1. **Convolutional Neural Network (CNN)**

🡪A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

🡪**Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces**. It is this property that makes convolutional neural networks so powerful for computer vision. **Unlike earlier computer vision algorithms, convolutional neural networks can operate directly on a raw image and do not need any preprocessing**.

🡪A convolutional neural network is a feed-forward neural network, often with up to 20 or 30 layers**. The power of a convolutional neural network comes from a special kind of layer called the convolutional layer**.

🡪Convolutional neural networks contain many convolutional layers stacked on top of each other, each one capable of recognizing more sophisticated shapes. **With three or four convolutional layers it is possible to recognize handwritten digits** and **with 25 layers it is possible to distinguish human faces**.

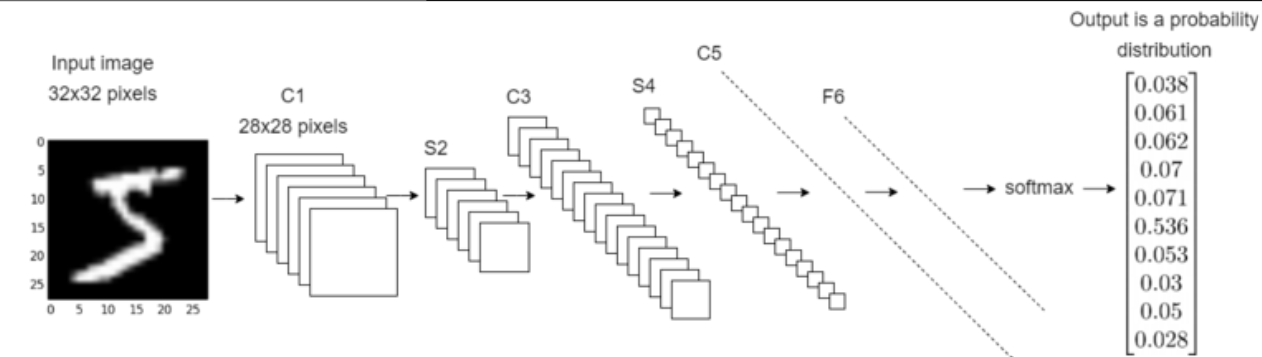
🡪The usage of convolutional layers in a convolutional neural network mirrors the structure of the human visual cortex, where a series of layers process an incoming image and identify progressively more complex features.

1. **Convolutional Neural Network Design**

🡪The architecture of a convolutional neural network is a multi-layered feed-forward neural network, made by stacking many hidden layers on top of each other in sequence. It is this sequential design that allows convolutional neural networks to learn hierarchical features.

🡪**The hidden layers are typically convolutional layers followed by activation layers, some of them followed by pooling layers**.

🡪A simple convolutional neural network that aids understanding of the core design principles is the early convolutional neural network LeNet-5, published by Yann LeCun in 1998. LeNet is capable of recognizing handwritten characters.



**🡺Example Convolutional Neural Network Layers Explained**

🡪LeNet takes an input image of a handwritten digit of size 32x32 pixels and passes it through a stack of the following layers. Each layer except the last is followed by a tanh activation function:

|  |  |
| --- | --- |
| **C1** | The first convolutional layer. This consists of six convolutional kernels of **size 5x5**, which ‘walk over’ the input image. C1 outputs six images of size 28x28. **The first layer of a convolutional neural network normally identifies basic features such as straight edges and corners**. |
| **S2** | **A subsampling layer, also known as an average pooling layer**. Each square of four pixels in the C1 output is averaged to a single pixel. S2 scales down the six 28x28 images by a factor of 2, producing six output images of size 14x14. |
| **C3** | The second convolutional layer. **This consists of 16 convolutional kernels**, each of **size 5x5**, which take the six 14x14 images and walk over them again, producing 16 images of size 10x10. |
| **S4** | The second average pooling layer. S4 scales down the sixteen 10x10 images to sixteen 5x5 images. |
| **C5** | **A fully connected convolutional layer with 120 outputs**. **Each of the 120 output nodes is connected to all of the 400 nodes (5x5x16) that came from S4**. **At this point the output is no longer an image, but a 1D array of length 120**. |
| **F6** | A fully connected layer mapping the 120-array to a new array of length 10. Each element of the array now corresponds to a handwritten digit 0-9. |
| **Output Layer** | A softmax function which transforms the output of F6 into a probability distribution of 10 values which sum to 1. |

🡪**LeNet-5 is one of the simplest convolutional neural networks, with six layers**. This gives it enough power to distinguish small handwritten digits but not, for example, the 26 letters of the alphabet, and especially not faces or objects.

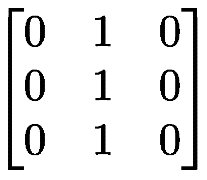
🡪**Today the most sophisticated networks may have more than 30 layers and millions of parameters, and also involve branching, however the basic building blocks of convolutional kernels remain the same**.

**🡺Convolutional Layer**

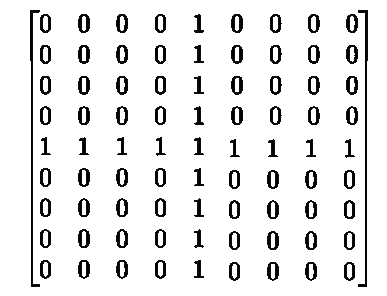
🡪**The key building block in a convolutional neural network is the convolutional layer. We can visualize a convolutional layer as many small square templates, called convolutional kernels, which slide over the image and look for patterns**. **Where that part of the image matches the kernel’s pattern, the kernel returns a large positive value, and when there is no match, the kernel returns zero or a smaller value**.

**🡺Example Calculation of Convolutional on a matrix**

**🡪Mathematically, the kernel is a matrix of weights. For example, the following 3x3 kernel detects vertical lines.**

****

**🡪Let us imagine an 9x9 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:**

****

**🡪**Imagine 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 9x9 with a 3x3 convolution kernel is a new image of size 7x7.

🡪So in the above example, first the kernel is placed in the top left corner and each element of the kernel is multiplied by each element in the red box in the top left of the original image. Since these values are all 0, the result for that cell is 0 in the top left of the output matrix.

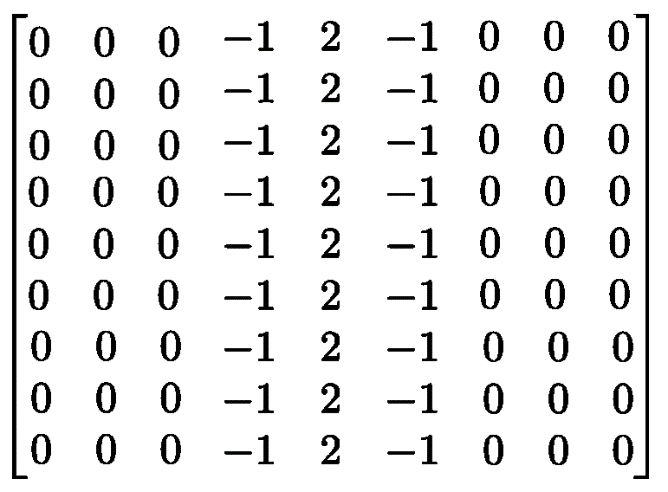
🡪Now let us consider the position of the blue box in the above example. It contains part of a vertical line. When the kernel is placed over this vertical line, it matches and returns 3.

🡪Recall that this convolution kernel is a vertical line detector. For the parts of the original image which contained a vertical line, the kernel has returned a value 3, whereas it has returned a value of 1 for the horizontal line, and 0 for the empty areas of the image.

🡪In practice, a convolution kernel contains both weights and biases, similar to the formula for linear regression. So an input pixel is multiplied by the weight and then the bias is added.

1. **Example of convolution in image**

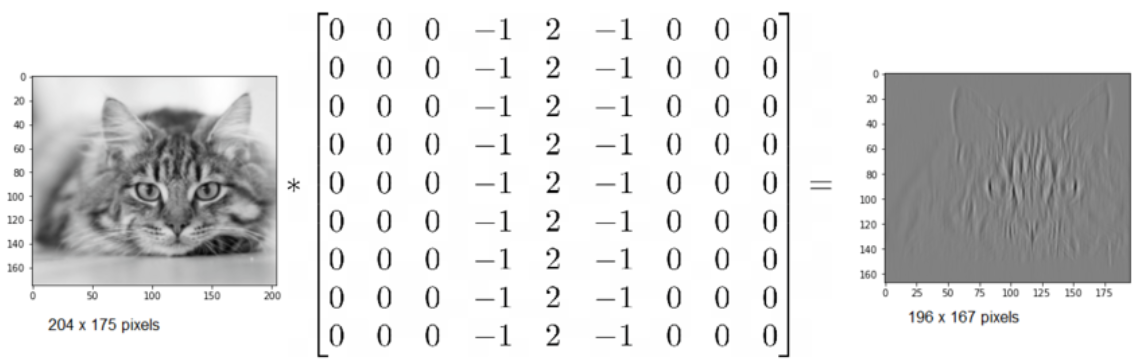
🡪 Let us consider the following 9x9 convolution kernel, which is a slightly more sophisticated vertical line detector than the kernel used in the last example:



🡪 And we can take the following image of a tabby cat with dimensions 204x175, which we can represent as a matrix with values in the range from 0 to 1, where 1 is white and 0 is black.



🡪 Applying the convolution, we find 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 (9x9).



🡪 Despite its simplicity, the ability to detect vertical or horizontal lines, corners, curves, and other simple features, is an extremely powerful property of the convolution kernel. We recall that a convolutional layer is made up of a series of convolution kernels. Typically, the first layer of a convolutional neural network contains a vertical line detector, a horizontal line detector, and various diagonal, curve and corner detectors. These feature detector kernels are not programmed by a human but in fact are learned by the neural network during training, and serve as the first stage of the image recognition process.

🡪 Later layers in the neural network are able to build on the features detected by earlier layers and identify ever more complex shapes.

1. **Activation functions in a Convolutional Neural Network**

🡪 After passing an image through a convolutional layer, the output is normally passed through an activation function. **Common activation functions include the sigmoid function**.

**🡺What is an activation function?**

🡪 An activation function is a mathematical function that controls the output of a neural network. Activation functions help in determining whether a neuron is to be fired or not.

🡪Some of the popular activation function are:

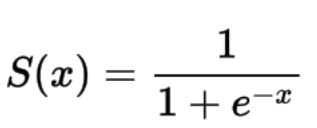
1. Binary Step
2. Linear
3. Sigmoid
4. Tanh
5. ReLu
6. Leaky ReLu
7. Softmax

🡪 Activation is responsible for adding non-linearity to the output of a neural network model. Without an activation function, a neural network is simply a linear regression.

🡪 The mathematical equation for calculating the output of a neural network is:



🡪The formula of the sigmoid function is as followed:



🡪 You can see that the denominator will always be greater than 1, **therefore the output will always be between 0 and 1**.

🡪 and the ReLU function, also known as the rectified linear unit, which is the same as taking the positive component of the input:



🡪 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 removes the noise originating from other non-vertical features.

