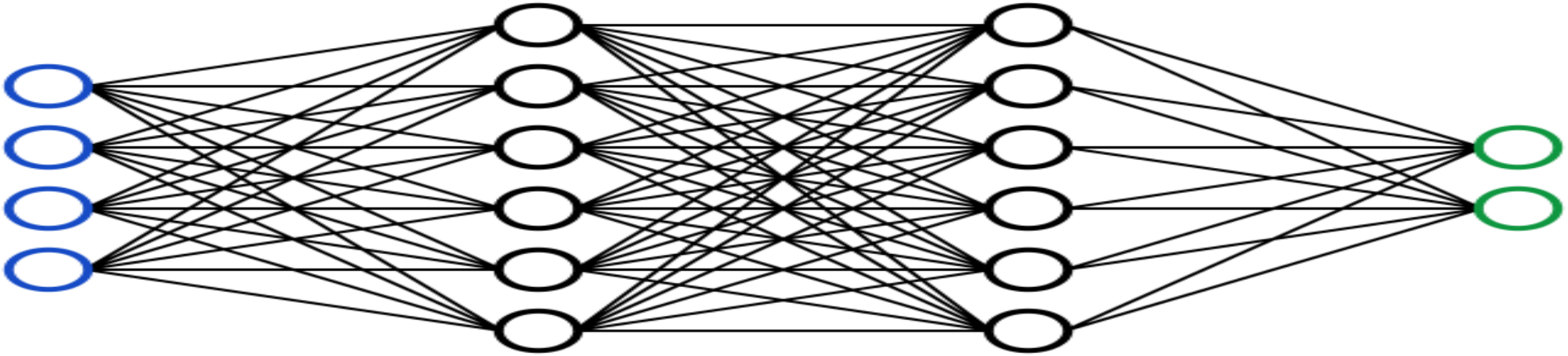


Artificial Neural Network (ANN)

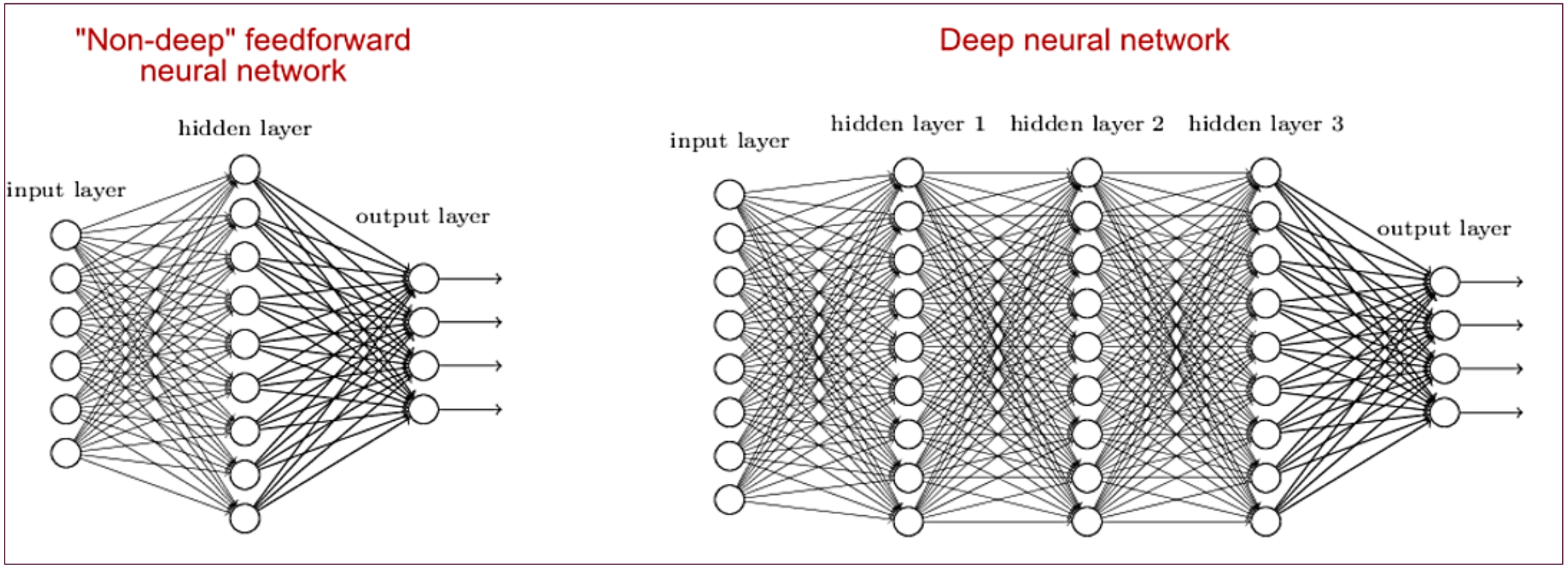
Lecture7

Dina El-Manakhly, Ph. D.

dina_almnakhly@science.suez.edu.eg

Deep neural network (DNN)

- In short, "shallow" neural networks is a term used to describe NN that usually have only one hidden layer as opposed to deep NN which have several hidden layers,



Deep neural network (Cont'd)

- In deep learning, the number of hidden layers, can be large, say about 1000 layers.
- Deep learning (DL) models produce much better results than normal networks.
- DNNs can model complex non-linear relationship
- We mostly use the gradient descent method for optimizing the network and minimizing the loss function. The cost function or the loss function is the difference between the generated output and the actual output.
- For complex patterns like a human face, shallow neural networks fail and have no alternative but to go for deep neural networks with more layers. The deep nets are able to do their job by breaking down the complex patterns into simpler ones.

Deep neural network (Cont'd)

- For example: In a deep neural network, especially in the context of computer vision tasks like recognizing human faces, each layer is responsible for detecting specific features or patterns in the input data.

Here's how it typically works:

- The first few layers of the network are responsible for detecting low-level features such as edges, textures, and gradients in the input images.
- Finally, in the deeper layers of the network, high-level features are detected, which represent more abstract and complex concepts. In the case of recognizing human faces, these high-level features might correspond to facial components like eyes, noses, mouths, or even entire faces.

By organizing the layers of the neural network in this hierarchical manner, with each layer responsible for detecting increasingly complex features, the network can effectively learn to represent and understand the input data. This hierarchical representation is one of the key factors that make deep neural networks so powerful for tasks like image recognition, where the input data exhibits complex structures and relationships.

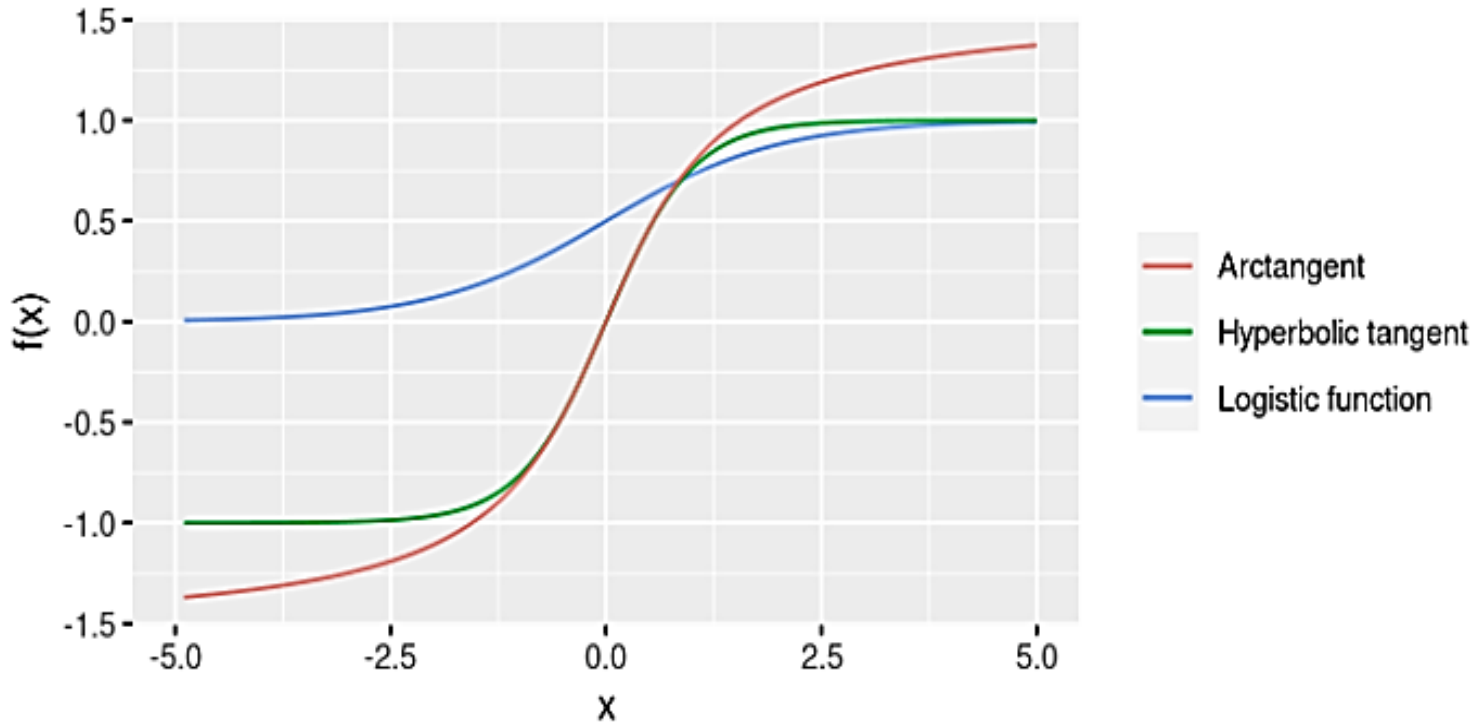
Deep learning problems

1) Vanishing gradients problem
   - Gradients often get smaller as the algorithm progresses down to the lower layers.
   - The gradient descent update leaves the lower layer weights unchanged and training never converges to good solution.
2) Exploding gradients problem
   - Gradients can grow bigger and bigger, so many layers get insanely large weight updates and the algorithm diverges.
3) With such a large network, training would be extremely slow and need a hardware constraint. However recent high performance GPUs have been able to train such deep nets under a week; while fast CPUs could have taken weeks or perhaps months to do the same.

Solving vanishing and exploding problem

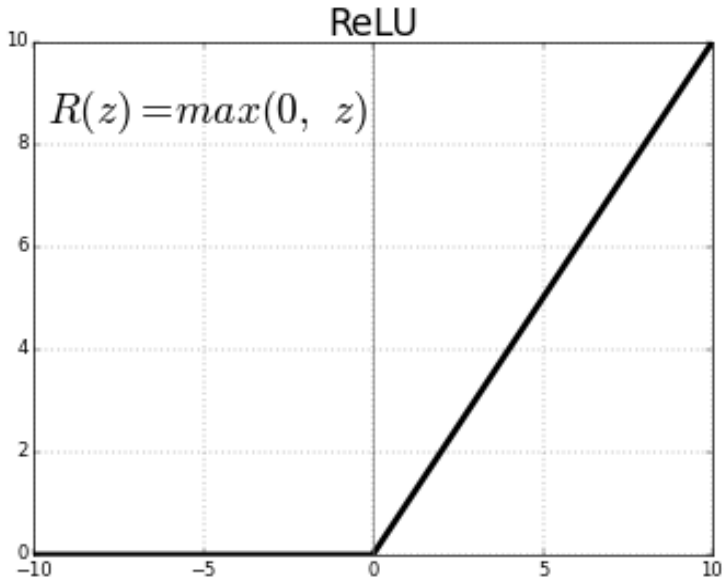
- Popular logistic sigmoid activation function.

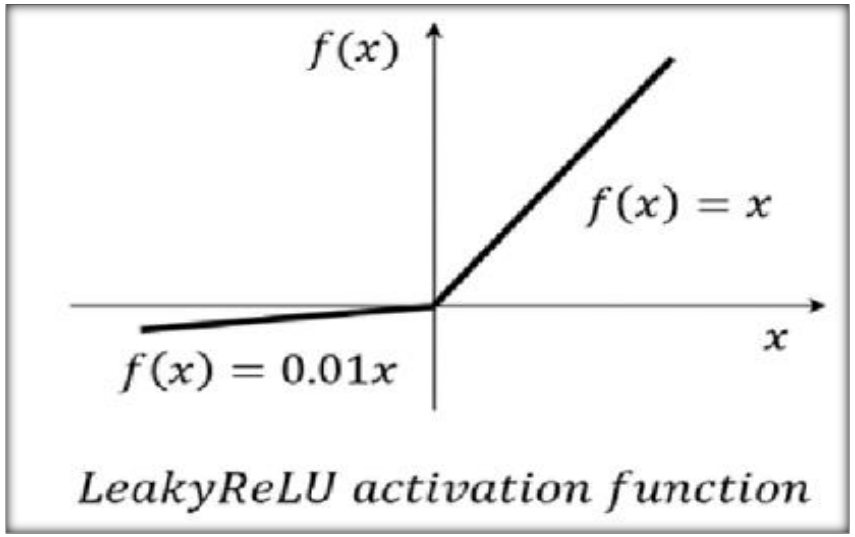
- Hyperbolic tangent function has a mean of 0 and behaves slightly better than the logistic function in DNN



The problem of ReLU

- It suffers from a problem (dying ReLUs) during training, some neurons effectively die.
- They stop outputting anything other than 0.
- In some cases, you may find that half of your network's neurons are dead training.
- To solve this problem, we may want to use a variant of the ReLU function such as the leaky ReLU.





Convolutional neural network (CNN)

- What is Convolutional Neural Network?

  A convolutional neural network is a feed-forward neural network that is generally used to detect and classify objects in an image. It's also known as a ConvNet.

- In image classification,
  - Input pixels are fed into a neural network.
  - Hidden layers extract features through various calculations, including convolution, ReLU, and pooling.
  - A fully connected layer then identifies the object in the image.

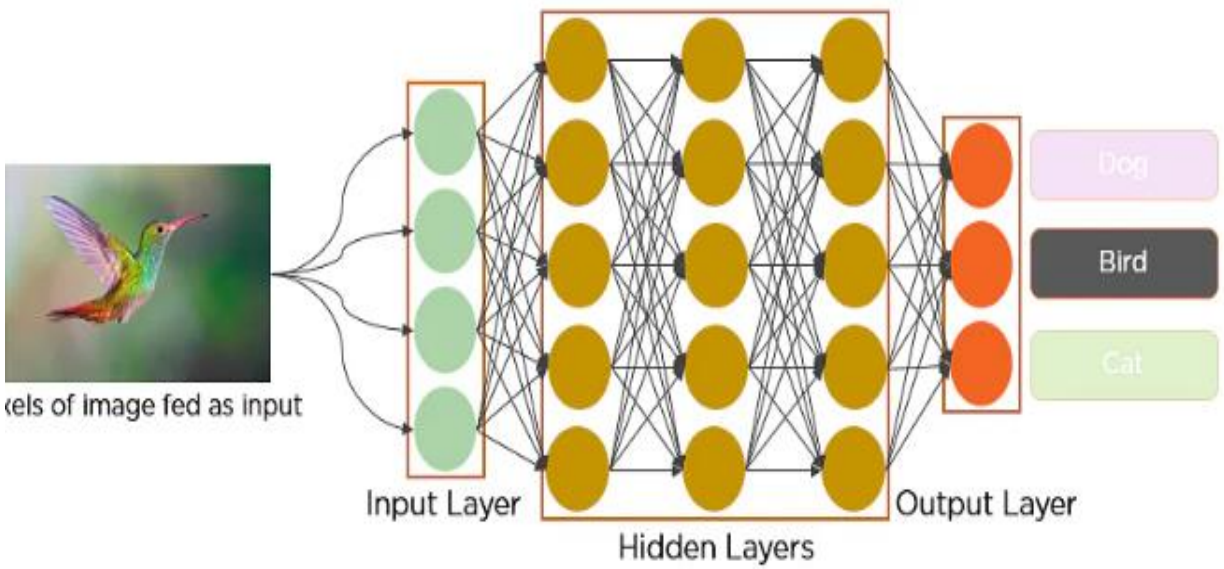


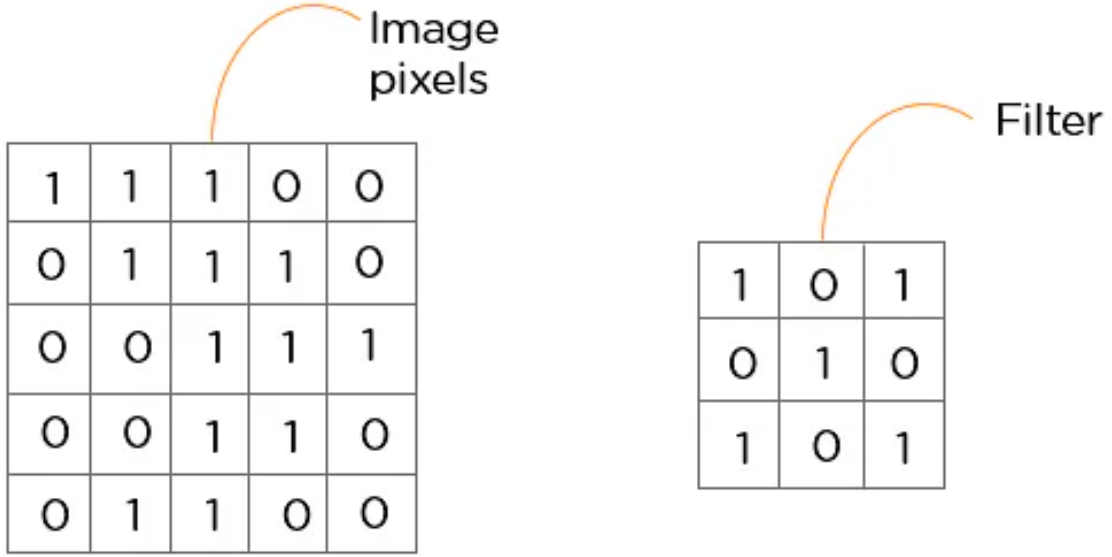
Fig: Convolutional Neural Network to identify the image of a bird

Layers in a convolutional neural network

- A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:
  1. Convolution layer
  2. ReLU layer
  3. Pooling layer
  4. Fully connected layer

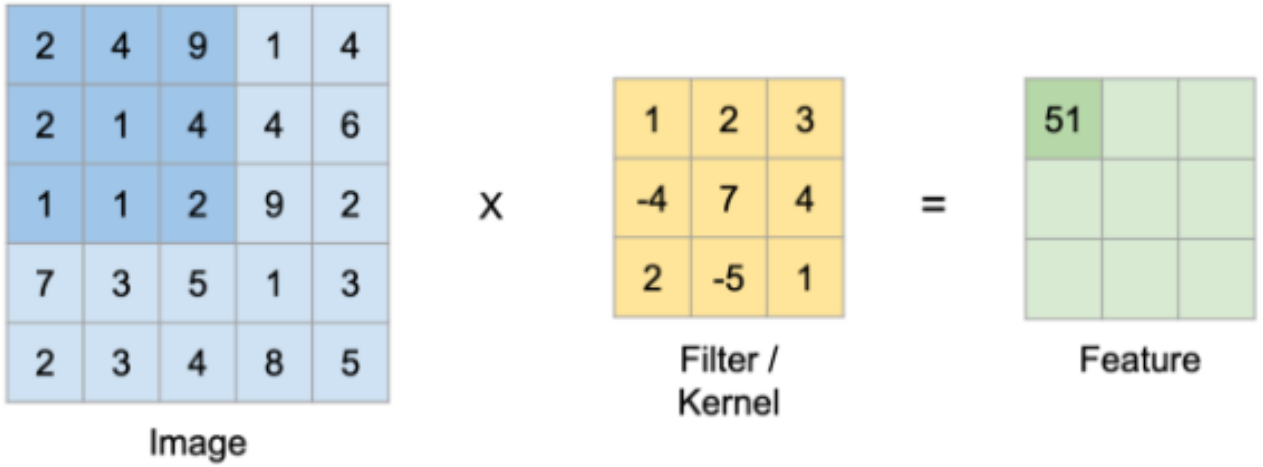
1- Convolution layer

- Consider the following 5x5 image whose pixel values are either 0 or 1. There's also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.
- The filter acts as a feature extractor by detecting specific patterns or features, such as edges, textures, shapes, or other visual elements, present in different parts of the input image.



Example 1: How does a convolution layer work?

- We use multiple convolution **filters** or **kernels** that run over the image and compute a dot product. Each filter extracts different features from the image.
- Lets consider a filter of size 3x3 and an image of size 5x5. We perform an element wise multiplication between the image pixel values that match the size of the kernel and the kernel itself and sum them up. This provides us a single value for the feature cell.

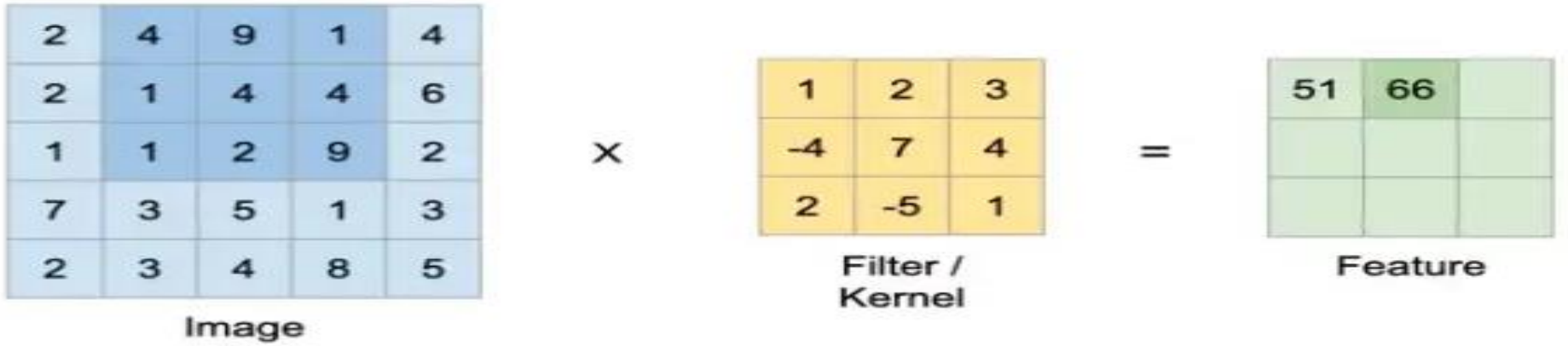


Convolution operation step — 1

$$2*1 + 4*2 + 9*3 + 2*(-4) + 1*7 + 4*4 + 1*2 + 1*(-5) + 2*1 = 51$$

Example1: How does a convolution layer work?

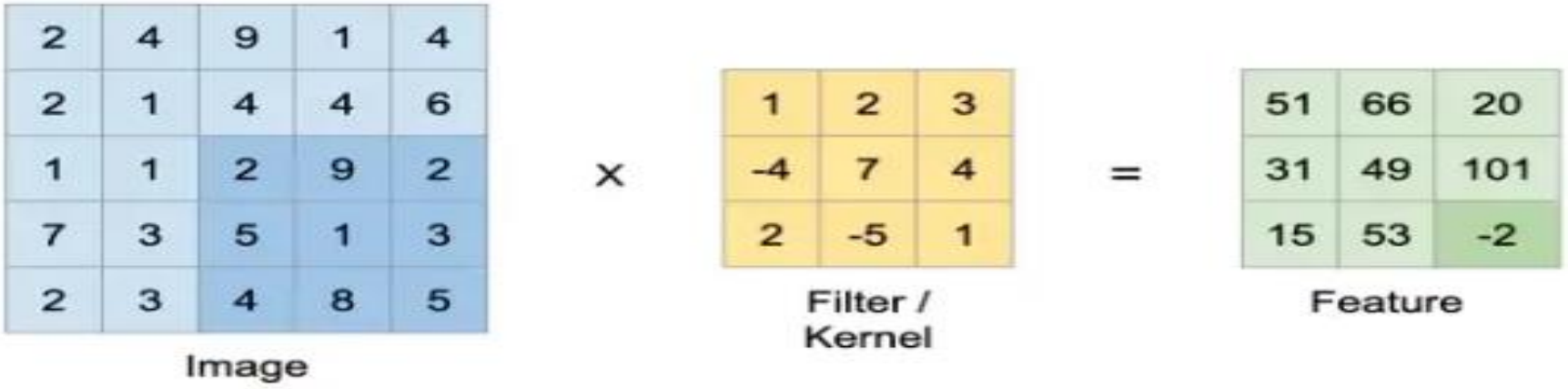
Filter continues to run further on the image and produce new values as shown below.



Convolution operation step — 2

$4*1 + 9*2 + 1*3 + 1*(-4) + 4*7 + 4*4 + 1*2 + 2*(-5) + 9*1 = 66$

Example1: How does a convolution layer work?



Convolution operation step — final

$2*1 + 9*2 + 2*3 + 5*(-4) + 1*7 + 3*4 + 4*2 + 8*(-5) + 5*1 = -2$

In the above example we are sliding the kernel by 1 pixel. This is called **stride.** We can have the kernel move by different stride values to extract different kinds of features.

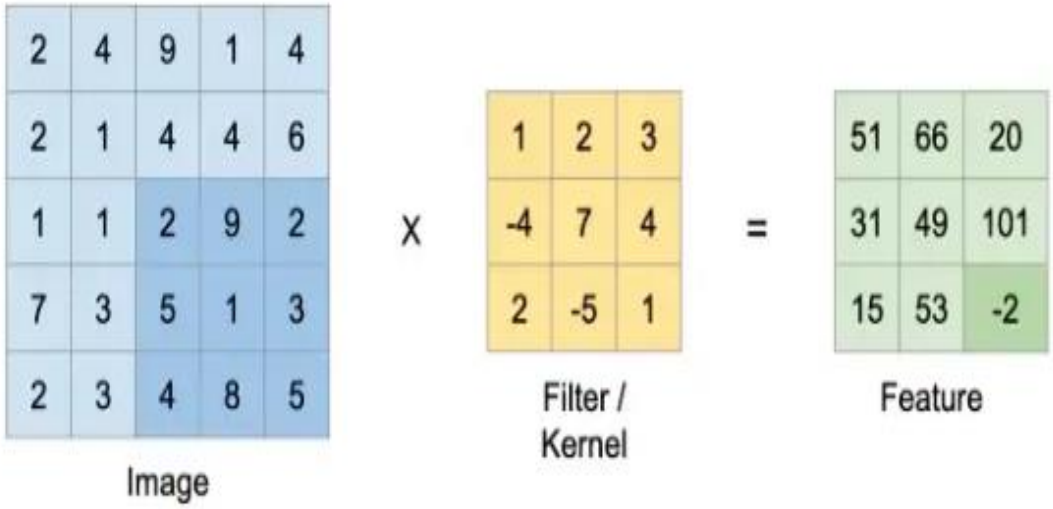
Calculating the size of feature for a particular kernel

- The amount of stride we choose affects the size of the feature extracted. The equation to calculate the size of feature for a particular kernel size is as follows:

Feature size = ((Image size − Kernel size) / Stride) + 1

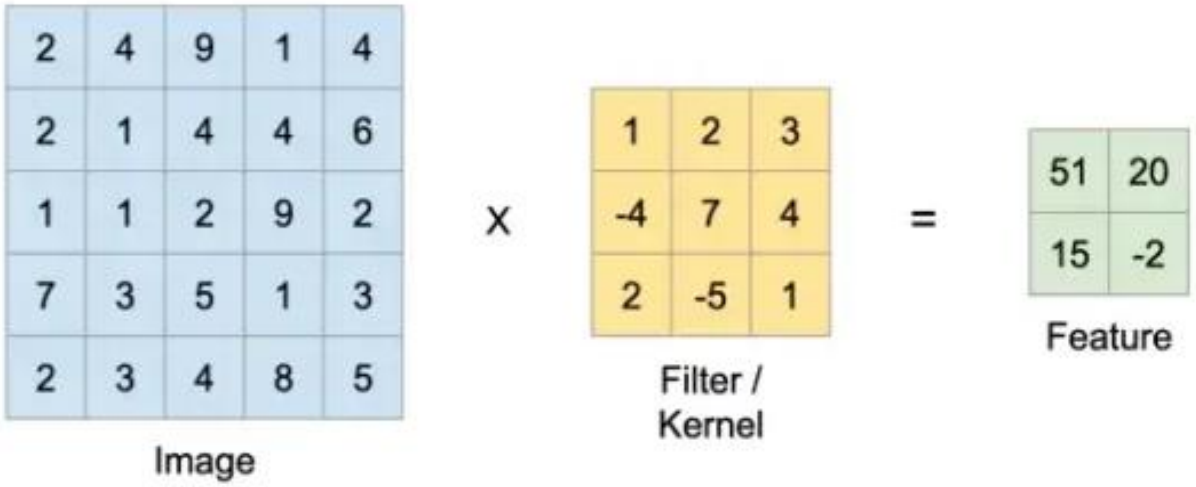
- We can put the values for the above example and verify it.

Feature size = ((5 − 3) / 1) + 1 = 3



Sliding the kernel by 2

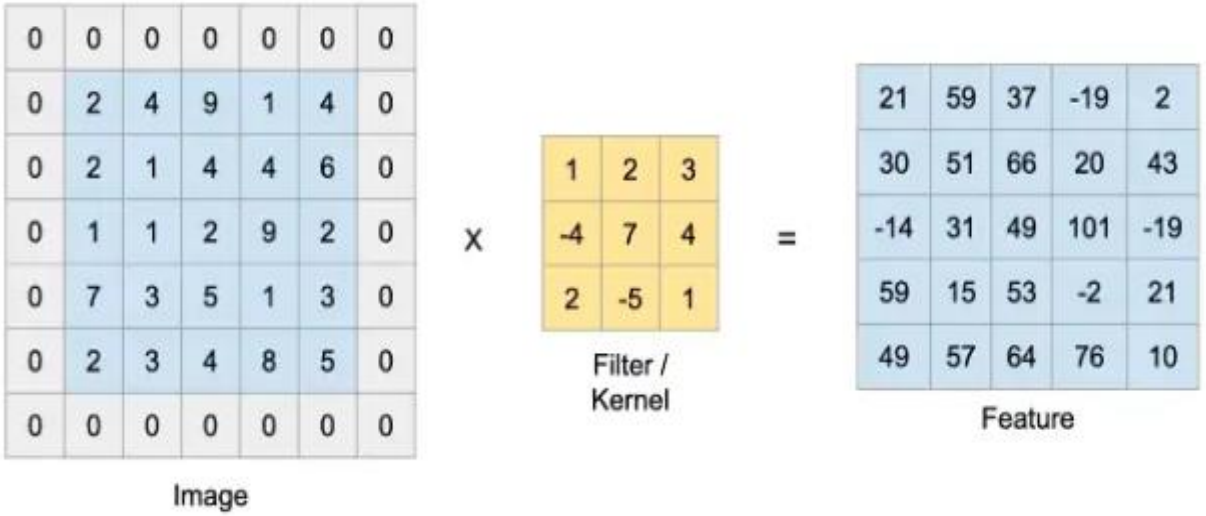
- with a stride of 2 the kernel of size 3x3 on a image of size 5x5 would only be able to extract a feature of size 2.



Convolution operation kernel size 3 and stride 2

What if you want the feature to be of the same size as the input image?

- You can achieve this by padding the image. **Padding** is a technique to simply add zeros around the margin of the image to increase it's dimension. Padding allows us to emphasize the border pixels and in order lose less information.
- Here is an example with an input image of size 5x5 which is padded to 7x7 i.e. padding size of 1 and convoluted by a kernel of size 3x3 with stride of 1 resulting in a feature of size 5x5.



Convolution operation kernel size 3, stride 1 and padding 1

What if you want the feature to be of the same size as the input image?

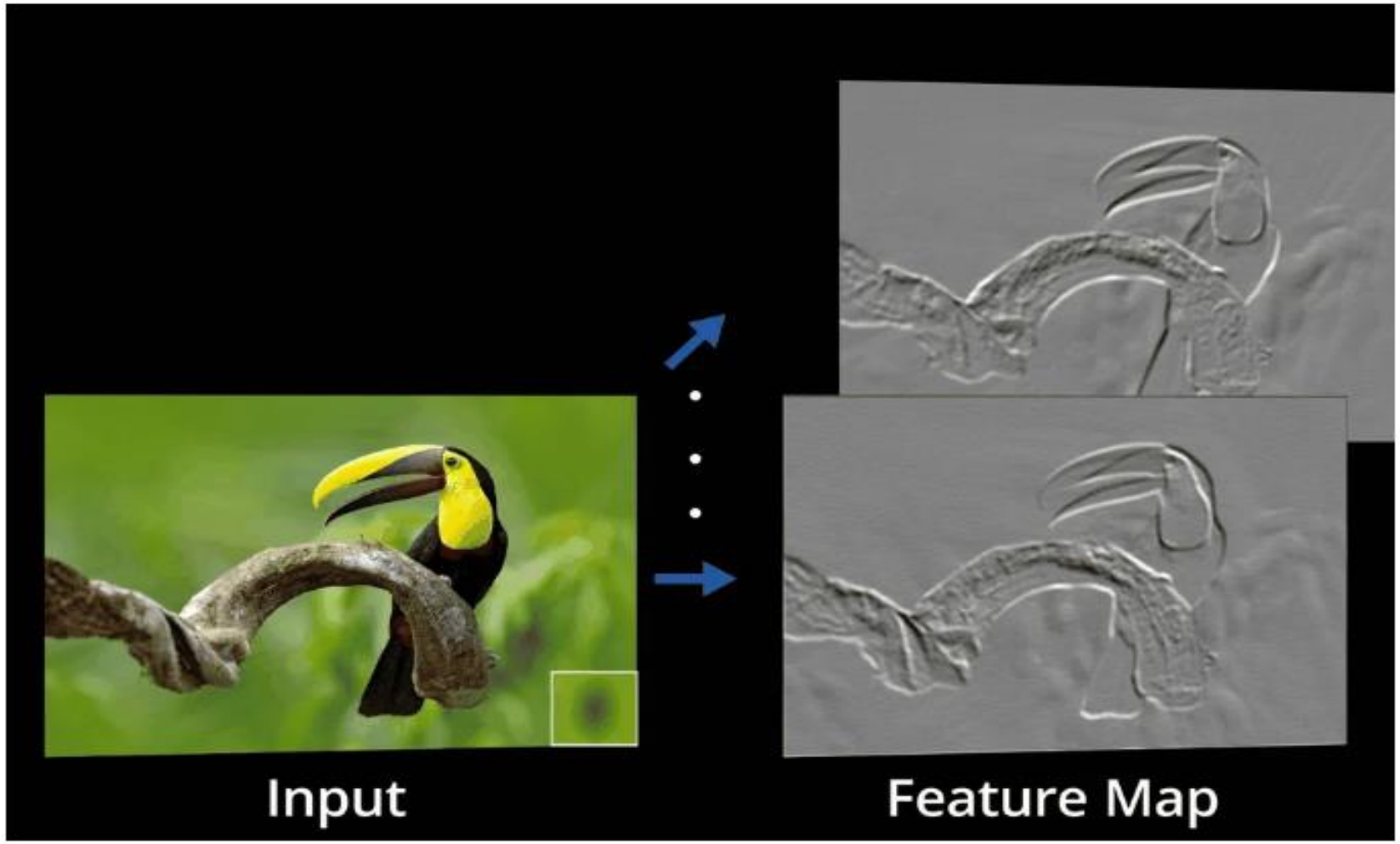
- The equation to calculate the size of feature for a particular kernel size when considering a padded image is as follows:

  Feature size = ((Image size + 2 * Padding size − Kernel size) / Stride)+1

- We can put in the values for the previous example and verify it.

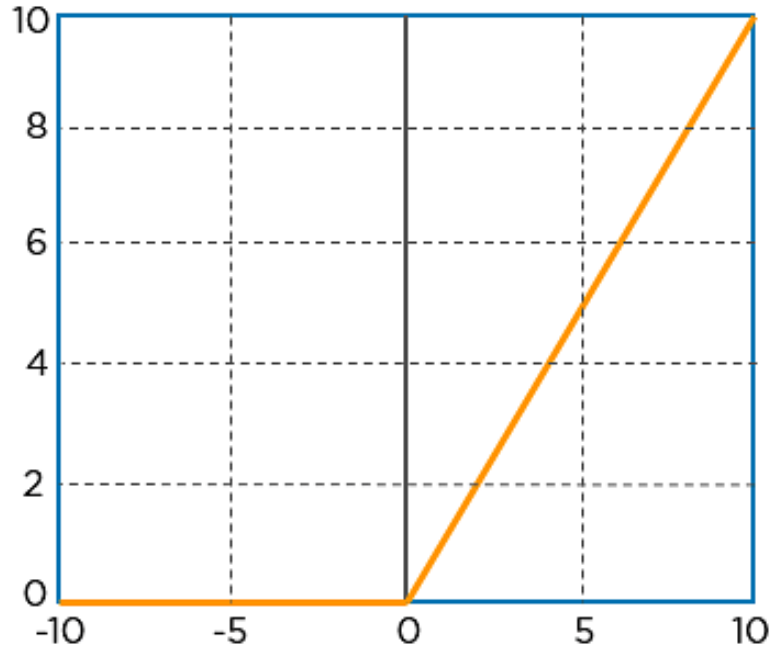
  Feature size = ((5 + 2 * 1 − 3) / 1) + 1= 5

Convolved map: the original image is scanned with multiple convolutions

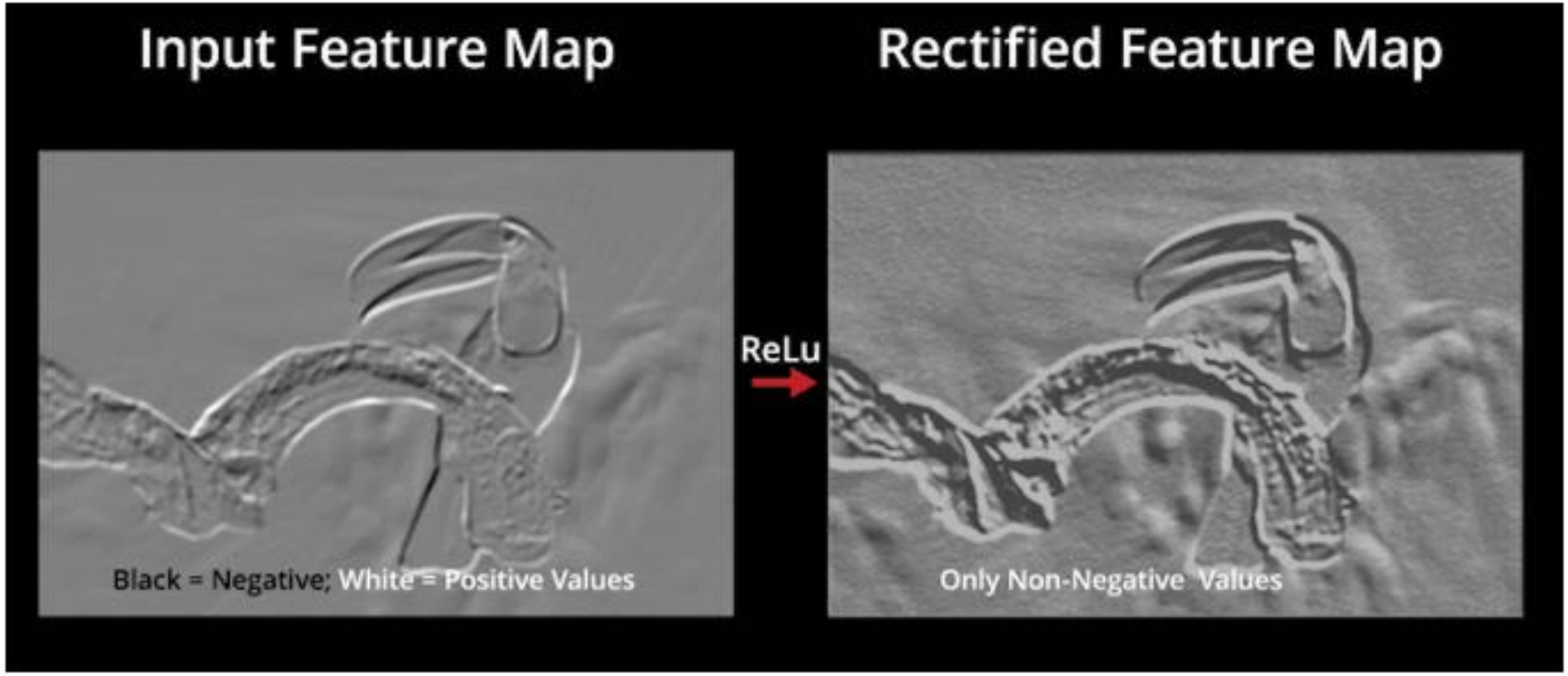


2- RELU LAYER

- ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.
- ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map.

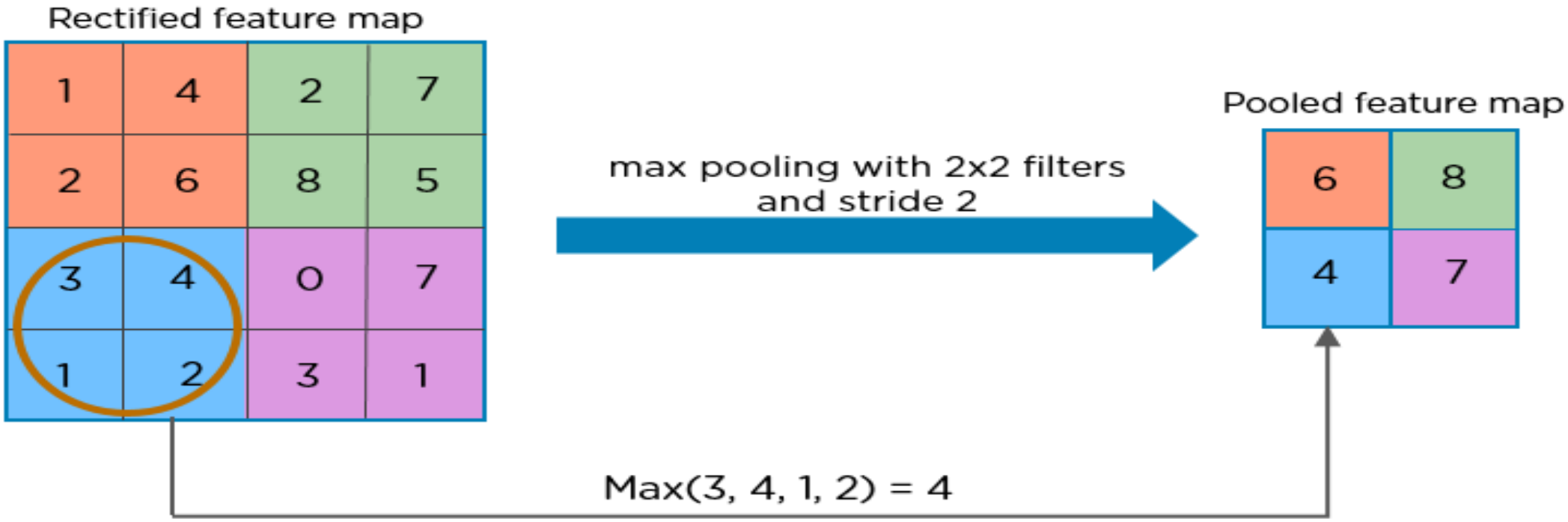


RELU LAYER


Input Feature Map
Rectified Feature Map
ReLu
Black = Negative; White = Positive Values
Only Non-Negative Values

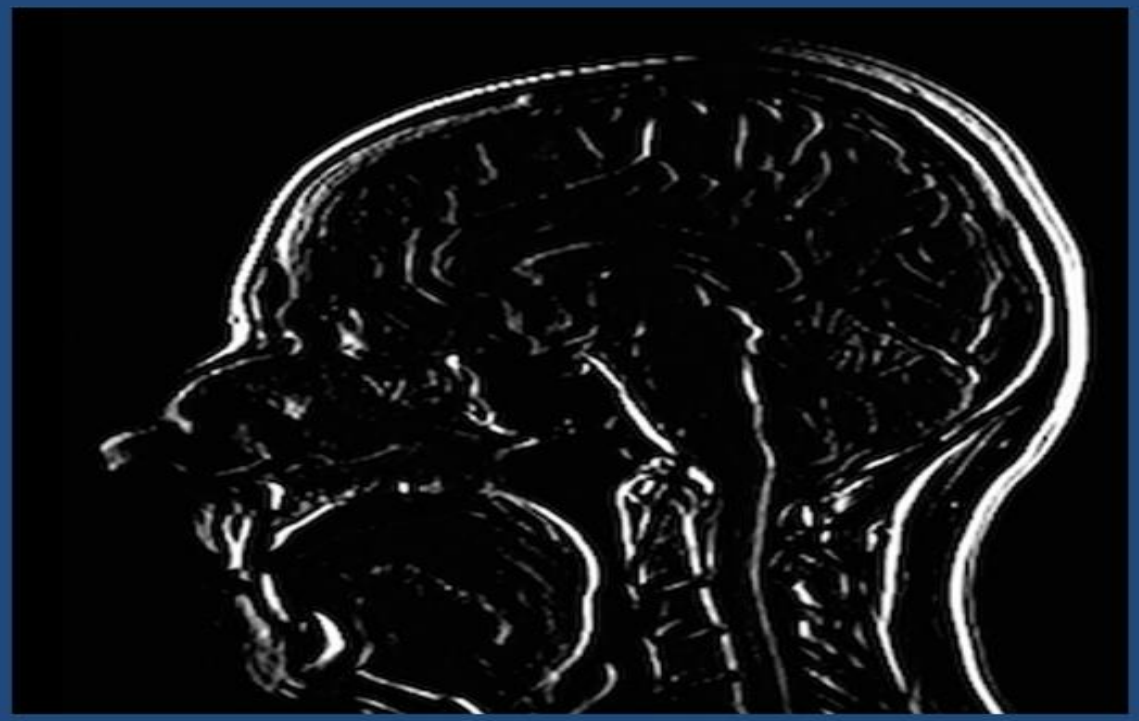
3- POOLING LAYER (MAX OR AVERAGE)

- Pooling is a down-sampling operation that reduces the dimensionality of the feature map. This helps in reducing the number of parameters and computations in the network, which can lead to faster. The rectified feature map now goes through a pooling layer to generate a pooled feature map.

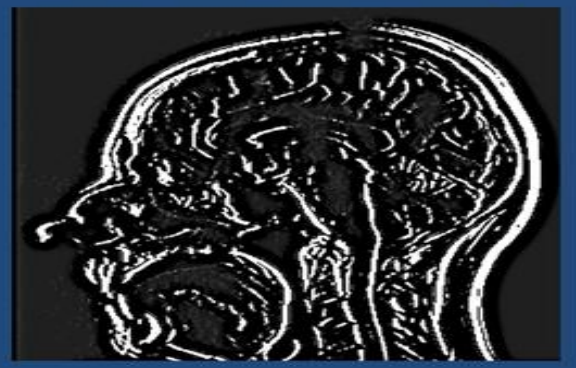


EXAMPLE: POOLING LAYER

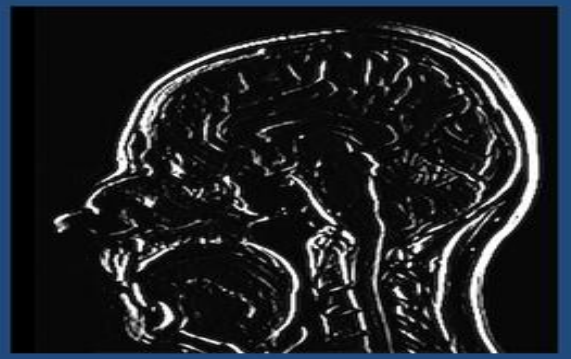
Rectified Feature Map



2x2 Max Pooling

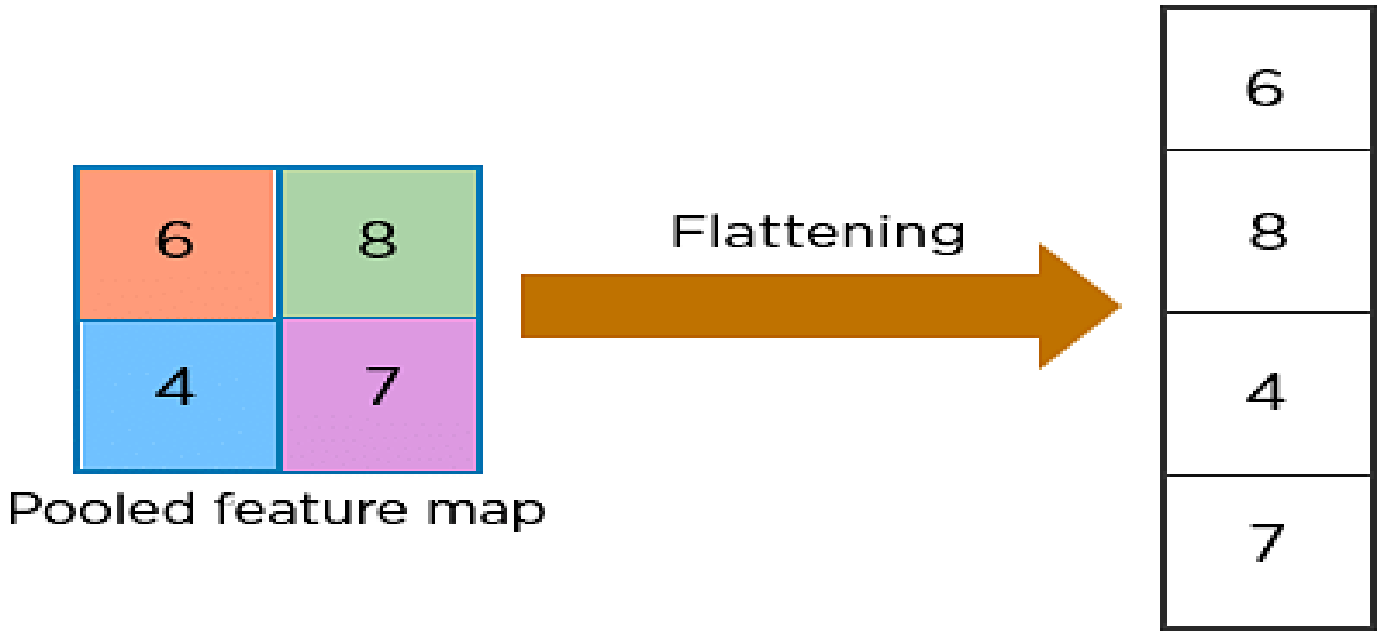


2x2 Avg Pooling

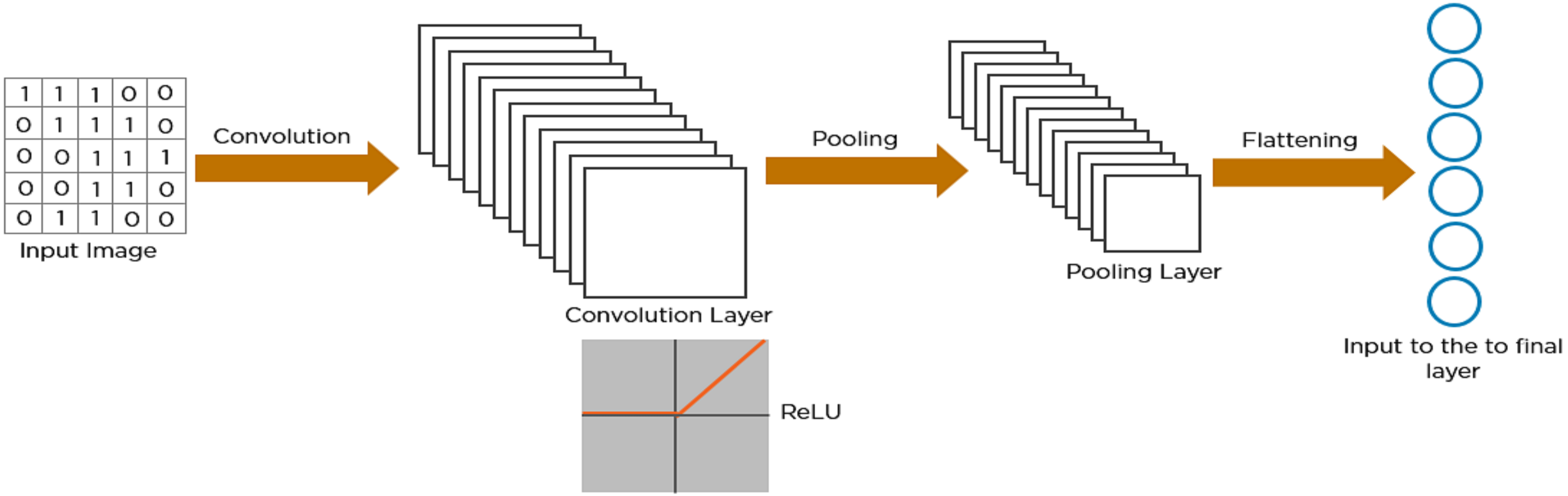


4- Flattening

- The next step in the process is called flattening. Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.



Finally : how the structure of the convolution neural network looks so far



HOW EXACTLY CNN RECOGNIZES A BIRD:

- The pixels from the image are fed to the convolutional layer that performs the convolution operation
- It results in a convolved map
- The convolved map is applied to a ReLU function to generate a rectified feature map
- The image is processed with multiple convolutions and ReLU layers for locating the features
- Different pooling layers with various filters are used to identify specific parts of the image
- The pooled feature map is flattened and fed to a fully connected layer to get the final output