 3 fully connected layers)
- VGG-19: 19 weight layers (16 convolutional layers + 3 fully connected layers)

VGG Net

- **Key Features:**
- Small 3×3 Filters – Instead of using larger filters (like 5×5 or 7×7), VGGNet uses multiple 3×3 convolutional filters stacked on top of each other, capturing complex patterns.
- Deep Network – Increasing depth improves feature learning, making it highly effective for image recognition tasks.
- Max-Pooling Layers – Down-samples the feature maps, reducing spatial dimensions while preserving important features.
- Fully Connected Layers – The final layers are fully connected, ending in a Softmax layer for classification.
- ReLU Activation – The Rectified Linear Unit (ReLU) activation function is applied after each convolutional layer to introduce non-linearity.

Classic Network: VGG-16



VGG-16 from *Very Deep Convolutional Networks for Large-Scale Image Recognition* paper by Karen Simonyan and Andrew Zisserman (2014).
The number 16 refers to the fact that the network has 16 trainable layers (i.e. layers that have weights).

Number of parameters: ~ 138 millions

ResNet

- **ResNet (Residual Network)** is a deep convolutional neural network (CNN) architecture introduced by **Kaiming He et al.** in 2015
- It won the **ILSVRC 2015** (ImageNet Large Scale Visual Recognition Challenge) and revolutionized deep learning by **enabling extremely deep networks** without suffering from the **vanishing gradient problem**.
- Deep residual networks were a breakthrough idea which enabled the development of much deeper networks (hundreds of layers as opposed to tens of layers).
- Its a generally accepted principle that deeper networks are capable of learning more complex functions and representations of the input which should lead to better performance.
- However, many researchers observed that adding more layers eventually had a negative effect on the final performance.

Why ResNet?

- **Vanishing gradients**: As networks go deeper, gradients shrink, making weight updates very small, leading to poor learning.
- **Degradation problem**: Deeper networks sometimes performed **worse** than shallower ones, even though deeper networks should theoretically learn better.

The ResNet Solution: Residual Learning

- Instead of learning the desired function **H(x)** directly, ResNet learns the **residual function**:

$$F(x) = H(x) - x$$

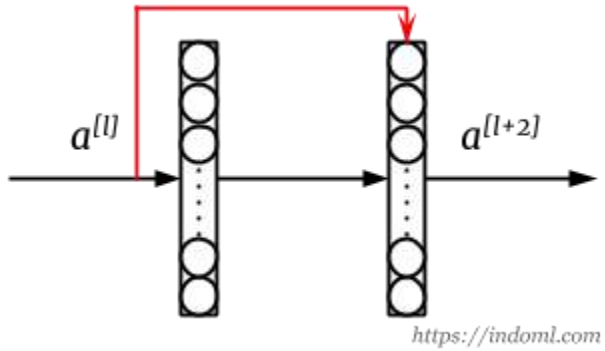
So, the final output is:

$$H(x) = F(x) + x$$

- This is done using **skip connections (shortcut connections)** that **bypass one or more layers**, allowing gradients to flow directly backward.

Key Features ResNet (Residual Network), proposed by He at all in *Deep Residual Learning for Image Recognition paper* (2015)

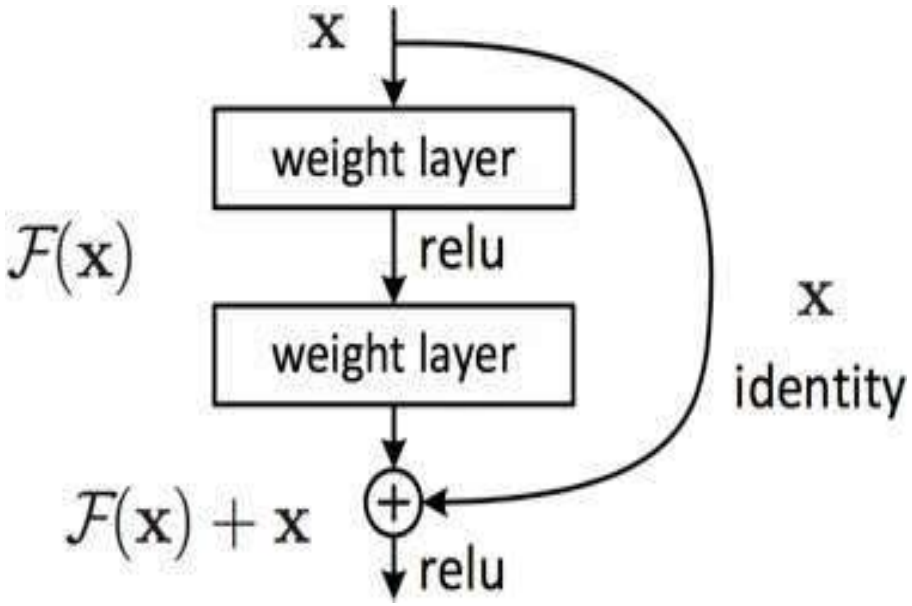
- Skip Connections (Residual Learning)
- Bottleneck Blocks for Efficiency
- Batch Normalization
- Global Average Pooling (GAP)



skip connection where output from one layer is fed to layer deeper in the network

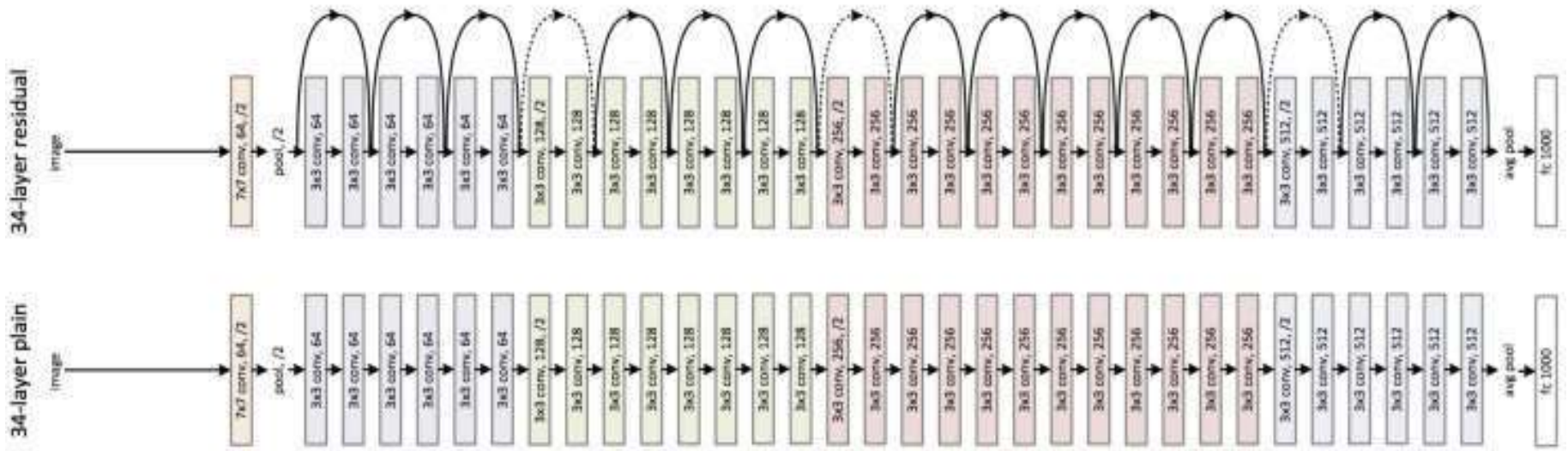
Feature	ResNet
Year	2015
Architecture	18, 34, 50, 101, 152 layers
Activation	ReLU
Regularization	Batch Normalization
Optimizer	SGD with Momentum
Batch Size	256
Pooling	Global Average Pooling
Skip Connections	Yes (Residual Learning)
Performance	State-of-the-art accuracy

ResNet



- Resnet Blocks: 'Skip Connection', identity mapping. This identity mapping has no additional parameters and is just there to add the output from the previous layer to the layer ahead.
- The identity mapping is multiplied by a linear projection W to expand the channels of the shortcut to match the residual. This allows for the input x and F(x) to be combined as input to the next layer.

Wide residual networks: Although the original ResNet focused on creating a network architecture to enable deeper structures by alleviating the degradation problem, other researchers have since pointed out that increasing the network's width (channel depth) can be a more efficient way of expanding the overall capacity of the network.



Dense Net in CNN

- **DenseNet (Densely Connected Convolutional Network)** is a deep learning architecture introduced by **Gao Huang et al.** in 2017

Why DenseNet

- **Vanishing gradient problem** in deep networks.
- **Redundant parameters** due to unnecessary feature maps.
- **Difficulty in feature reuse**—important features from earlier layers may not reach later layers efficiently.
- **Key Features**

  - **ResNet:** Uses **skip connections** that allow gradients to bypass certain layers.
  - **DenseNet:** Every layer is **directly connected** to every previous layer. Instead of summing outputs (like ResNet), DenseNet **concatenates feature maps**.

  - **In a Dense Block, each layer receives feature maps from all preceding layers:**

$$X_\ell = H_\ell([X_0, X_1, ..., X_{\ell-1}])$$

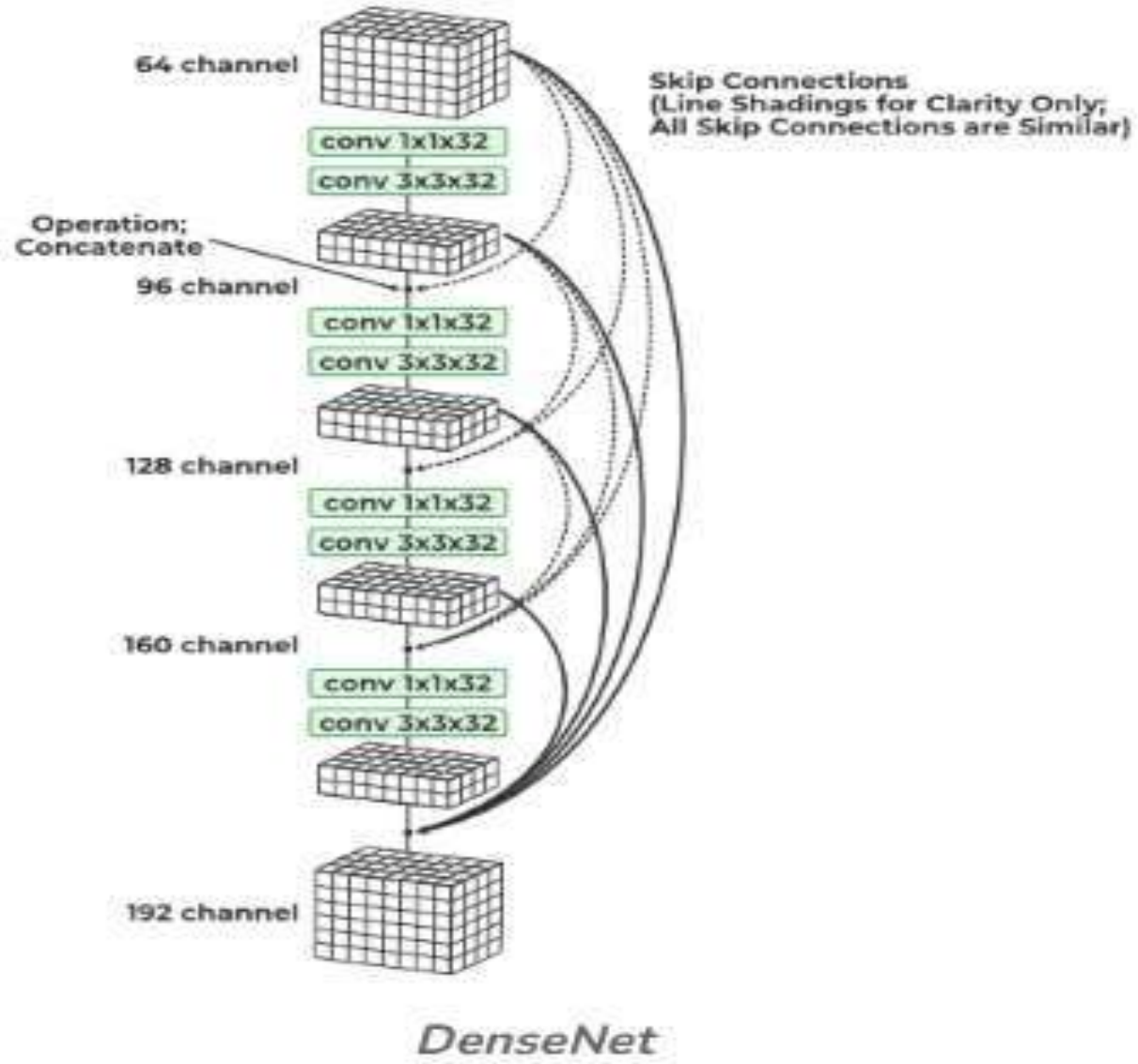
Where:

- X_ℓ = Output of the current layer ℓ.
- H_ℓ = Transformation function (BatchNorm → ReLU → Conv).
- $[X_0, X_1, ..., X_{\ell-1}]$ = **Concatenation** of all previous layers' outputs.

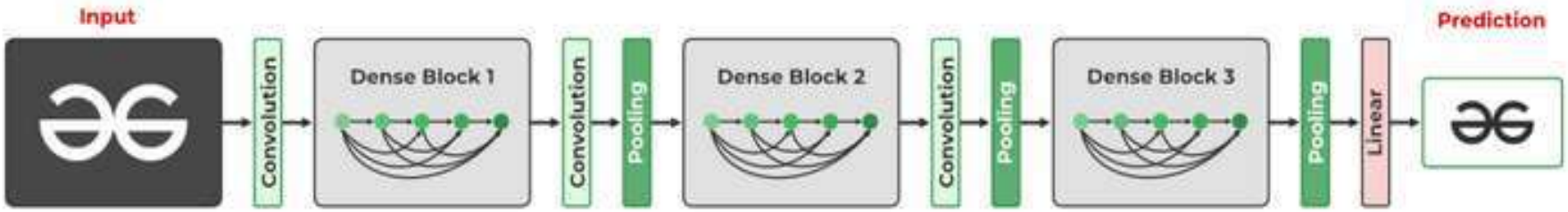
Dense Net in CNN

- The DenseNet model introduced the concept of a densely connected convolutional network, where the output of each layer is connected to the input of every subsequent layer.
- In simpler terms, due to the long distance between the input and output layer, the data is lost before it reaches its destination.
- The Dense Net model introduced the concept of a densely connected convolutional network, where the output of each layer is connected to the input of every subsequent layer.
- All convolutions in a dense block are ReLU-activated and use batch normalization. Channel-wise concatenation is only possible if the height and width dimensions of the data remain unchanged, so convolutions in a dense block are all of stride 1. Pooling layers are inserted between dense blocks for further dimensionality reduction.
- The resulting channels from each step of the DenseNet are concatenated to the collection of all previously generated outputs.
- Dense blocks and pooling layers are combined to form a Tu DenseNet network

Dense Net


64 channel
conv 1x1x32
conv 3x3x32
Operation;
Concatenate
96 channel
conv 1x1x32
conv 3x3x32
128 channel
conv 1x1x32
conv 3x3x32
160 channel
conv 1x1x32
conv 3x3x32
192 channel
Skip Connections
(Line Shadings for Clarity Only;
All Skip Connections are Similar)
DenseNet

Dense Net in CNN



DenseNet

InceptionNet (GoogleNet)

- Inception Net is a type of **Convolutional Neural Network (CNN)** architecture introduced by **Google** in the research paper **"Going Deeper with Convolutions" (2014).**
- It was primarily designed to **improve computational efficiency** and **enhance performance** by using **multiple filter sizes at each layer**, allowing the network to **capture information at different scales**.
- The first and most famous version of Inception Net is **GoogLeNet (Inception v1)**, which won the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014**.
- It was later discovered that the earliest auxiliary output had no discernible effect on the final quality of the network.
- The addition of auxiliary outputs primarily benefited the end performance of the model, converging at a slightly better value than the same network architecture without an auxiliary branch.
- It is believed the addition of auxiliary outputs had a regularizing effect on the network.
- A revised, deeper version of the Inception network which takes advantage of the more efficient Inception cells is shown in next page.

InceptionNet (GoogleNet)

- The model is comprised of a basic unit referred to as an "Inception cell" in which we perform a series of convolutions at different scales and subsequently aggregate the results.
- In order to save computation, 1x1 convolutions are used to reduce the input channel depth.
- For each cell, we learn a set of 1x1, 3x3, and 5x5 filters which can learn to extract features at different scales from the input.
- Max pooling is also used, albeit with "same" padding to preserve the dimensions so that the output can be properly concatenated

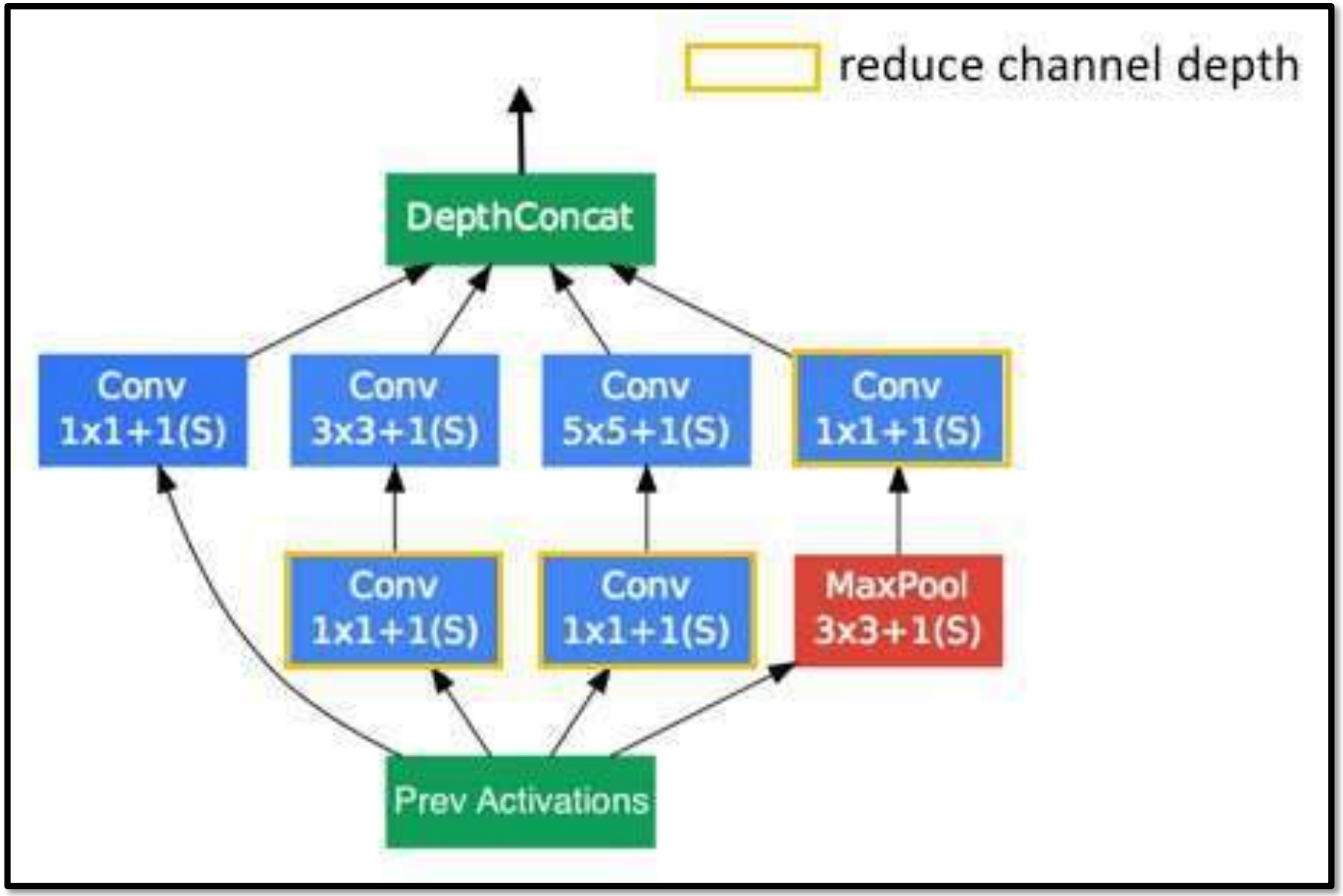
Traditional CNNs use sequential convolution layers which may not effectively capture features at multiple scales (small and large). Also, deeper networks face:

- **Computational inefficiency**.
- **Overfitting** due to large parameters.
- **Vanishing gradient problem**.

✅ **Inception Net** addresses these issues by:

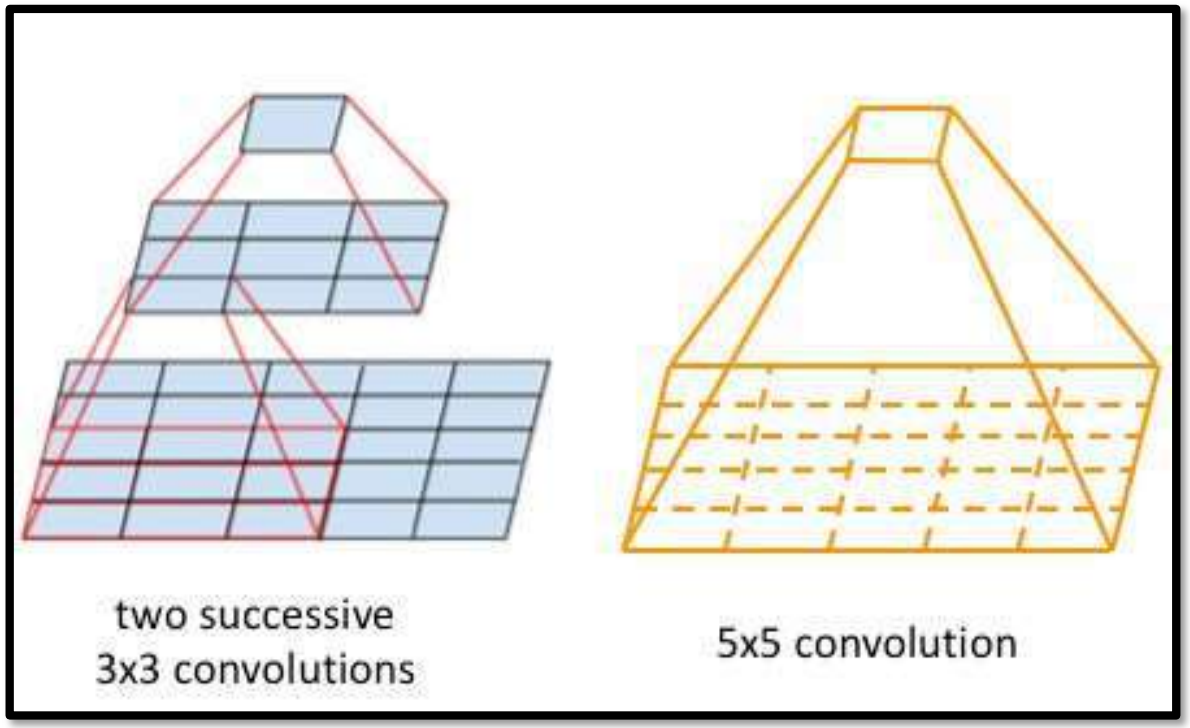
1. **Multi-scale feature extraction**.
2. **Reduced computational cost** using **1x1 convolutions** for dimensionality reduction.
3. **Efficient deeper architectures** without exploding parameters.

InceptionNet (GoogleNet)



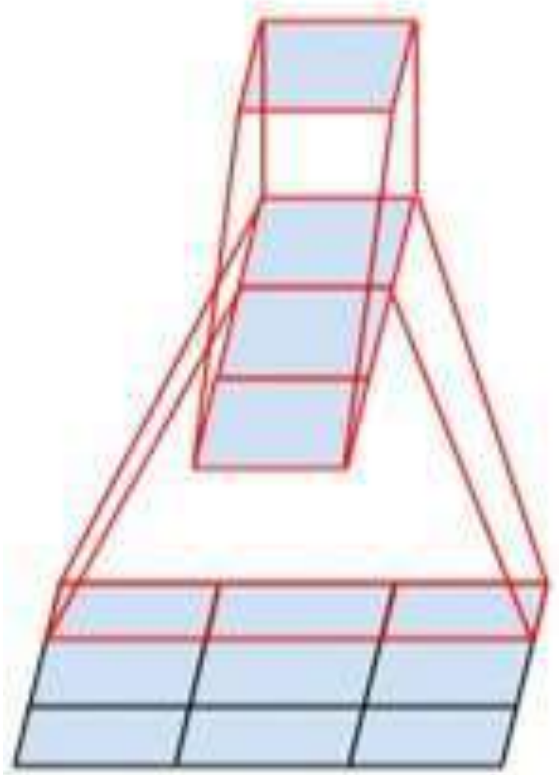
InceptionNet (GoogleNet)

- These researchers published a follow-up paper which introduced more efficient alternatives to the original Inception cell.
- Convolutions with large spatial filters (such as 5x5 or 7x7) are beneficial in terms of their expressiveness and ability to extract features at a larger scale, but the computation is disproportionately expensive.
- The researchers pointed out that a 5x5 convolution can be more cheaply represented by two stacked 3x3 filters.

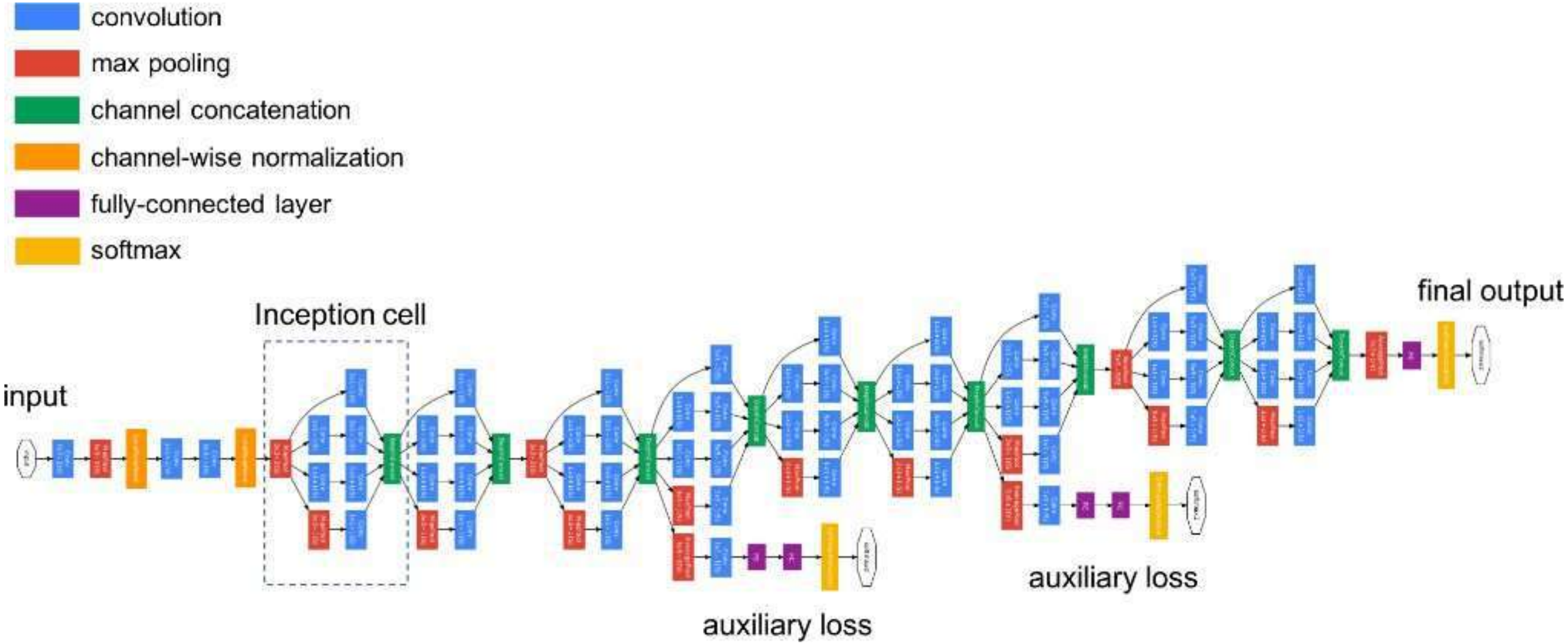


InceptionNet (GoogleNet)

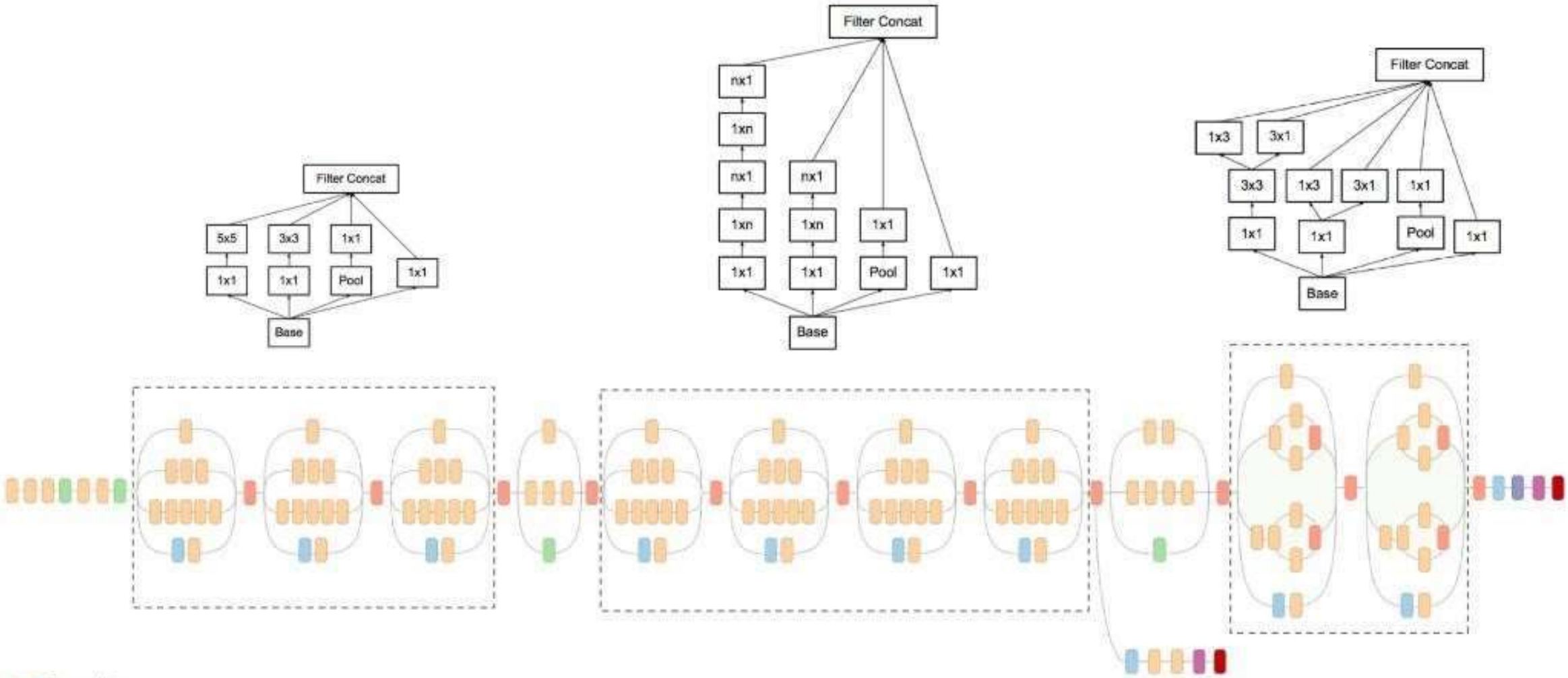
- Whereas a 5×5×c filter requires 25c parameters, two 3×3×c filters only require 18 parameters.
- In order to most accurately represent a 5x5 filter, we shouldn't use any nonlinear activations between the two 3x3 layers.
- However, it was discovered that "linear activation was always inferior to using rectified linear units in all stages of the factorization.“
- It was also shown that 3x3 convolutions could be further deconstructed into successive 3x1 and 1x3 convolutions.
- Generalizing this insight, we can more efficiently compute an n×n convolution as a 1×n convolution followed by a n×1 convolution.



InceptionNet (GoogleNet)

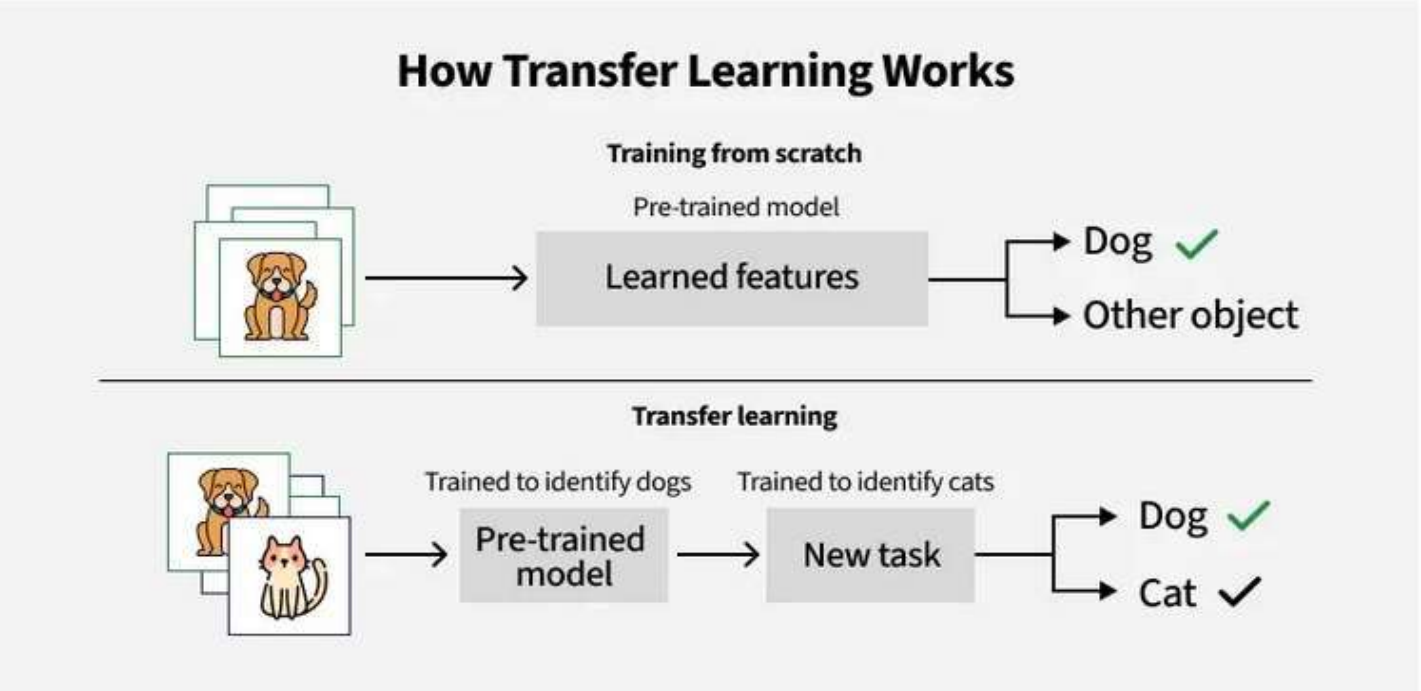


InceptionNet (GoogleNet)



Transfer Learning

- **Transfer learning** is a machine learning technique where a model trained on one task is repurposed as the foundation for a second task. This approach is beneficial when the second task is related to the first or when data for the second task is limited.
- Leveraging learned features from the initial task, the model can adapt more efficiently to the new task, accelerating learning and improving performance. Transfer learning also reduces the risk of overfitting, as the model already incorporates generalizable features useful for the second task. (Overcome challenges like: Limited data, **Enhanced Performance, Time and cost Efficiency, Adaptability)**



Transfer Learning

Transfer learning involves a structured process to leverage existing knowledge from a pre-trained model for new tasks:

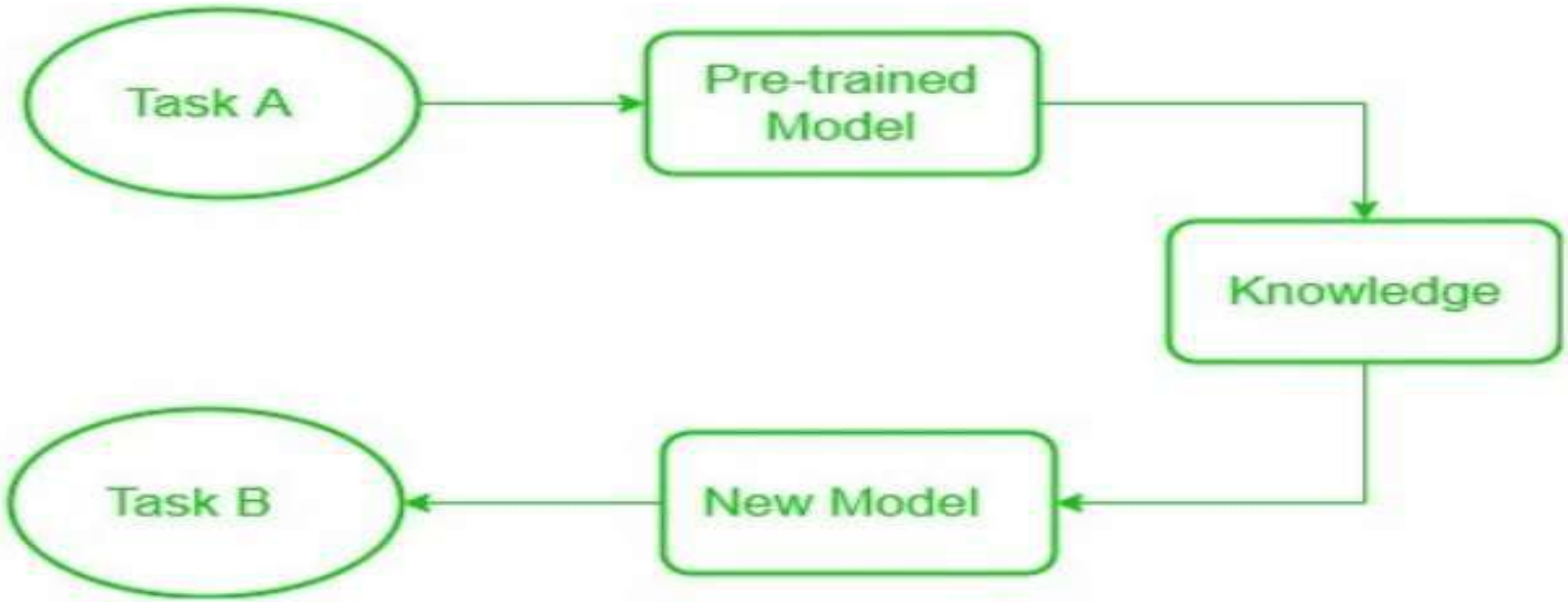
1.Pre-trained Model: Start with a model already trained on a large dataset for a specific task. This pre-trained model has learned general features and patterns that are relevant across related tasks.

2.Base Model: This pre-trained model, known as the base model, includes layers that have processed data to learn hierarchical representations, capturing low-level to complex features.

3.Transfer Layers: Identify layers within the base model that hold generic information applicable to both the original and new tasks. These layers, often near the top of the network, capture broad, reusable features.

4.Fine-tuning: Fine-tune these selected layers with data from the new task. This process helps retain the pre-trained knowledge while adjusting parameters to meet the specific requirements of the new task, improving accuracy and adaptability.

Transfer Learning



Transfer Learning

Thank You