TASK 2: Research Study on Deep Learning Models

1. Convolutional Neural Networks

The structure of CNNs was inspired by neurons in human and animal brains, similar to a conventional neural network.
A commonly used type of CNN, which is similar to the multi-layer perceptron (MLP), consists of numerous convolution layers preceding sub-sampling (pooling) layers, while the ending layers are FC layers. An example of CNN architecture for image classification is illustrated in Fig. 1

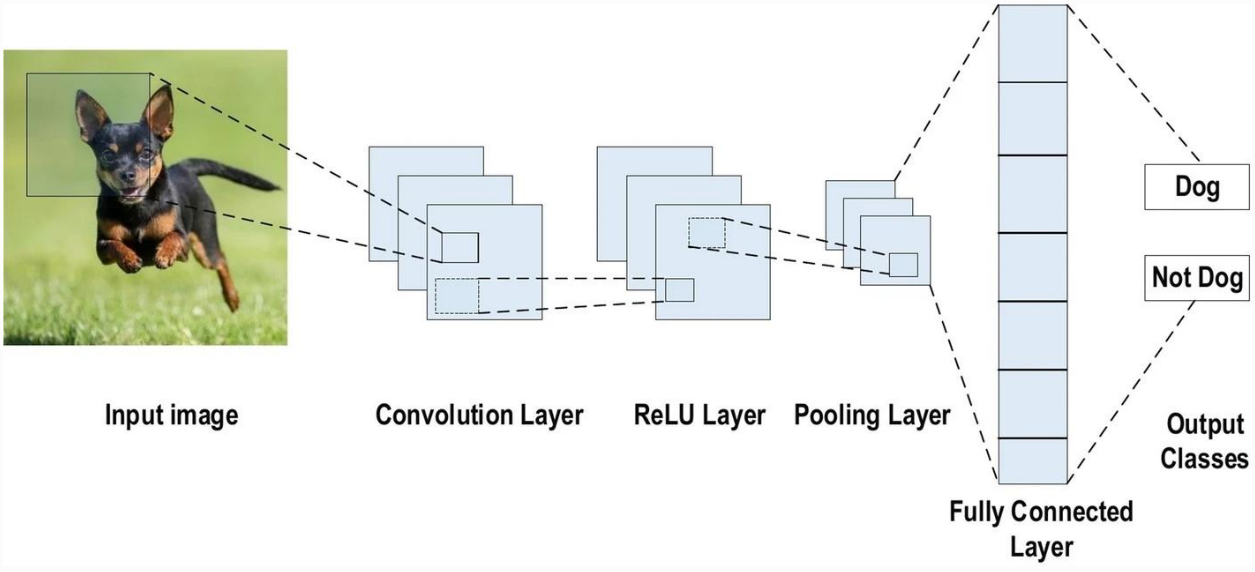
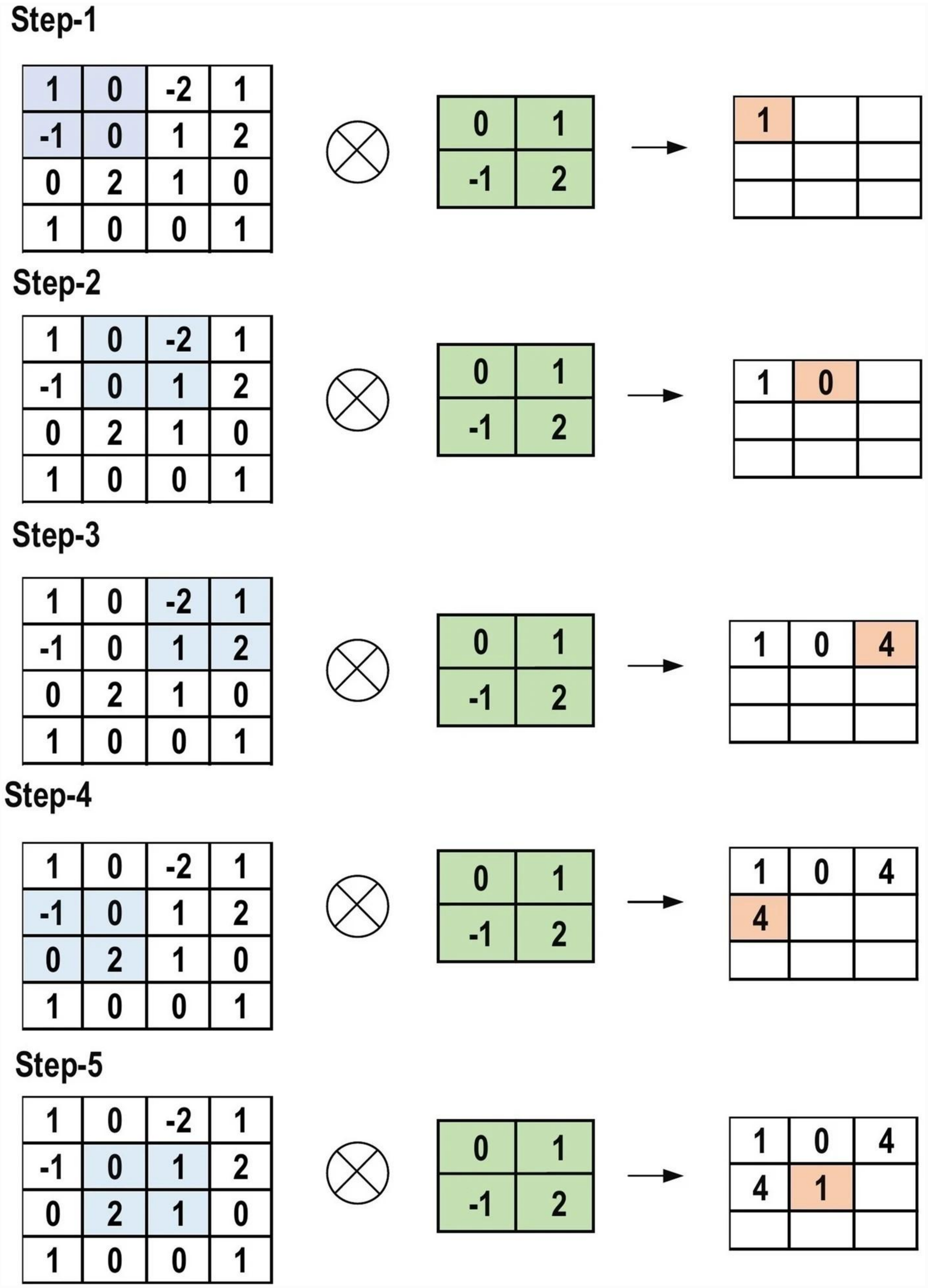


Fig1: An example of CNN architecture for image classification

CNN Layers:

1. **Convolutional Layer:** In CNN architecture, the most significant component is the convolutional layer. It consists of a collection of convolutional filters (so-called kernels). The input image, expressed as N-dimensional metrics, is convolved with these filters to generate the output feature map.To understand the convolutional operation, let us take an example of a 4×4 gray-scale image with a 2×2 random weight-initialized kernel.
   The process can be understand by the image below:



Step-1

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

$\otimes$

0	1
-1	2

→

1		

Step-2

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

$\otimes$

0	1
-1	2

→

1	0	

Step-3

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

$\otimes$

0	1
-1	2

→

1	0	4

Step-4

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

$\otimes$

0	1
-1	2

→

1	0	4
4		

Step-5

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

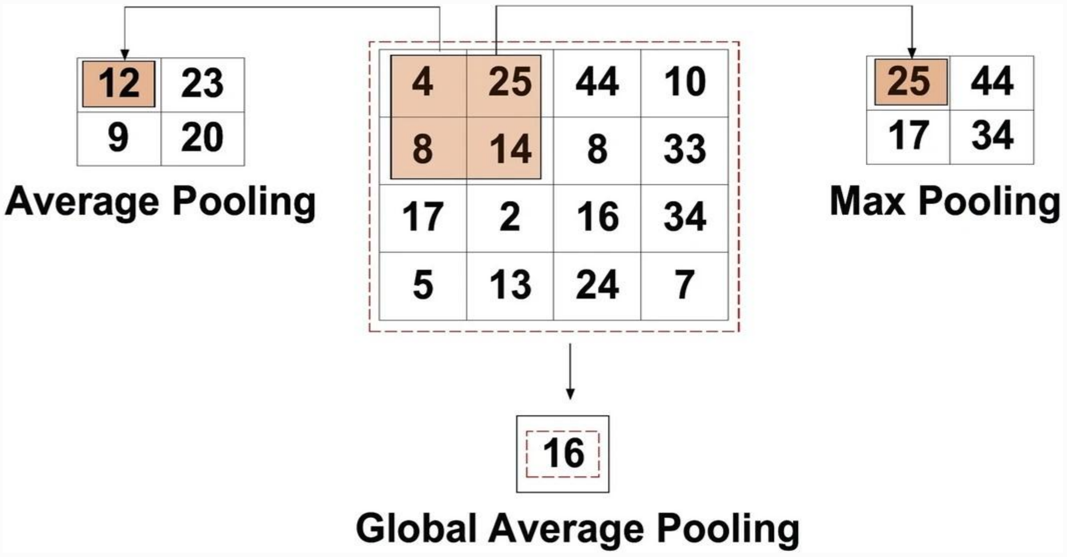
$\otimes$

0	1
-1	2

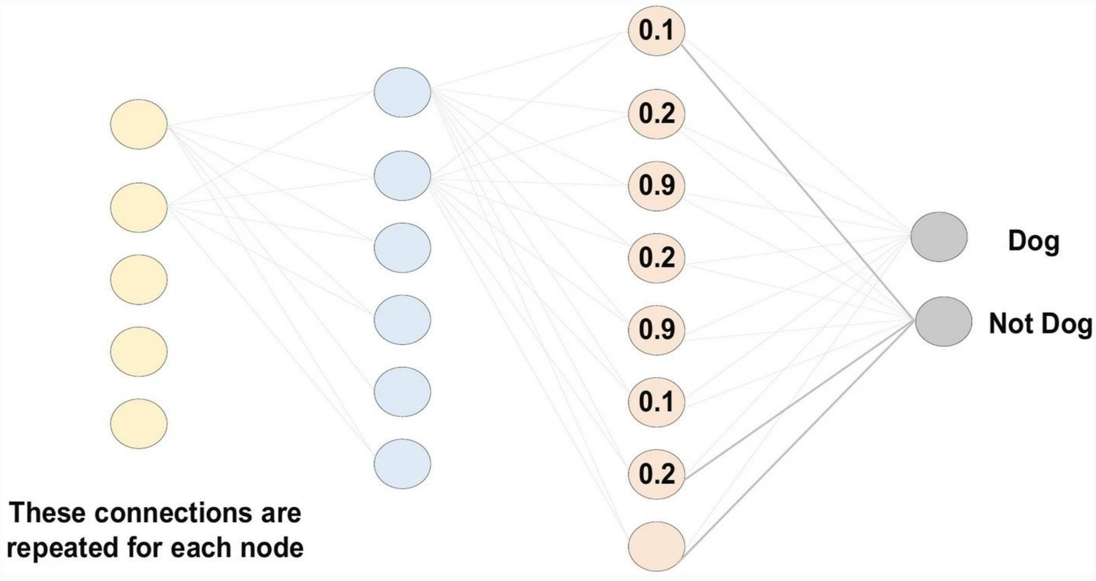
→

1	0	4
4	1	

2. **Pooling Layer:** The main task of the pooling layer is the sub-sampling of the feature maps. These maps are generated by following the convolutional operations. In other words, this approach shrinks large-size feature maps to create smaller feature maps. Several types of pooling methods are available for utilization in various pooling layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling.



3. **Fully Connected Layer:** Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach. The input of the FC layer comes from the last pooling or convolutional layer. This input is in the form of a vector, which is created from the feature maps after flattening.



Benefits of employing CNNs

The benefits of using CNNs over other traditional neural networks in the computer vision environment are listed as follows:

1. The main reason to consider CNN is the weight sharing feature, which reduces the number of trainable network parameters and in turn helps the network to enhance generalization and to avoid overfitting.

2. Concurrently learning the feature extraction layers and the classification layer causes the model output to be both highly organized and highly reliant on the extracted features.

3. Large-scale network implementation is much easier with CNN than with other neural networks.

2. ResNet

Residual Network (ResNet) is a deep learning model used for computer vision applications. It is a Convolutional Neural Network (CNN) architecture designed to support hundreds or thousands of convolutional layers.
ResNet provides an innovative solution to the vanishing gradient problem, known as "skip connections". ResNet stacks multiple identity mappings (convolutional layers that do nothing at first), skips those layers, and reuses the activations of the previous layer. Skipping speeds up initial training by compressing the network into fewer layers.

Then, when the network is retrained, all layers are expanded and the remaining parts of the network—known as the residual parts—are allowed to explore more of the feature space of the input image.

Most ResNet models skip two or three layers at a time with nonlinearity and batch normalization in between. More advanced ResNet architectures, known as HighwayNets, can learn "skip weights", which dynamically determine the number of layers to skip.

What Is a Residual Block?

Residual blocks are an important part of the ResNet architecture. In older architectures such as VGG16, convolutional layers are stacked with batch normalization and nonlinear activation layers such as ReLu between them. This method works with a small number of convolutional layers—the maximum for VGG models is around 19 layers. However, subsequent research discovered that increasing the number of layers could significantly improve CNN performance.

The ResNet architecture introduces the simple concept of adding an intermediate input to the output of a series of convolution blocks. This is illustrated below.

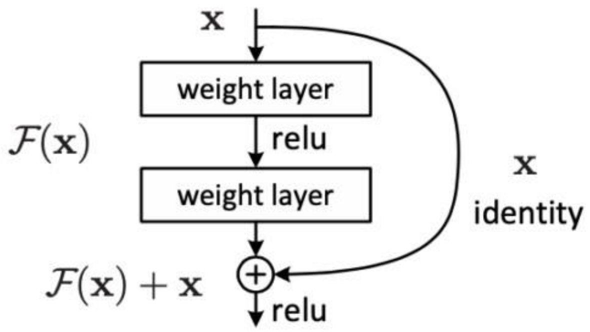
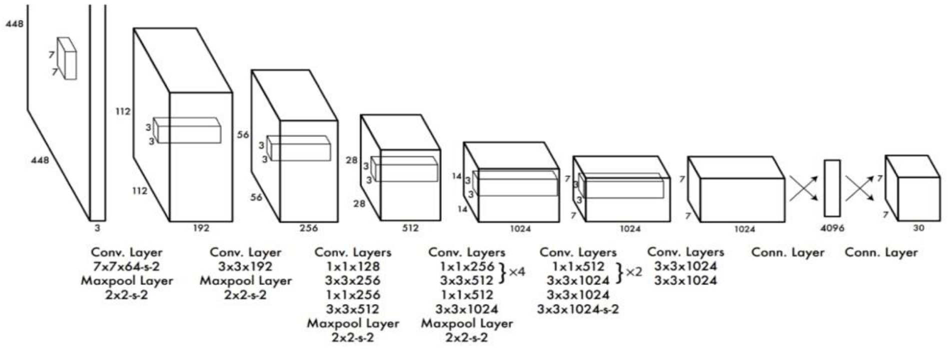

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

3. YOLO (You Only Look Once)

YOLO (You Only Look Once) is an advanced algorithm that can detect and recognize various objects in a picture in real-time, faster and more accurately than any other algorithm.
The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

The architecture works as follows:

- Resizes the input image into 448x448 before going through the convolutional network.

- A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
- The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
- Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting

The different versions of YOLO are as follows:

YOLOv1

This first version of YOLO was a game changer for object detection, because of its ability to quickly and efficiently recognize objects.

However, like many other solutions, the first version of YOLO has its own limitations:

- Struggles to detect small objects.
- Unable to detect new or unusual shapes.
- Incorrect localisation due to bad design of loss function

YOLOv2 or YOLO9000

YOLOv2 was created in 2016 with the idea of making the YOLO model better, faster and stronger. The improvement includes but is not limited to the use of Darknet-19 as new architecture, batch normalization, higher resolution of inputs, convolution layers with anchors, dimensionality clustering, and fine-grained features.

YOLOv3—An incremental improvement

Here the change mainly includes a new network architecture: Darknet-53. This is a 106 neural network, with upsampling networks and residual blocks. It is much bigger, faster, and more accurate compared to Darknet-19 which is the backbone of YOLOv2.

YOLOv4—Optimal Speed and Accuracy of Object Detection

The backbone of YOLOv4's architecture is CSPDarket53, a network containing 29 convolution layers with 3 × 3 filters. This architecture, compared to YOLOv3, adds the following information for better object detection:

- Spatial Pyramid Pooling (SPP)
- YOLOv4 uses PANet for parameter aggregation from different detection levels.
- Data augmentation uses the mosaic technique that combines four training images.
- Perform optimal hyper-parameter selection using genetic algorithms.

YOLOv5

YOLOv5, similarly to YOLOv4, uses CSPDarknet53 as the backbone of its architecture. The release includes five different model sizes: YOLOv5s (smallest), YOLOv5m, YOLOv5l, and YOLOv5x (largest). One of the major improvements in YOLOv5 architecture is the integration of theFocus Layer, represented by a single layer, which is created by replacing the first three layers of YOLOv3.

YOLOv6—A Single-Stage Object Detection Framework for Industrial Applications

YOLOv6 introduced three significant improvements to the previous YOLOv5: a hardware-friendly backbone and neck design, an efficient decoupled head, and a more effective training strategy.

YOLOv7

This version is making a significant move in the field of object detection, and it surpassed all the previous models in terms of accuracy and speed. YOLOv7 has made a major change in its (1) architecture and (2) at the Trainable bag-of-freebies level:

Architectural level: YOLOv7 reformed its architecture by integrating the Extended Efficient Layer Aggregation Network (E-ELAN) which allows the model to learn more diverse features for better learning.

Trainable bag-of-freebies: The term **bag-of-freebies** refers to improving the model's accuracy without increasing the training cost, and this is the reason why YOLOv7 increased not only the inference speed but also the detection accuracy.

YOLOv8

The latest YOLOv8 implementation comes with a lot of new features, we especially like the user-friendly CLI and GitHub repo.

It supports object detection, instance segmentation, and image classification.

YOLOv8's high accuracy and performance make it a strong contender for your next computer vision project.