Task 2

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 key part of a CNN is its convolutional layers, where filters are used to extract characteristics like edges, textures, and forms from the input image. The output of the convolutional layers is then transmitted via pooling layers, which are used to down-sample the feature maps and save the most crucial information while lowering the spatial dimensions. One or more fully connected layers are then applied to the output of the pooling layers in order to forecast or categorize the image.

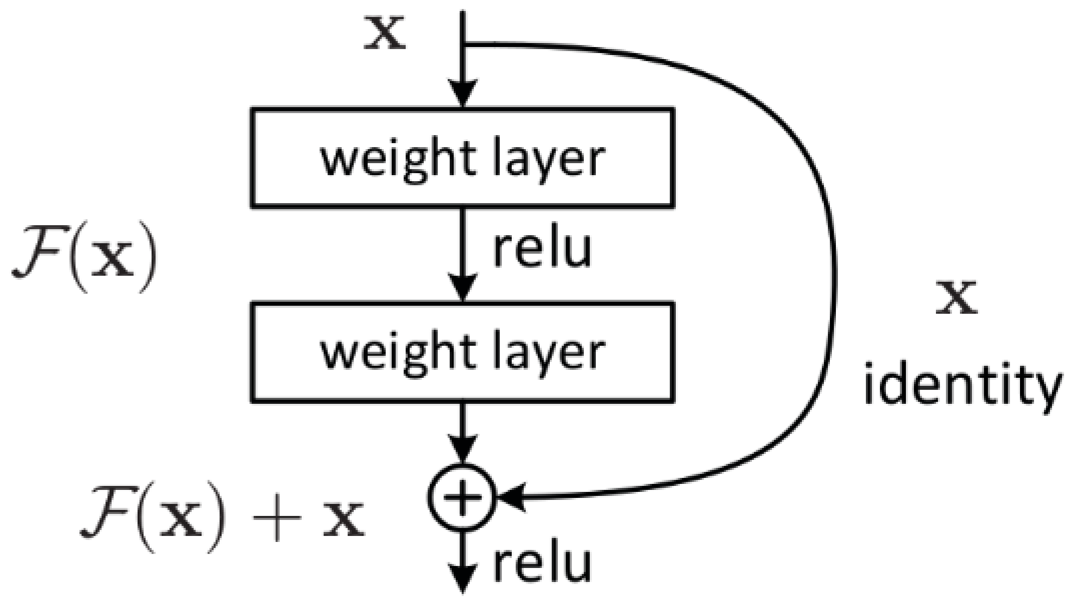
Layers

Input Layer: The input layer receives the raw input image data. Images are typically represented as multidimensional arrays of pixel values, with each dimension corresponding to the width, height, and color channels (e.g., red, green, blue). The input layer accepts these arrays as input.
Convolutional Layers: Convolutional layers are the core building blocks of CNNs. They consist of multiple filters, also known as kernels or feature detectors, which slide across the input image to perform convolution operations. Each filter applies a set of weights to the local receptive field of the input, producing a feature map. Convolutional layers learn to detect various patterns and features in the input images, such as edges, textures, or higher-level representations.
Activation Function: After the convolution operation, an activation function is applied element-wise to introduce non-linearities into the network. Common activation functions used in CNNs include ReLU (Rectified Linear Unit), which sets negative values to zero and keeps positive values unchanged. Activation functions introduce non-linearities, enabling the network to learn complex relationships between the input and output.
Pooling Layers: Pooling layers downsample the feature maps by reducing their spatial dimensions while preserving the most important information. Popular pooling operations include max pooling, where the maximum value within each pooling window is retained, and

average pooling, which calculates the average value within each window. Pooling layers help reduce the computational complexity of the network, introduce spatial invariance, and improve the network's ability to generalize.
Fully Connected Layers: Fully connected layers, also known as dense layers, are typically

added towards the end of the network. These layers connect every neuron from the previous layer to every neuron in the current layer. Fully connected layers capture high-level abstractions and global dependencies in the data. They perform non-linear transformations on the feature maps extracted by the earlier layers and produce the final output of the network.
Output Layer: The output layer produces the final predictions or outputs of the network. The number of neurons in the output layer depends on the specific task. For example, in image classification, the output layer may have neurons corresponding to different classes, with each neuron representing the probability of the input image belonging to a particular class. The activation function used in the output layer depends on the task requirements, such as softmax for multi-class classification.

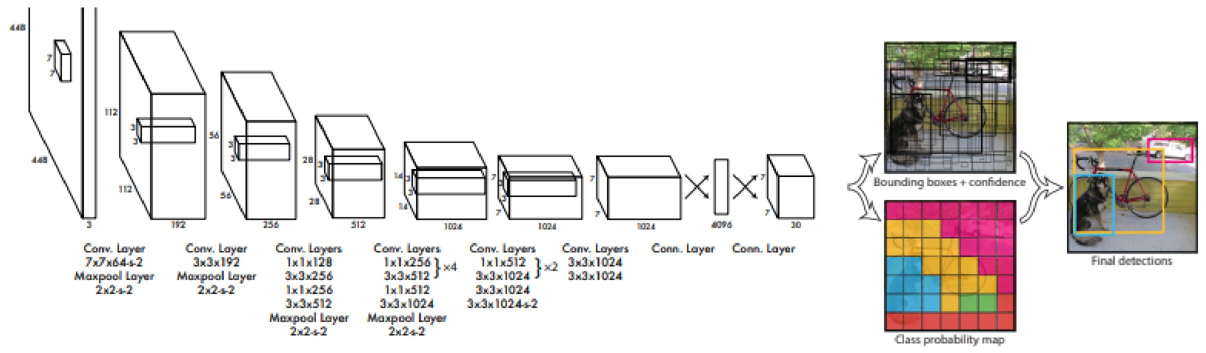
ResNet



ResNet-50 has an architecture based on the model depicted above, but with one important difference. The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a “bottleneck”, which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.ResNet solves the Vanishing Gradient Problem.When the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed.

YOLO

YOLOv1 unified the object detection steps by detecting all the bounding boxes simultaneously. To accomplish this, YOLO divides the input image into a S × S grid and predicts B bounding boxes of the same class, along with its confidence for C different classes per grid element. Each bounding box prediction consists of five values: P c, bx, by, bh, bw where P c is the confidence score for the box that reflects how confident the model is that the box contains an object and how accurate the box is. The bx and by coordinates are the centers of the box relative to the grid cell, and bh and bw are the height and width of the box relative to the full image. The output of YOLO is a tensor of S × S × (B × 5+ C) optionally followed by non-maximum suppression (NMS) to remove duplicate detections.


448
448
3
192
256
512
1024
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
Bounding boxes + confidence
Class probability map
Final detections