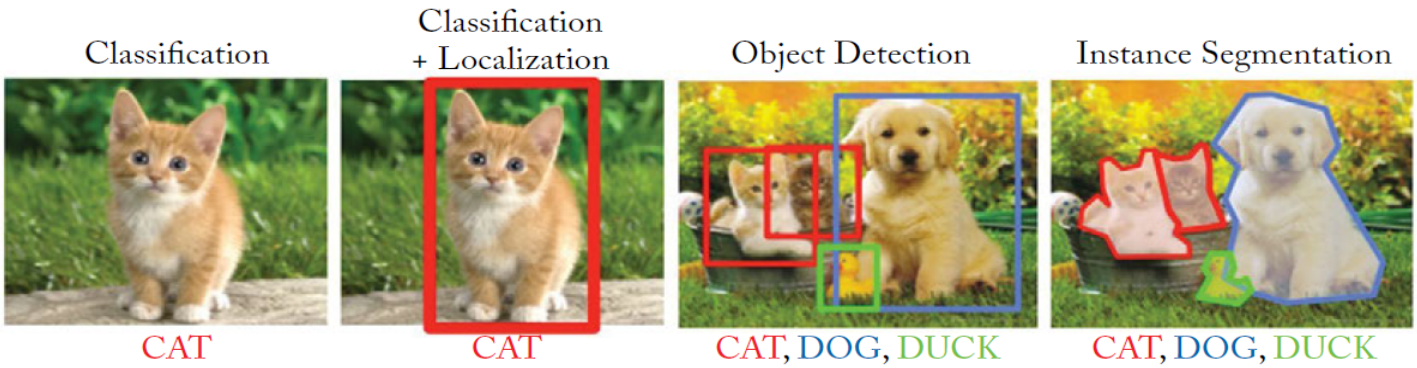
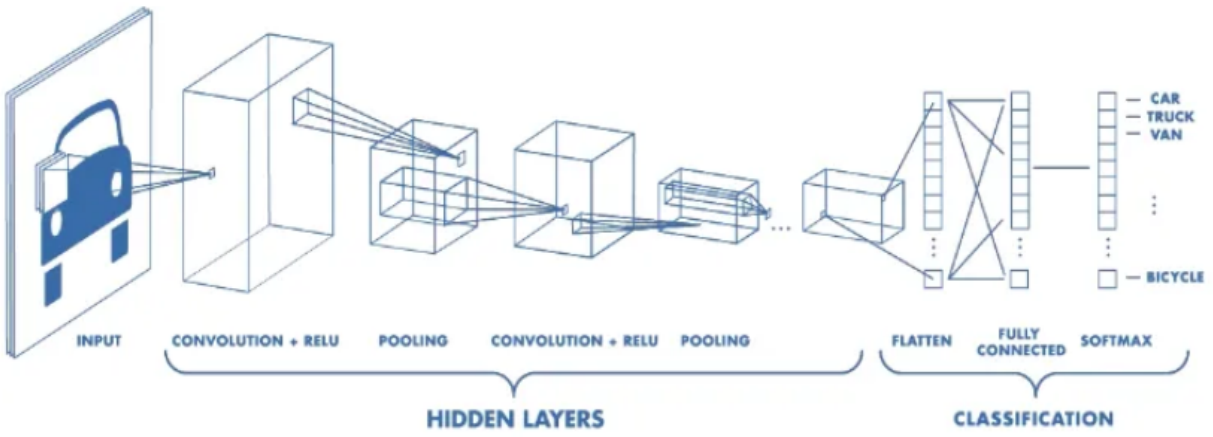
CONVOLUTIONAL NEURAL NETWORK

A **convolutional neural network** (CNN) is one of the most popular types of deep learning algorithms. It works well on images.



A CNN is an excellent tool for (i) image classification (categorization of image: cat/dog/horse), (ii) classification and localization of an object in the image (drawing a **bounding box** around an object and naming the class); (iii) object detection (localization and labeling of all objects present in the image); and (iv) instance segmentation (segmenting individual objects present in the picture).

Historical Note. The earliest form of CNN was the Neocognitron model proposed by Kunihiko Fukushima (Fukushima and Miyake, 1982). The Neocognitron was motivated by the seminal work by David Hubel and Torsten Wiesel (1959) which demonstrated that the neurons in the brain are organized in the form of layers. These layers learn to recognize visual patterns by first extracting local features and subsequently combining them to obtain a higher-level representations.



The architecture of CNN comprises several layers: a **convolution** layer, a **pooling** layer, and a **fully connected** layer.

Convolution Layer

An image of size $r \times c$ pixels is represented by three matrices of size $r \times c$ each, containing the intensity values of red, green, and blue primary colors (numbers between 0 and 255) on the RGB scale. The convolution layer performs a **dot product** between two matrices, where one matrix is the set of **learnable parameters** (otherwise known as a **kernel** or **filter** or **feature detector**), and the other matrix is the restricted portion of the image.

Example. Mathematically, the kernel is a matrix of weights. For example, the following 3×3 kernel detects vertical lines.

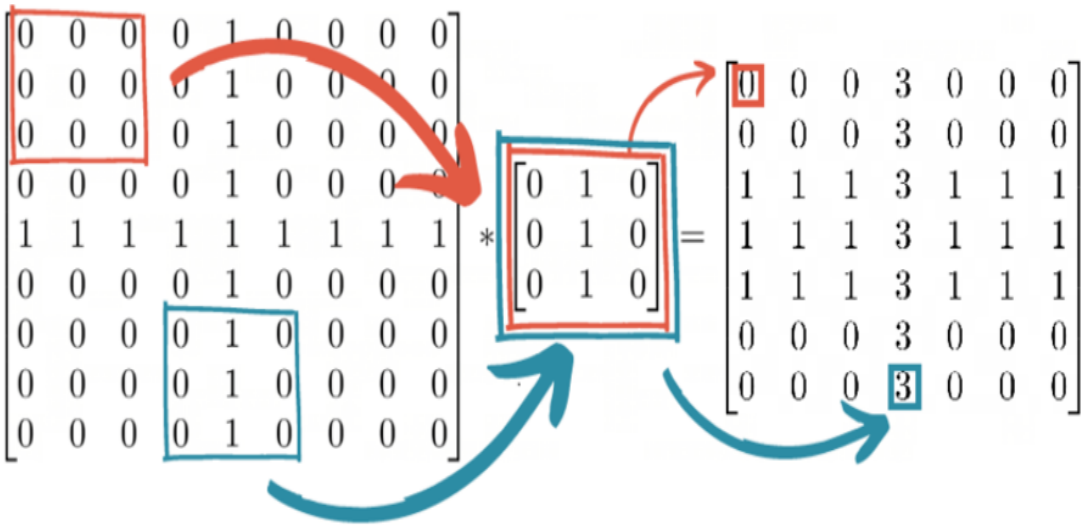
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Suppose we have an 9×9 input image of a plus sign. This has two kinds of lines, horizontal and vertical, and a crossover. In matrix format, the image would look as follows:

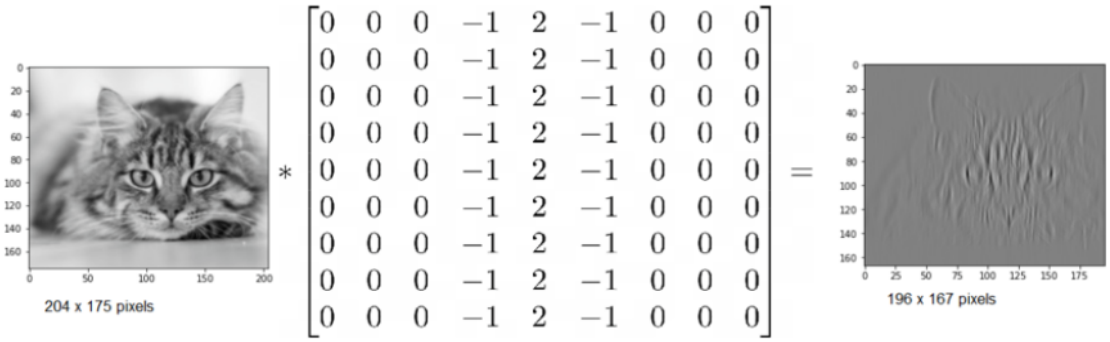
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Suppose we want to test the vertical line detector kernel on the plus sign image. To perform the convolution, we slide the convolution kernel over the image. At each position, we multiply each element of the convolution kernel by the element of the image that it covers and sum the results.

Since the kernel has width 3, it can only be positioned at 7 different positions horizontally in an image of width 9. So the end result of the convolution operation on an image of size 9×9 with a 3×3 convolution kernel is a new image of size 7×7.



To see how it works on an image, we consider a grayscaled image of a tabby cat with dimensions 204×175 pixels, which we can be represented by a matrix with values in the range between 0 and 1, where 1 is white and 0 is black.



Applying the convolution with a sophisticated vertical line detector, a 9×9 convolution kernel, we see that the filter has performed a kind of vertical line detection. The vertical stripes on the tabby cat's head are highlighted in the output. The output image is 8 pixels smaller in both dimensions

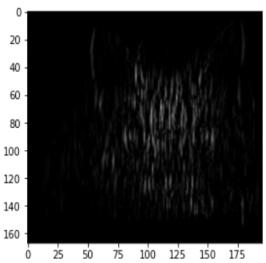
due to the size of the kernel (9×9).

A convolution layer is made up of a series of convolution kernels: a vertical line detector, a horizontal line detector, and various diagonal, border, curve, and corner detectors. These feature detector kernels serve as the first stage of the image recognition process.

Later layers in the neural network can build on the features detected by earlier layers and identify ever more complex shapes.

Activation Function

After passing an image through a convolutional layer, the output is normally passed through an activation function. A common activation function is called **rectified linear unit** (ReLU) defined as $f(x) = \max(0, x)$. The activation function has the effect of adding non-linearity into the convolutional neural network. If the activation function was not present, all the layers of the neural network could be condensed down to a single matrix multiplication. In the case of the cat image above, applying a ReLU function to the first layer output results in a stronger contrast highlighting the vertical lines, and removing the noise originating from other non-vertical features.



Repeating Structure of a CNN

A CNN can be viewed as a series of convolutional layers, followed by an activation function, followed by a **pooling** (**downscaling**) layer, repeated many times.

With the repeated combination of these operations, the first layer detects simple features such as edges in an image, and the second layer begins to detect higher-level features. By the tenth layer, a convolutional neural network can detect more complex shapes such as eyes. By the twentieth layer, it is often able to differentiate faces from one another.

This power comes from the repeated layering of operations, each of which can detect slightly higher-order features than its predecessor.

Once the image is processed through the convolution and pooling layers, it goes into the classification stage consisting of a flatten layer, a connected layer, and a softmax layer. The **flatten** layer collapses the spatial dimensions of the input into a single channel dimension. **Fully connected** layers connect every neuron in one layer to every neuron in another layer. The flattened matrix goes through a fully connected layer to classify the images.

Softmax Function

The **softmax function** is a normalized exponential function that takes as input a vector $(z_1, \ldots, z_K)$ of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, before applying softmax, some vector components could be negative, or greater than one, and might not sum to 1, but after applying softmax, each component will be in the interval $(0, 1)$, and the components will add up to 1, so that they can be interpreted as probabilities. We have

$$softmax(z_1, \ldots, z_K) = \Big(\frac{e^{z_1}}{\sum_{j=1}^{K} e^{z_j}}, \ldots, \frac{e^{z_K}}{\sum_{j=1}^{K} e^{z_j}}\Big).$$

Note that $\max(z_1, \ldots, z_K)$ is the "hard" maximum because it picks a single value, ignoring the rest of the data. Softmax, on the other hand, chooses the maximum value, making use of the entire dataset.