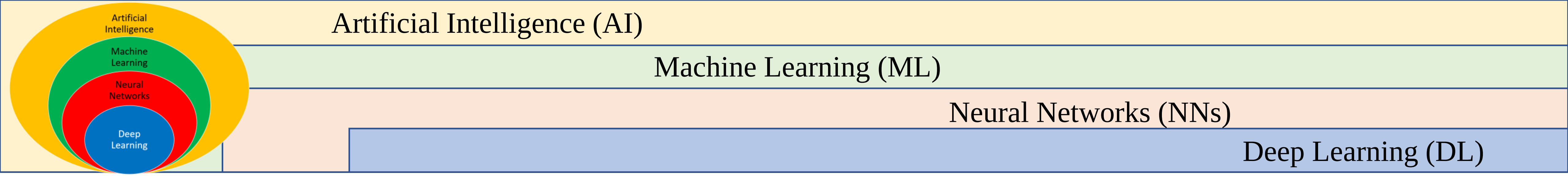
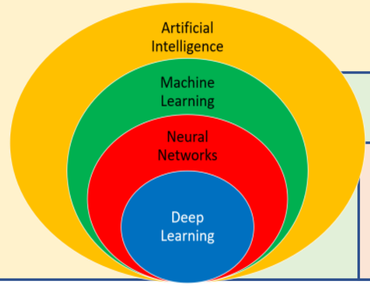


Advanced Artificial Intelligence

Dr. Rastgoo

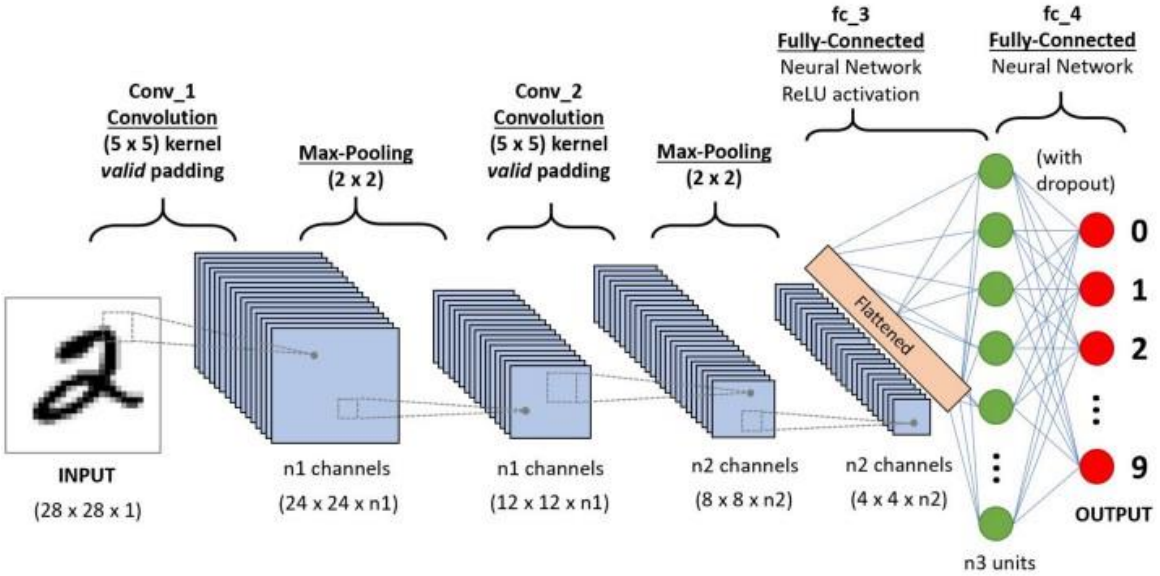

Part 1: Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

- A convolutional neural network (CNN) is a type of artificial neural network used primarily for image recognition and processing, due to its ability to recognize patterns in images.

- A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.



Convolutional Neural Network (ConvNets/CNN)

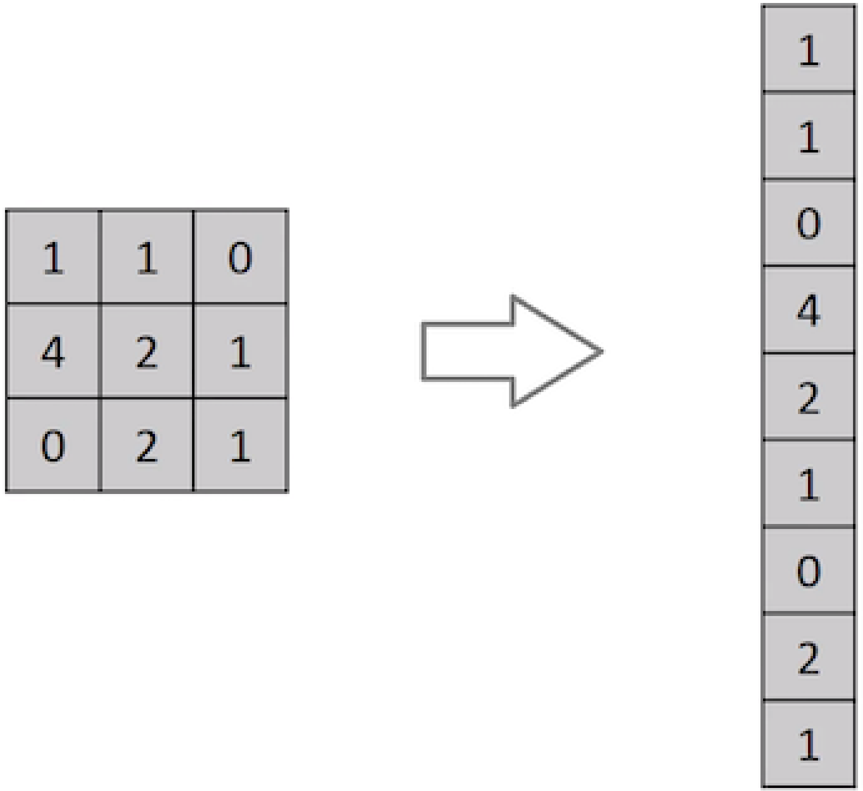
- The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

- While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

Why ConvNets over Feed-Forward Neural Nets?

- An image is nothing but a matrix of pixel values.

- So why not just flatten the image (e.g. 3x3 image matr into a 9x1 vector) and feed it to a Multi-Level Perceptrc for classification purposes?



Why ConvNets over Feed-Forward Neural Nets?

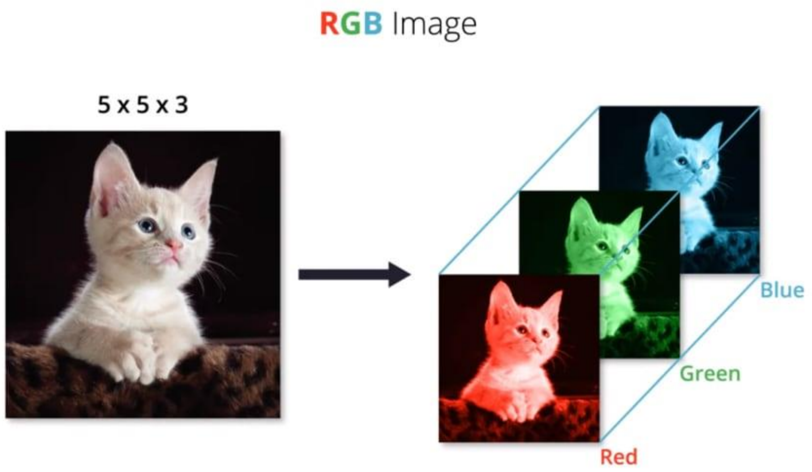
- A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters.

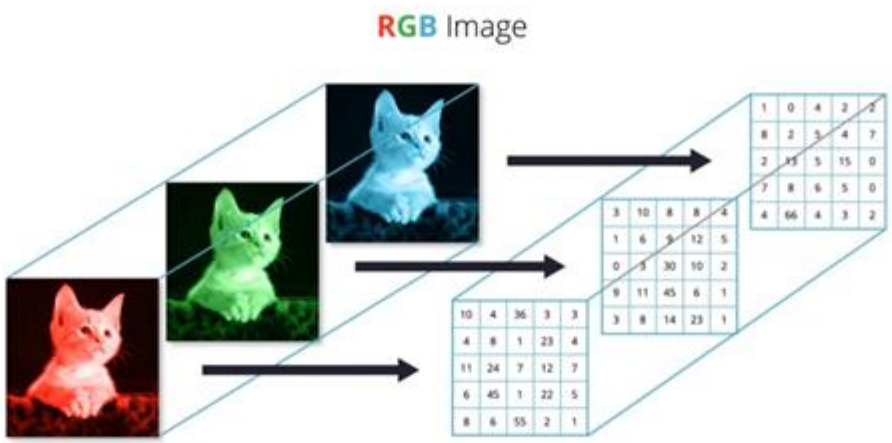
- The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.

- In other words, the network can be trained to understand the sophistication (complexity) of the image better.

Input image

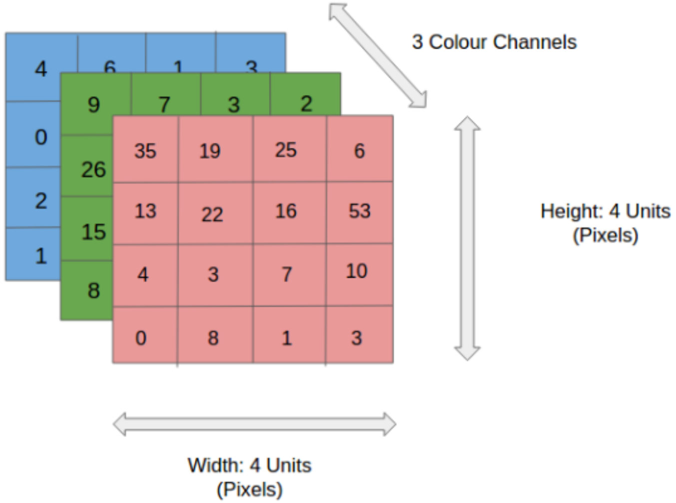
- We have an RGB image which has been separated by its three color planes — Red, Green, and Blue.
- There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.



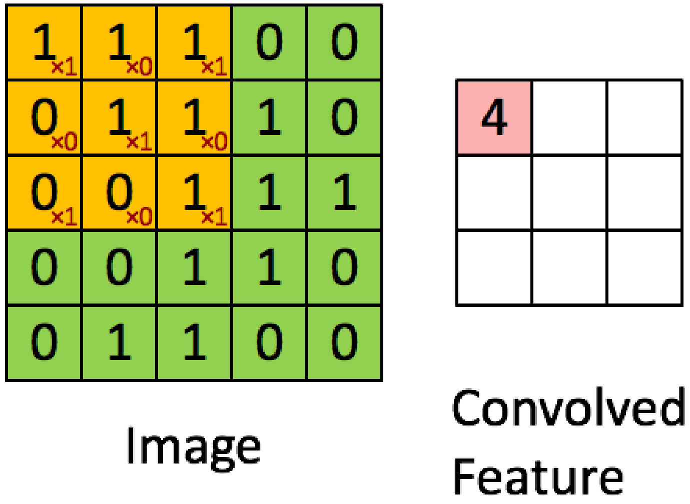


Input image

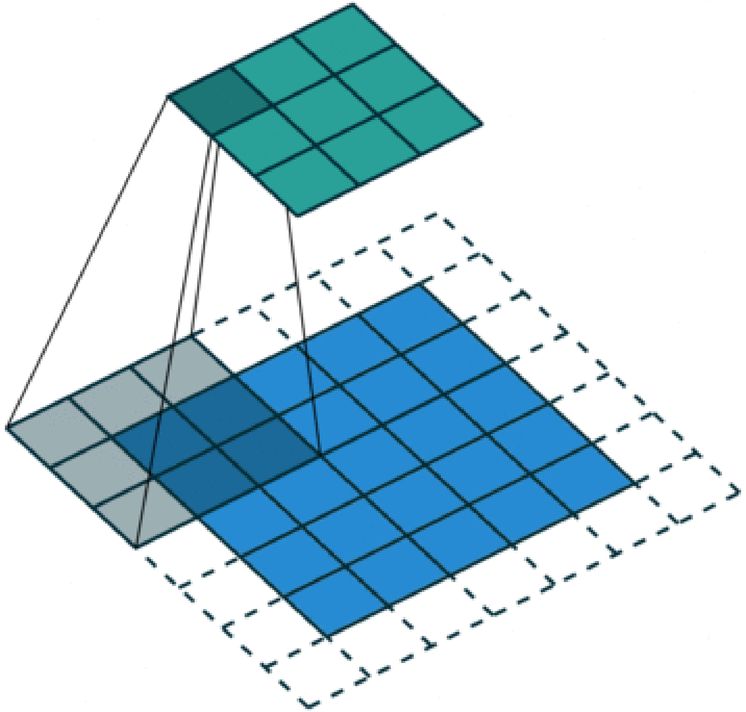
- You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320).
- The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.
- This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.



Convolution Layer: The Kernel

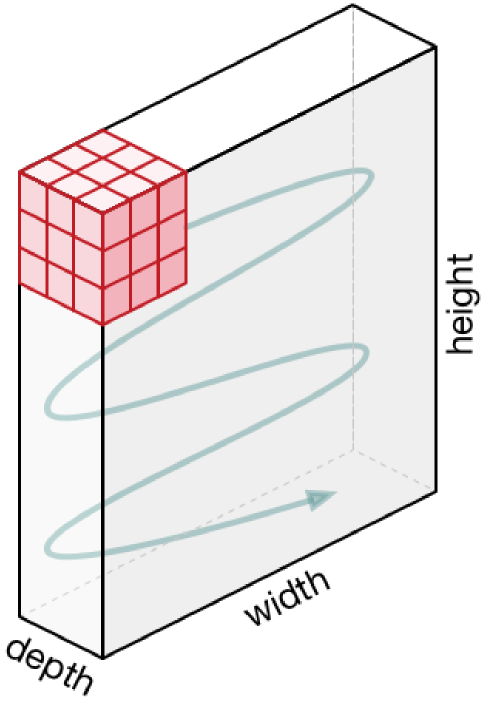


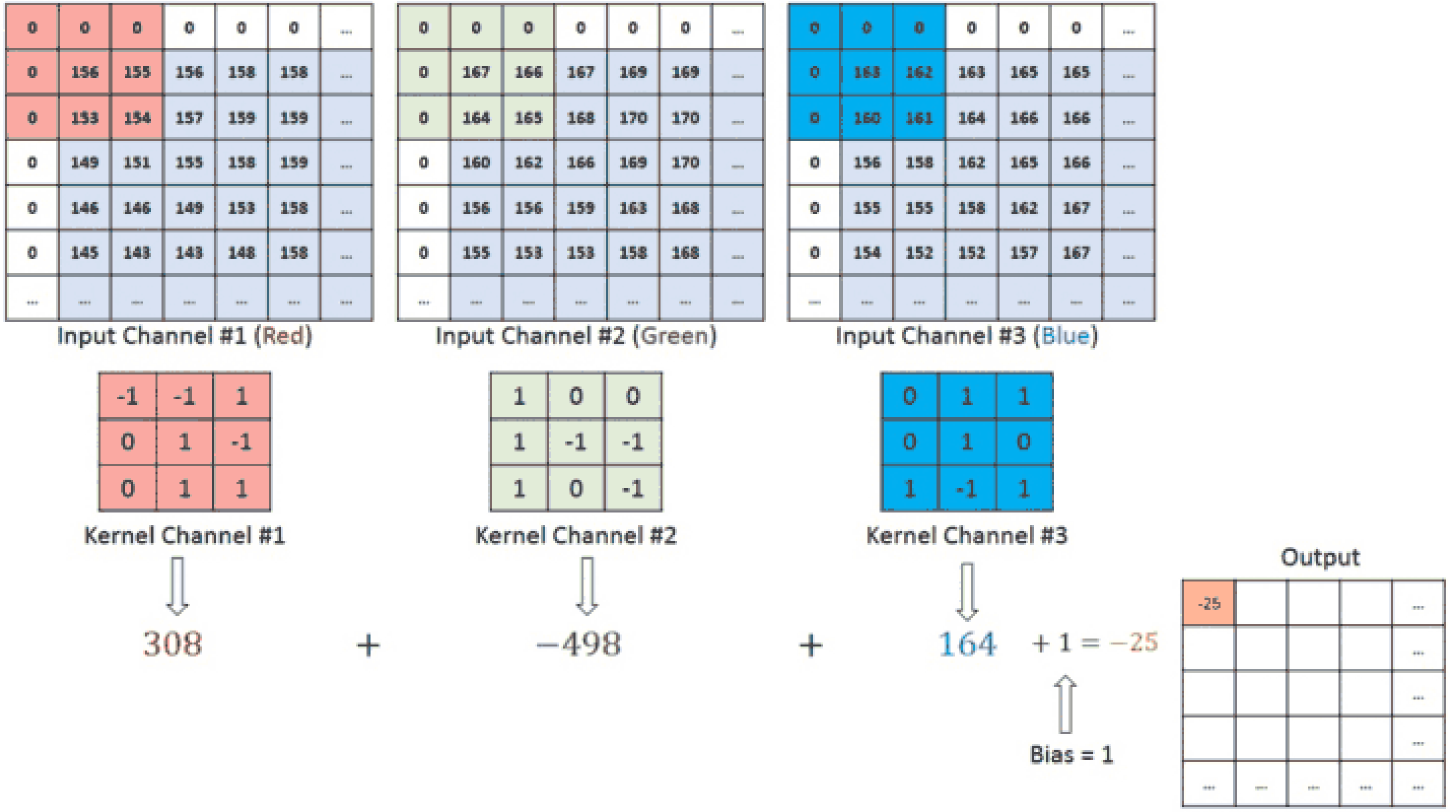
Convolving a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature



Convolution Operation with Stride Length = 2

Convolution Layer: The Kernel





Convolution Layer: The Kernel

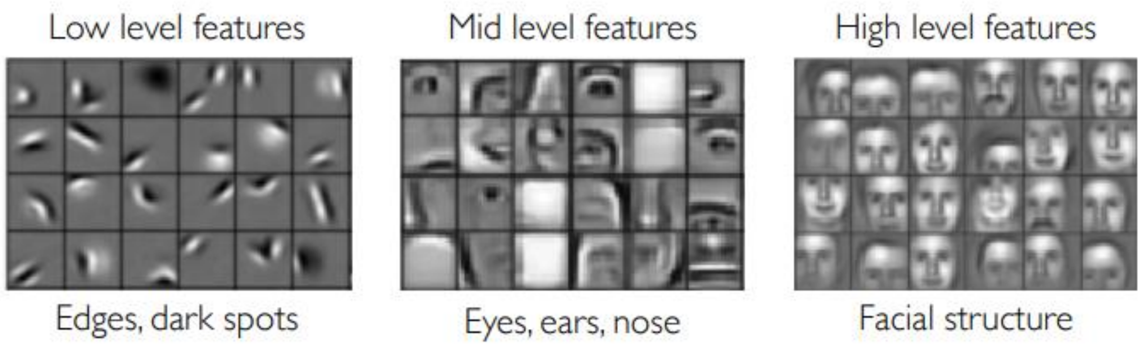
- In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image.

- Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

Convolution Layer: The Kernel

- The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image.

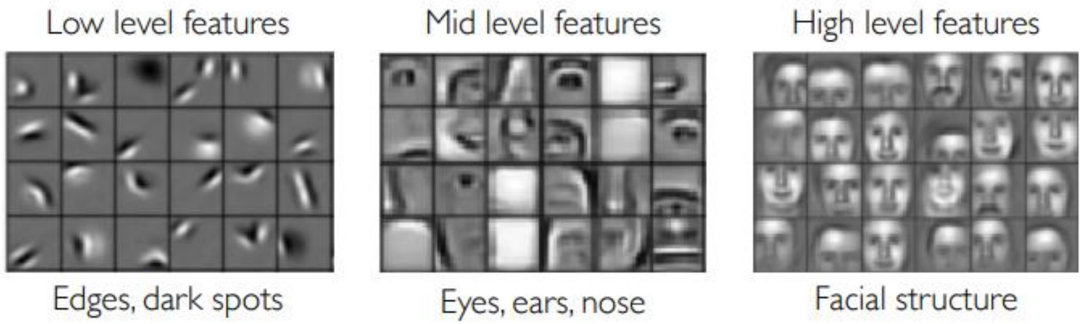
- ConvNets need not be limited to only one Convolutional Layer.

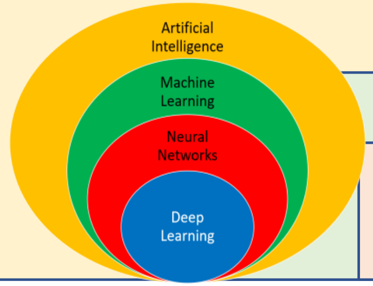


Convolution Layer: The Kernel

- Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc.

- With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.





Convolution Layer: The Kernel

- There are two types of results to the operation: one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same.

- This is done by applying Valid Padding in case of the former, or Same Padding in the case of the latter.

Convolution Layer: The Kernel

- Same Padding: When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1.

Pixel Values

0	0	0	0	0	0	0	0
0	1	0	4	2	125	67	0
0	8	2	5	4	34	12	0
0	20	13	25	15	240	2	0
0	76	8	6	6	100	76	0
0	34	66	134	223	201	3	0
0	255	123	89	55	32	2	0
0	0	0	0	0	0	0	0

Kernel 3 x 3 Pixels

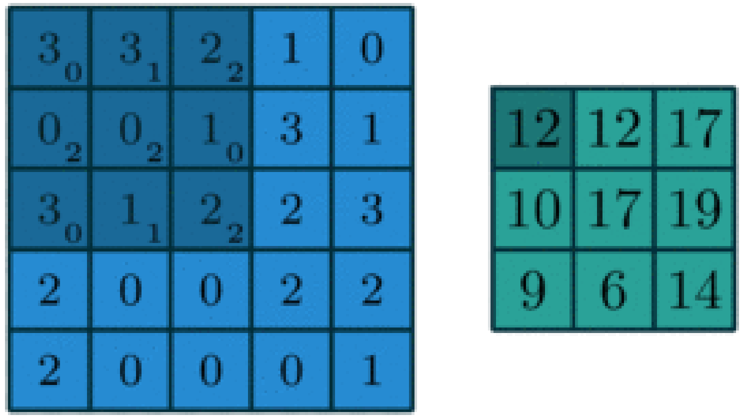
1	2	1
2	4	2
1	2	1

Convoluted Image

22	27	36	313	722	576
91	110	120	522	984	576
284	257	198	755	1360	798
507	567	687	1312	1689	955
1061	1288	1496	1911	1659	702
1400	1480	1269	1249	870	279

Convolution Layer: The Kernel

- Valid Padding: On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself.



Pooling Layer

- Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature.

- This is to decrease the computational power required to process the data through dimensionality reduction.

Pooling Layer

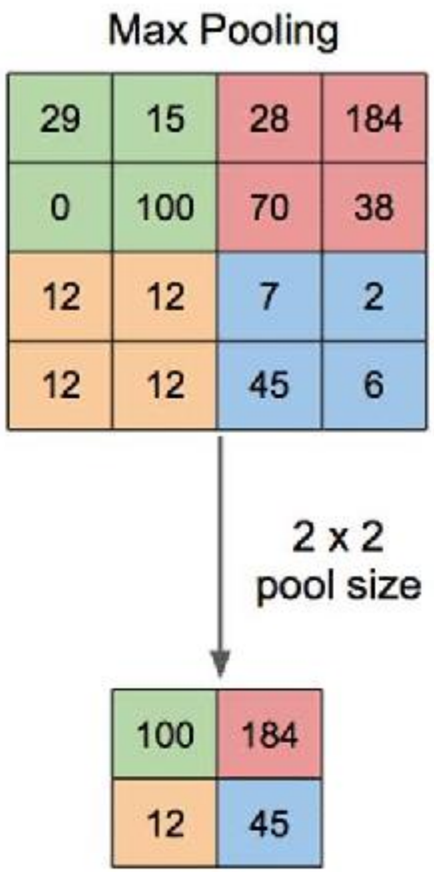
- There are two types of Pooling: Max Pooling and Average Pooling.

- Max Pooling returns the maximum value from the portion of the image covered by the Kernel.

- On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Pooling Layer

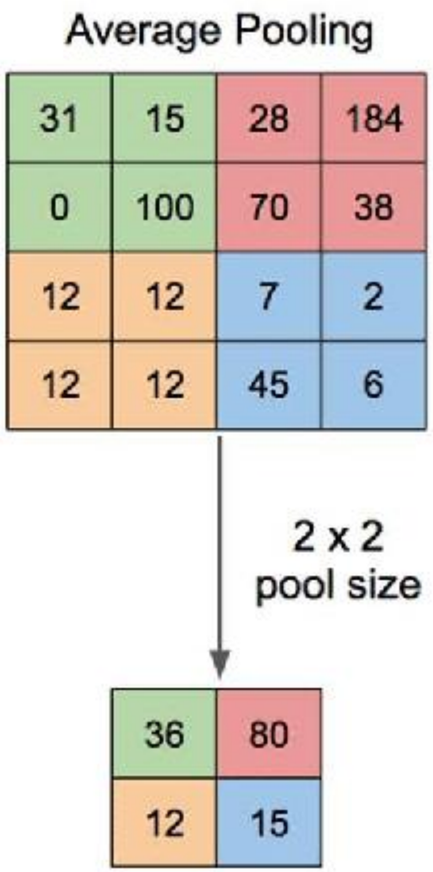
- Max Pooling also performs as a Noise Suppressant.
- It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.



Pooling Layer

- On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism.

- Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.



Pooling Layer

- The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network.

- Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

Pooling Layer

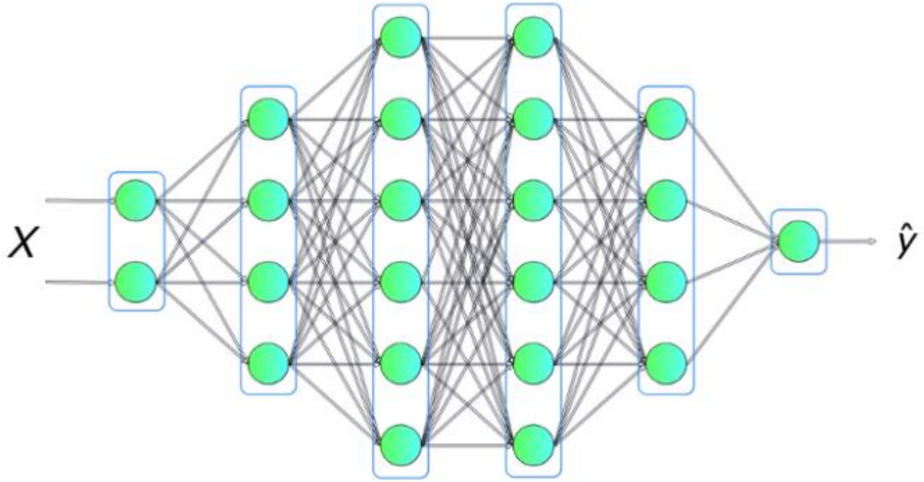
- After going through the above process, we have successfully enabled the model to understand the features.

- Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

Classification – Fully Connected Layer (FC Layer)

- Adding a Fully-Connected layer is a way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer.

- The Fully-Connected layer is learning a possibly non-linear function in that space.



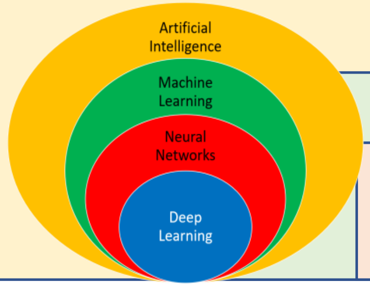
Classification – Fully Connected Layer (FC Layer)

- Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector.

- The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training.

- Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

Part 2: Activation functions



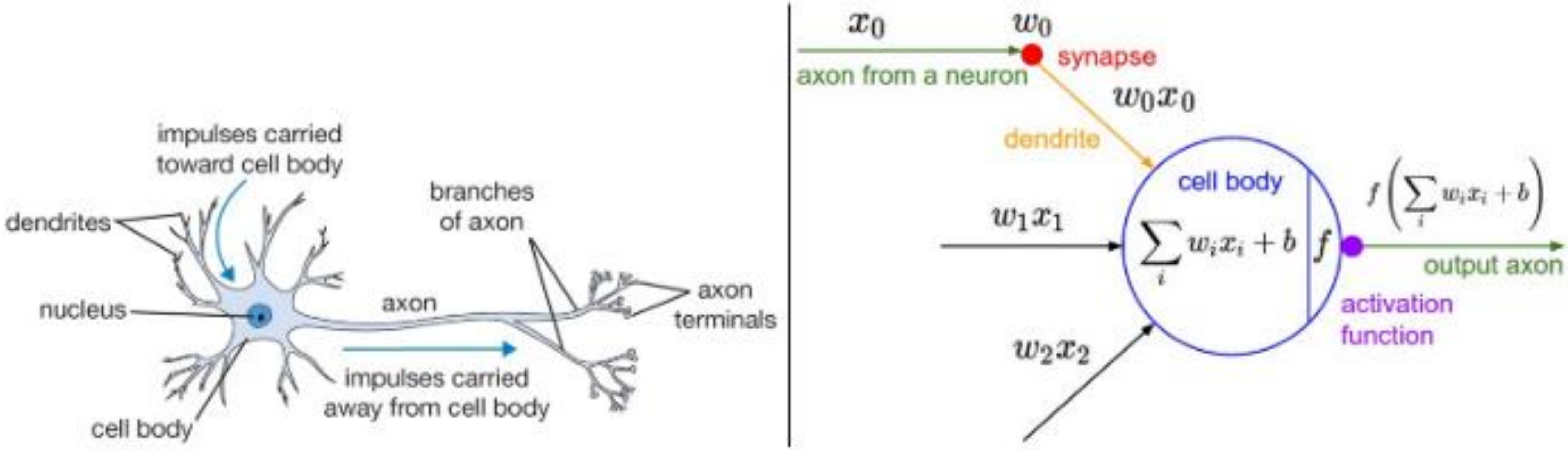
What is an activation function?

- An activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data.

- When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron.

- That is exactly what an activation function does in an ANN as well. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell.

What is an activation function?



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Why is there a need for it?

There are multiple reasons for having non-linear activation functions in a network.

- Apart from the biological similarity, they also help in keeping the value of the output from the neuron restricted to a certain limit as per our requirement.

- This is important because input into the activation function is W*x + b where W is the weights of the cell and the x is the inputs and then there is the bias b added to that.

- This value if not restricted to a certain limit can go very high in magnitude especially in case of very deep neural networks that have millions of parameters.

Why is there a need for it?

There are multiple reasons for having non-linear activation functions in a network.

- This will lead to computational issues.

- For example, there are some activation functions (like Softmax) that out specific values for different values of input (0 or 1).

- The most important feature in an activation function is its ability to add non-linearity into a neural network.

Desirable features of an activation function

- Vanishing Gradient problem: The gradients tend to vanish because of the depth of the network and the activation shifting the value to zero. This is called the vanishing gradient problem. So we want our activation function to not shift the gradient towards zero.

- Zero-Centered: Output of the activation function should be symmetrical at zero so that the gradients do not shift to a particular direction.

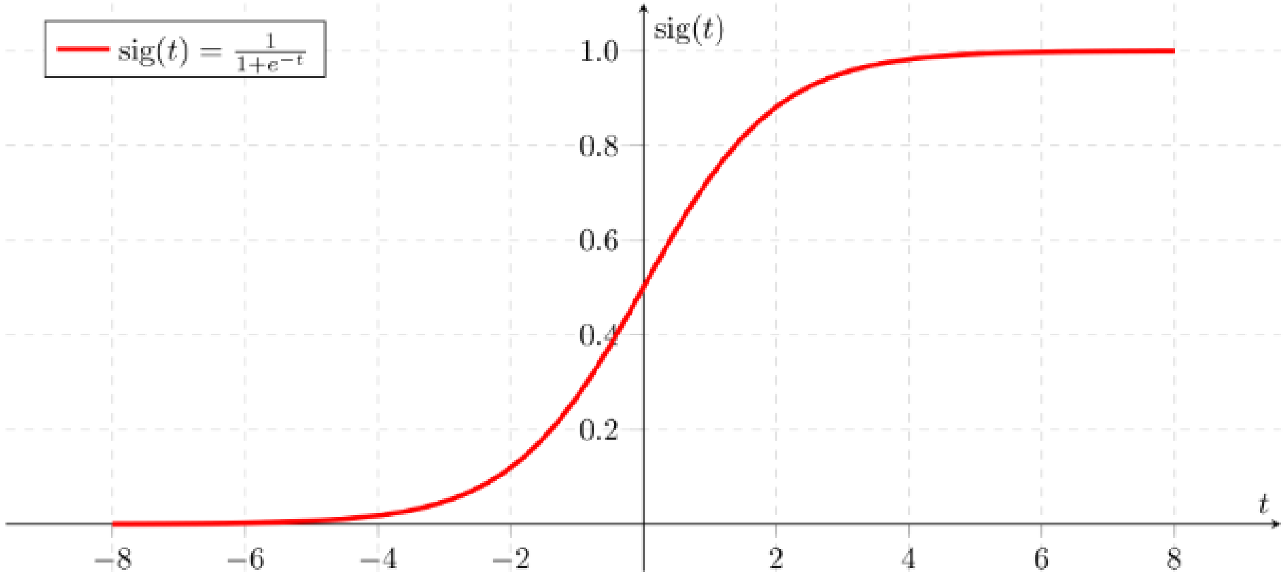
Desirable features of an activation function

- Computational Expense: Activation functions are applied after every layer and need to be calculated millions of times in deep networks. Hence, they should be computationally inexpensive to calculate.

- Differentiable: Neural networks are trained using the gradient descent process, hence the layers in the model need to differentiable or at least differentiable in parts. This is a necessary requirement for a function to work as activation function layer.

Various non-linear activations in use

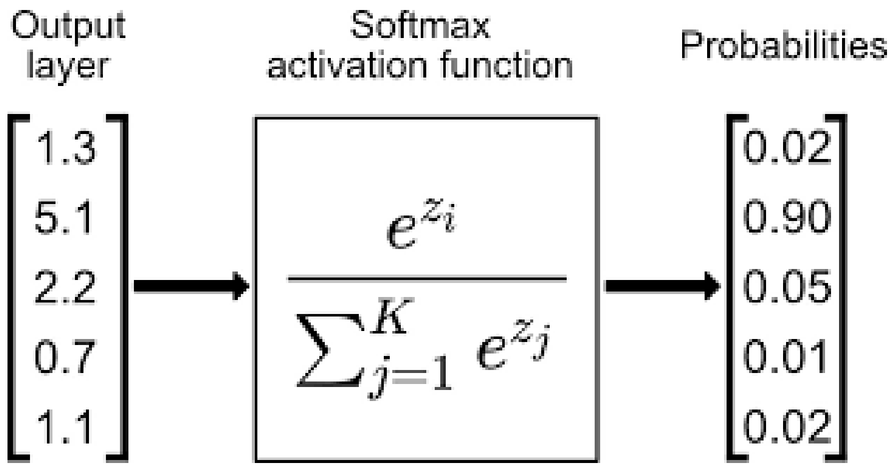
- Sigmoid

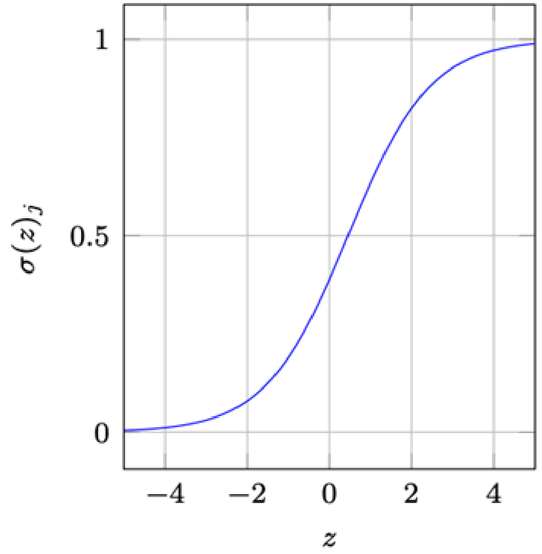


Sigmoid is generally used for binary classification problems.

Various non-linear activations in use

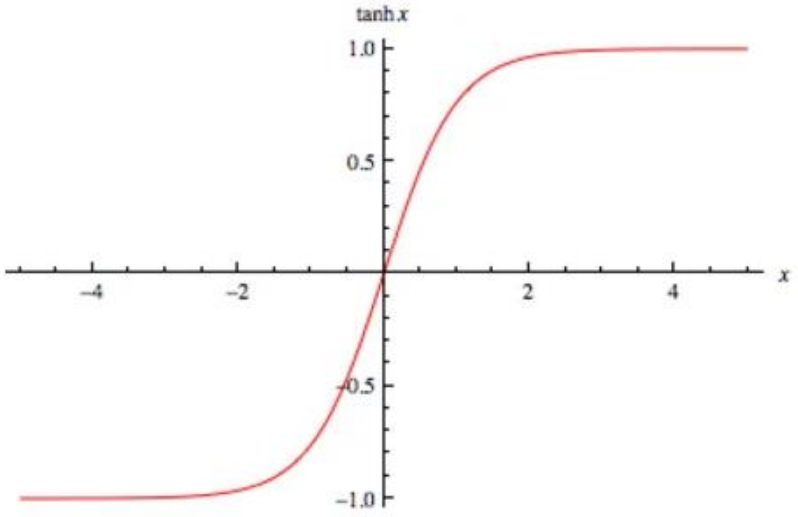
- Softmax: The Softmax is a more generalized form of the sigmoid. It is used in multi-class classification problems. Similar to sigmoid, it produces values in the range of 0–1 therefore it is used as the final layer in classification models.





Various non-linear activations in use

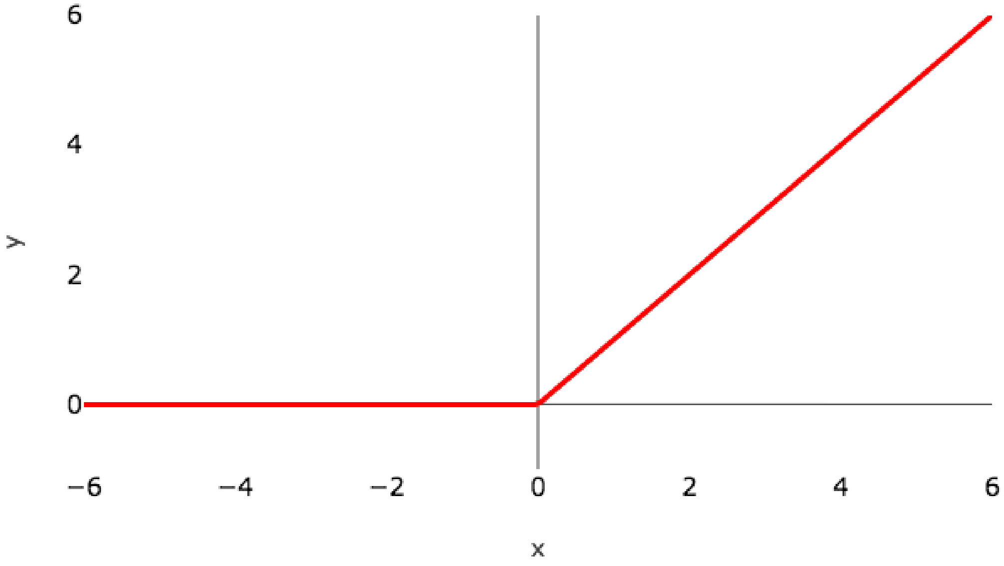
- Tanh



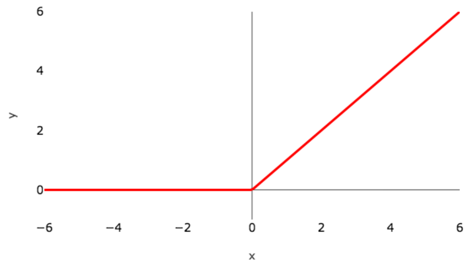
If you compare it to sigmoid, it solves just one problem of being zero-centered.

Various non-linear activations in use

- **ReLU:** ReLU (Rectified Linear Unit) is defined as f(x) = max(0,x)



Various non-linear activations in use

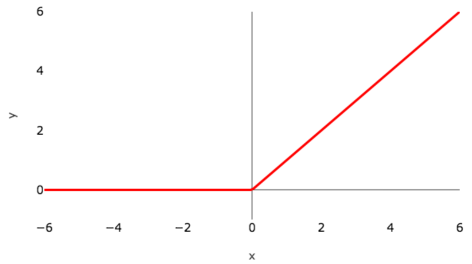


- ReLU: ReLU (Rectified Linear Unit) is defined as f(x) = max(0,x)

- This is a widely used activation function, especially with Convolutional Neural networks.

- It is easy to compute and does not saturate and does not cause the Vanishing Gradient Problem.

Various non-linear activations in use

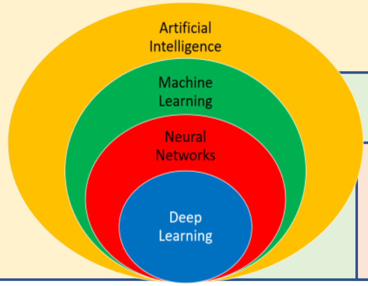


- ReLU: ReLU (Rectified Linear Unit) is defined as f(x) = max(0,x)

- It has just one issue of not being zero centered.

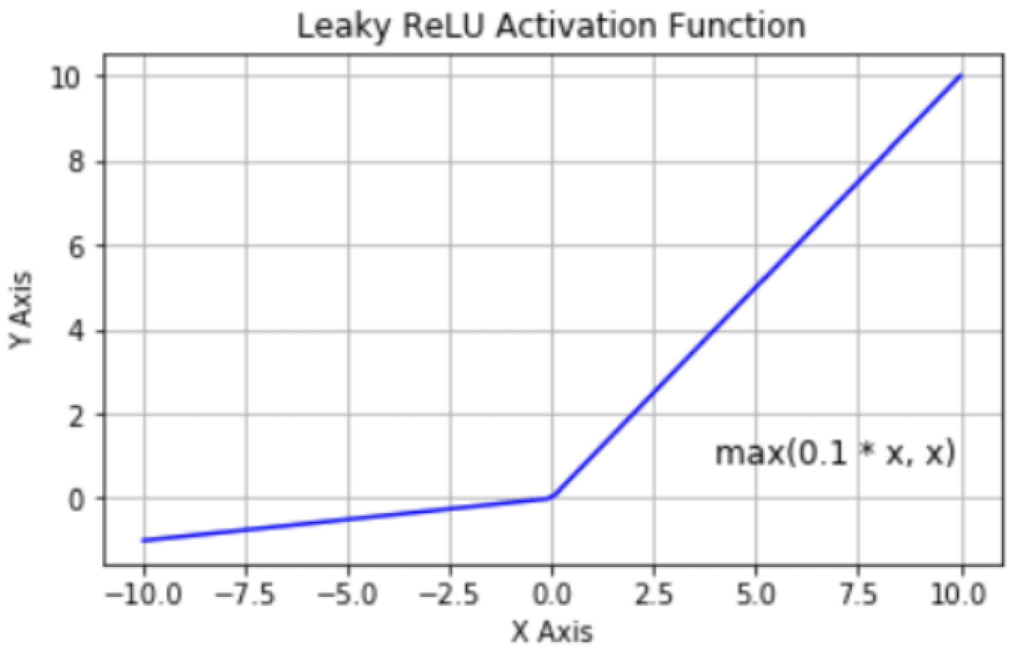
- It suffers from "dying ReLU" problem. Since the output is zero for all negative inputs. It causes some nodes to completely die and not learn anything.

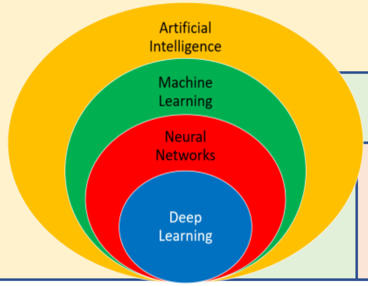
- Another problem with ReLU is of exploding the activations since it higher limit is, well, inf. This sometimes leads to unusable nodes.



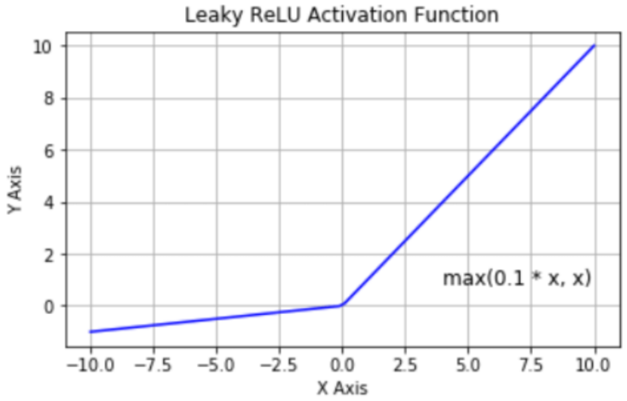
Various non-linear activations in use

- Leaky ReLU and Parametric ReLU: It is defined as $f(x) = \max(\alpha x, x)$

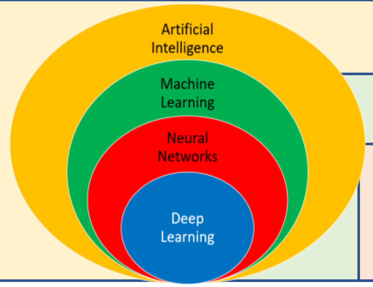




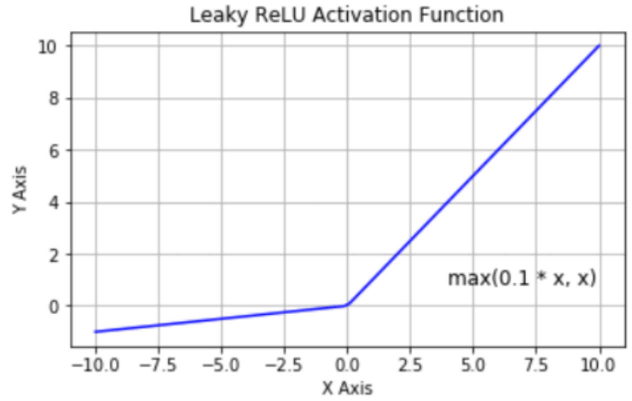
Various non-linear activations in use



- Leaky ReLU and Parametric ReLU: It is defined as f(x) = max(αx, x)
- Here α is a hyperparameter generally set to 0.01.
- Clearly, Leaky ReLU solves the "dying ReLU" problem to some extent.
- Note that, if we set α as 1 then Leaky ReLU will become a linear function f(x) = x and will be of no use.



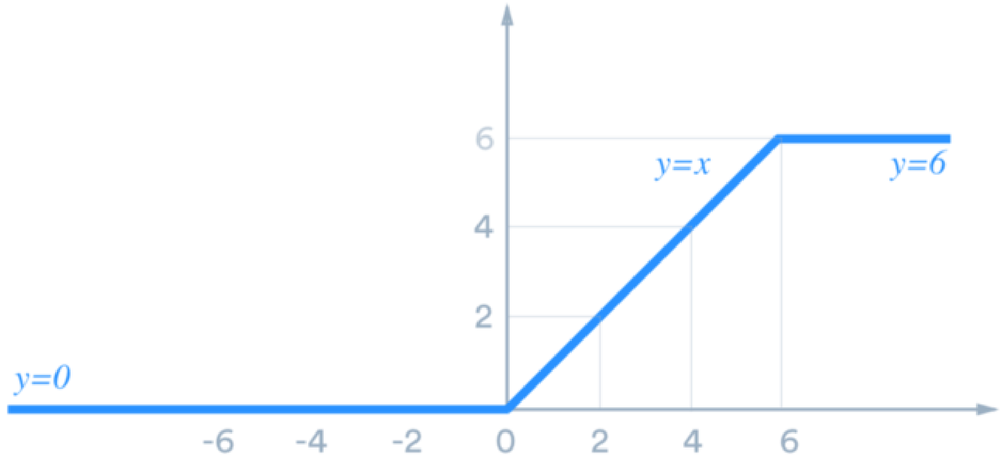
Various non-linear activations in use



- Leaky ReLU and Parametric ReLU: It is defined as $f(x) = \max(\alpha x, x)$
- Hence, the value of α is never set close to 1.
- If we set α as a hyperparameter for each neuron separately, we get parametric ReLU or PReLU.

Various non-linear activations in use

- ReLU6: It is basically ReLU restricted on the positive side and it is defined as f(x) = min(max(0,x),6)

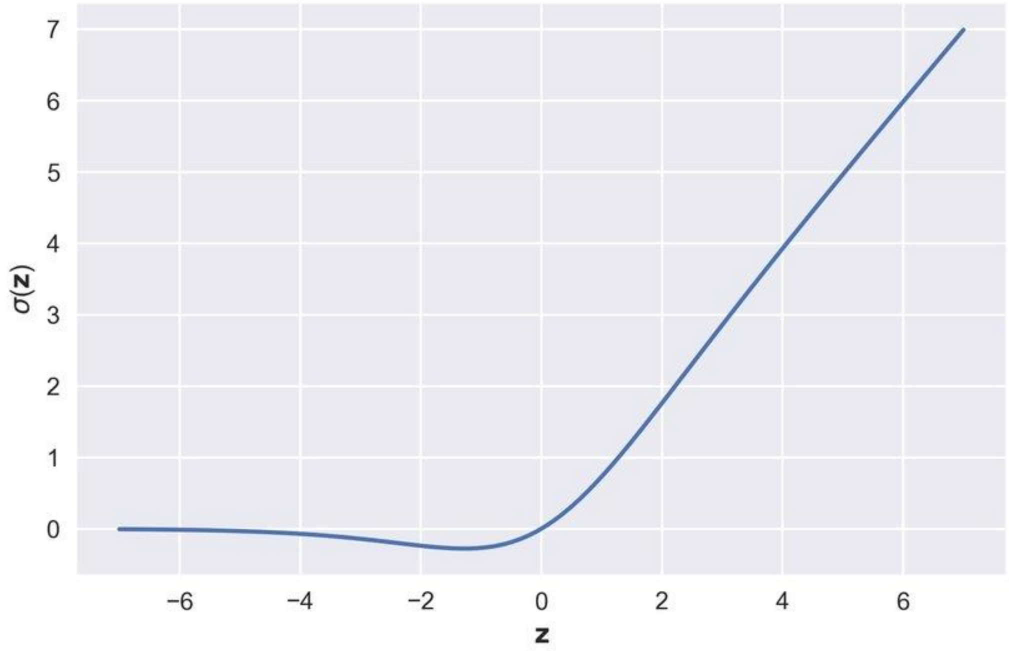


RELU — 6

- This helps to stop blowing up the activation thereby stopping the gradients to explode (going to inf) as well another of the small issues that occur with normal ReLUs.

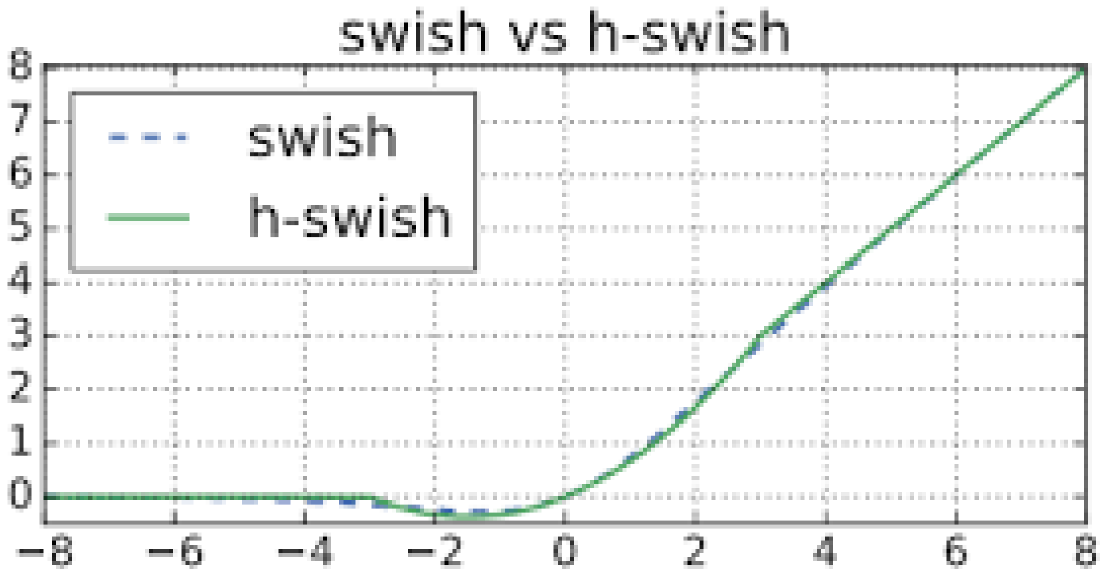
Various non-linear activations in use

- Swish: It is defined as f(x) = x*sigmoid(x).



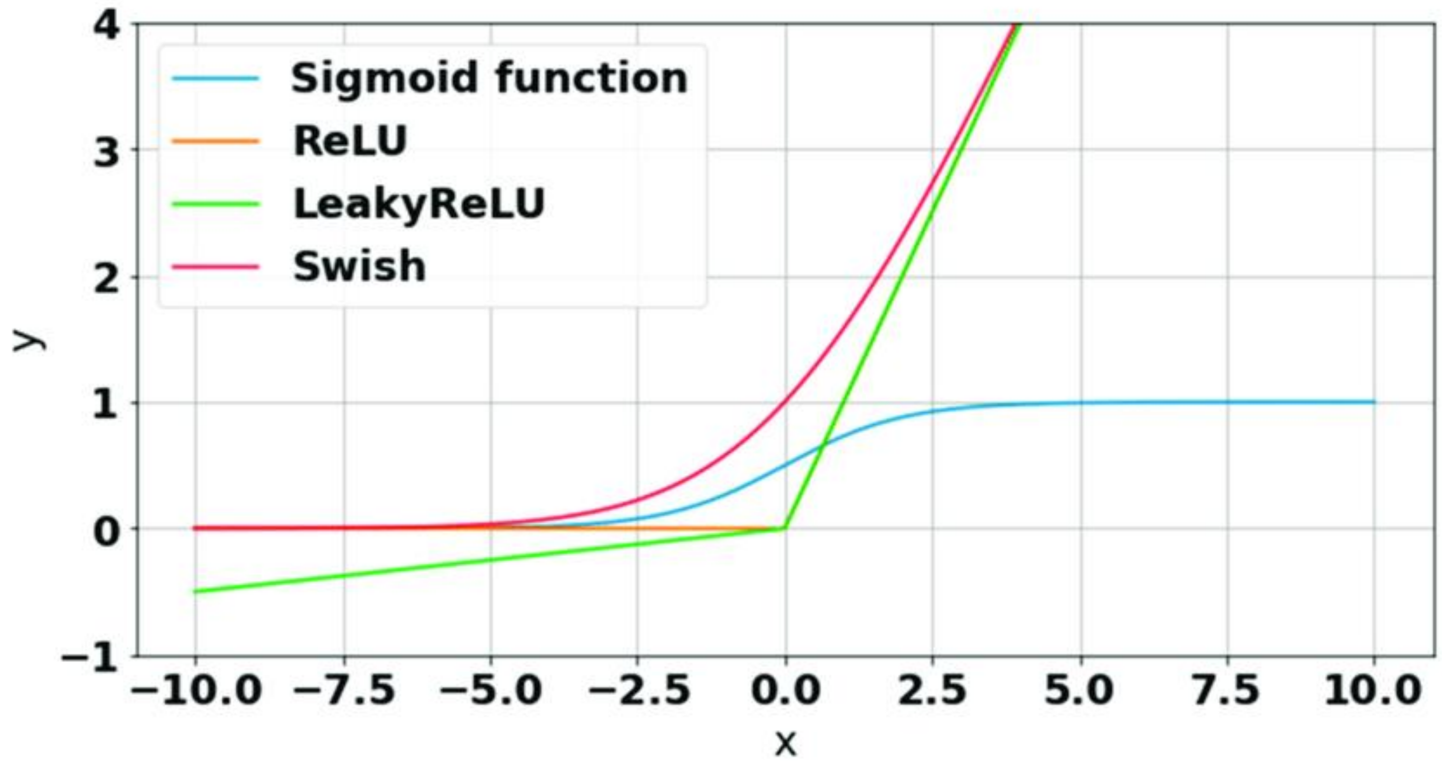
Various non-linear activations in use

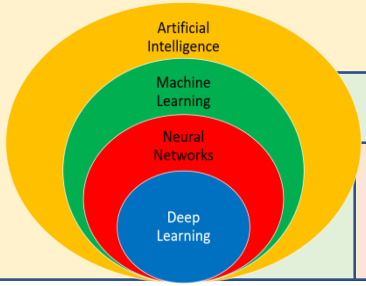
- Swish: Hard-Swish or H-Swish



- The best part is that it is almost similar to swish but it is less expensive computationally since it replaces sigmoid (exponential function) with a ReLU (linear type).

Various non-linear activations in use



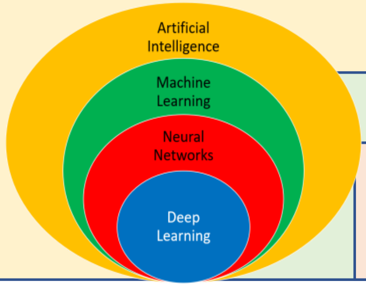


Part 3: Normalization

Normalization

- Imagine that we have two features and a simple neural network. One is age with a range between 0 and 65, and another is salary ranging from 0 to 10 000. We feed those features to the model and calculate gradients.

  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

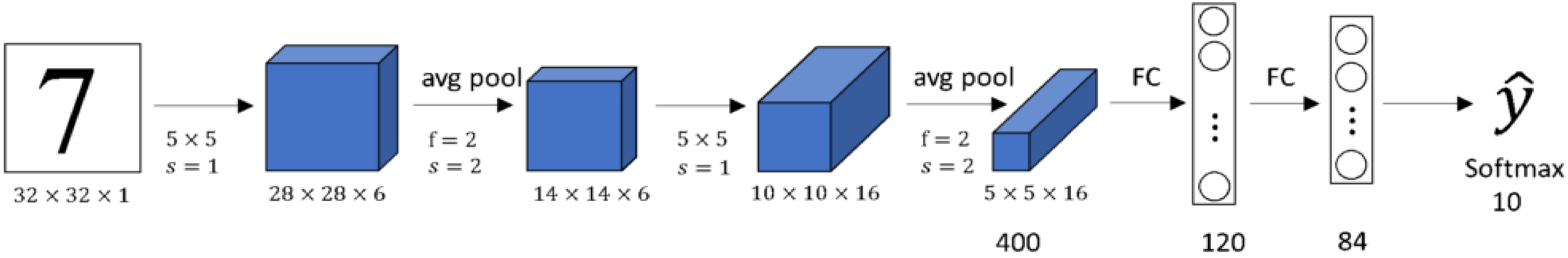


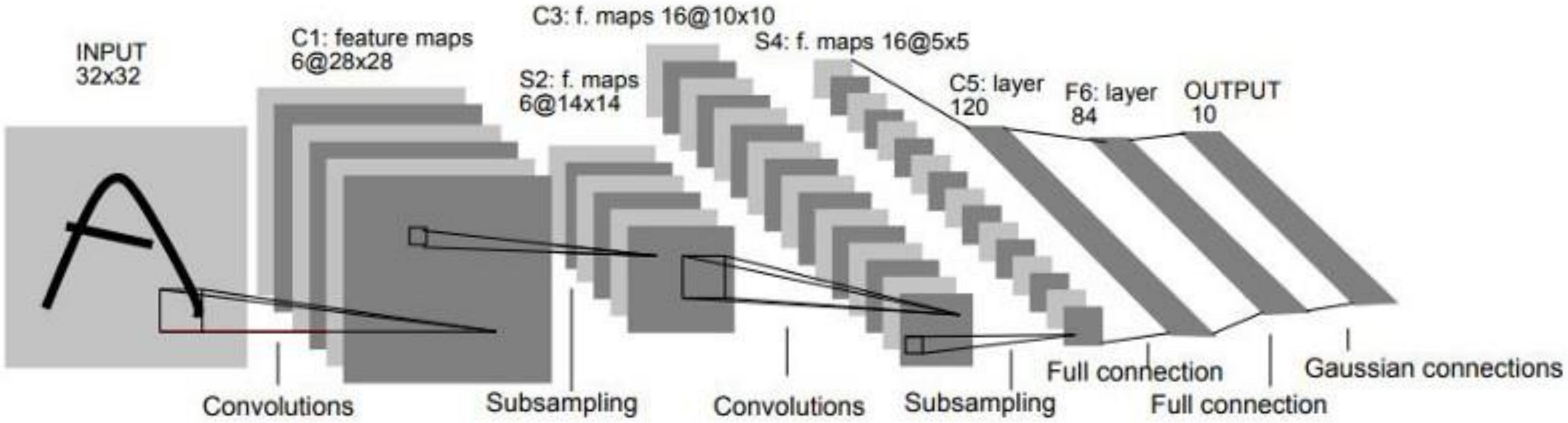
Part 4: Pre-trained Models

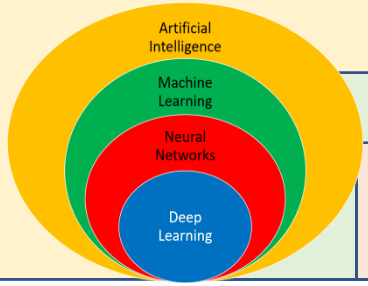
Pre-Trained CNNs

- There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:
  - LeNet
  - AlexNet
  - VGGNet
  - GoogLeNet
  - ResNet
  - ZFNet

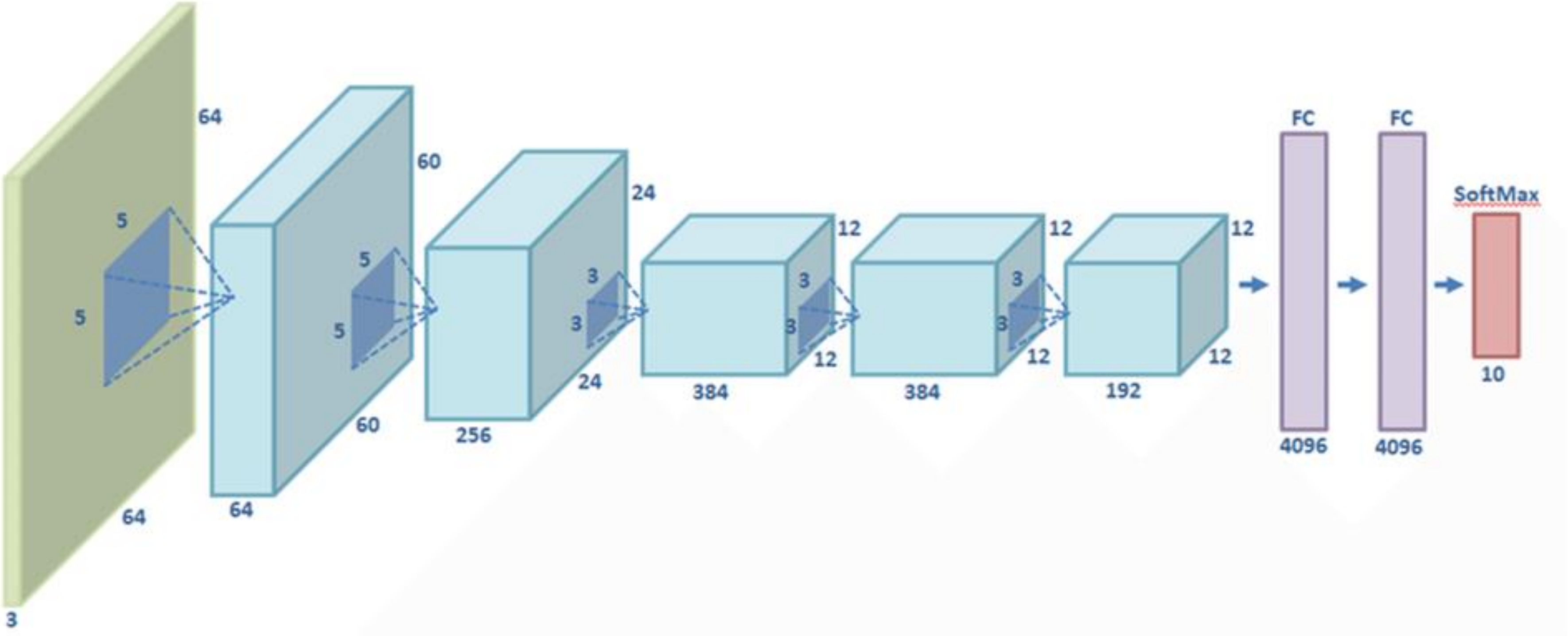
LeNet


7
32 × 32 × 1
5 × 5
s = 1
28 × 28 × 6
avg pool
f = 2
s = 2
14 × 14 × 6
5 × 5
s = 1
10 × 10 × 16
avg pool
f = 2
s = 2
5 × 5 × 16
400
FC
120
FC
84
ŷ
Softmax
10

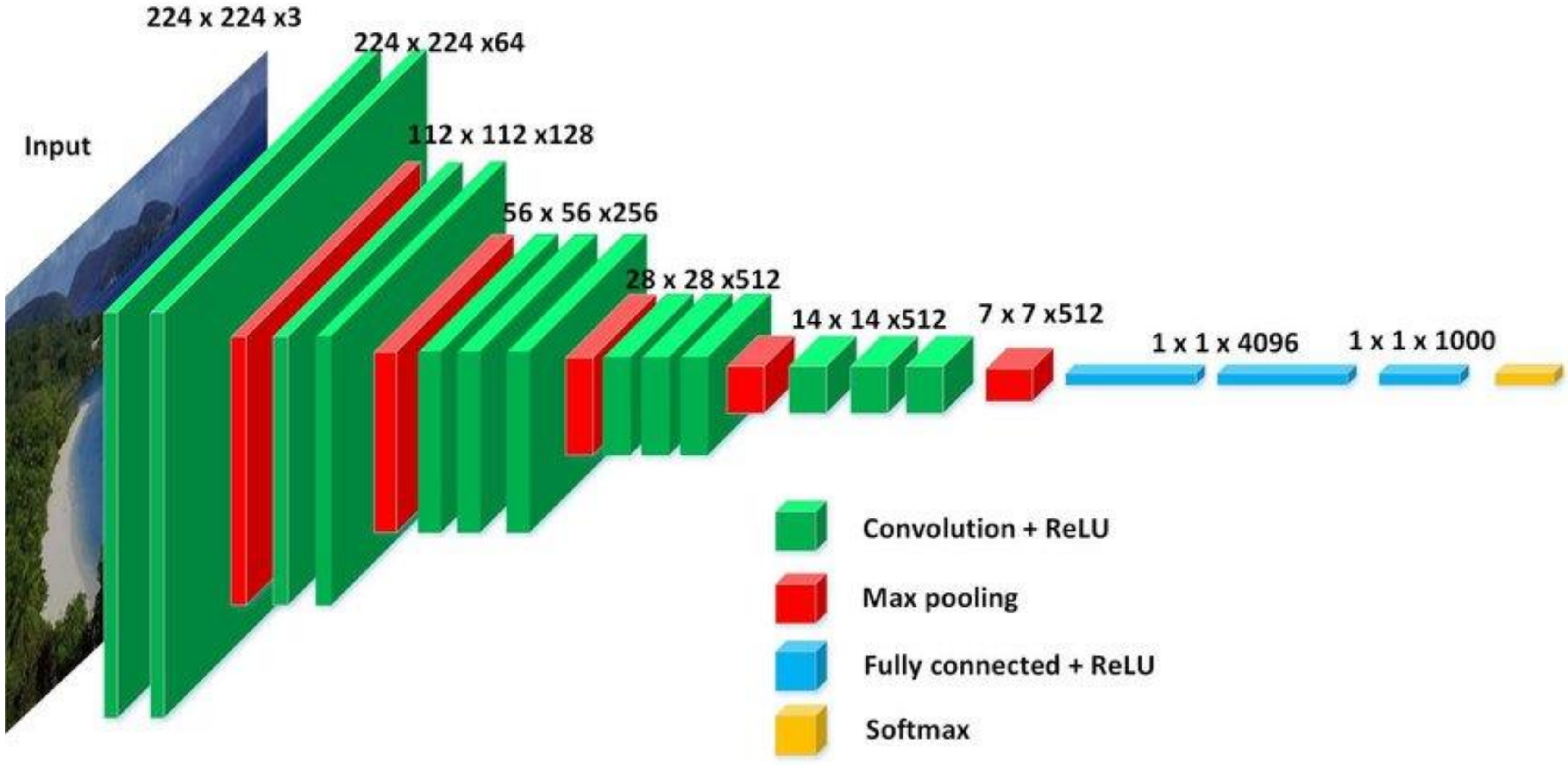

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


Artificial Intelligence
Machine Learning
Neural Networks
Deep Learning

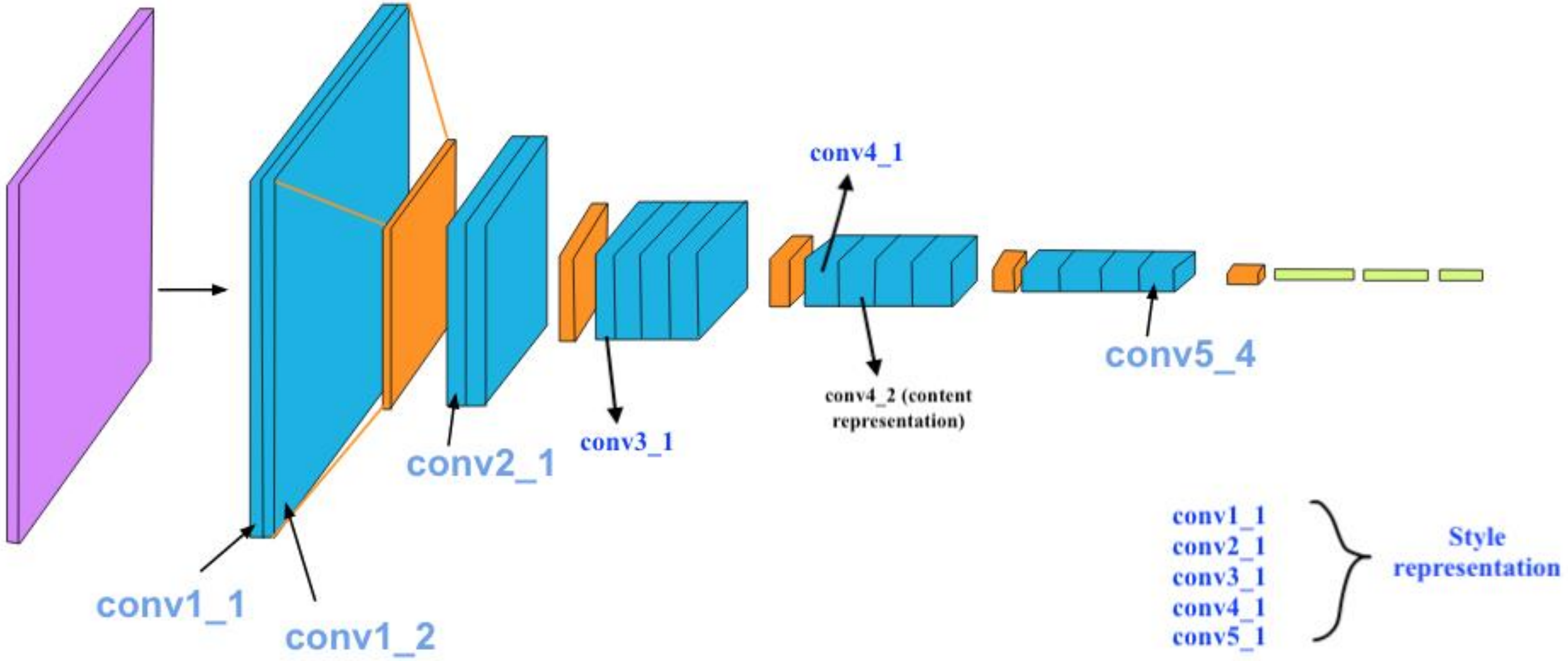
AlexNet


64
5
5
64
3
60
5
5
60
64
24
3
3
24
256
12
3
3
12
384
12
3
3
12
384
12
12
192
FC
4096
FC
4096
SoftMax
10

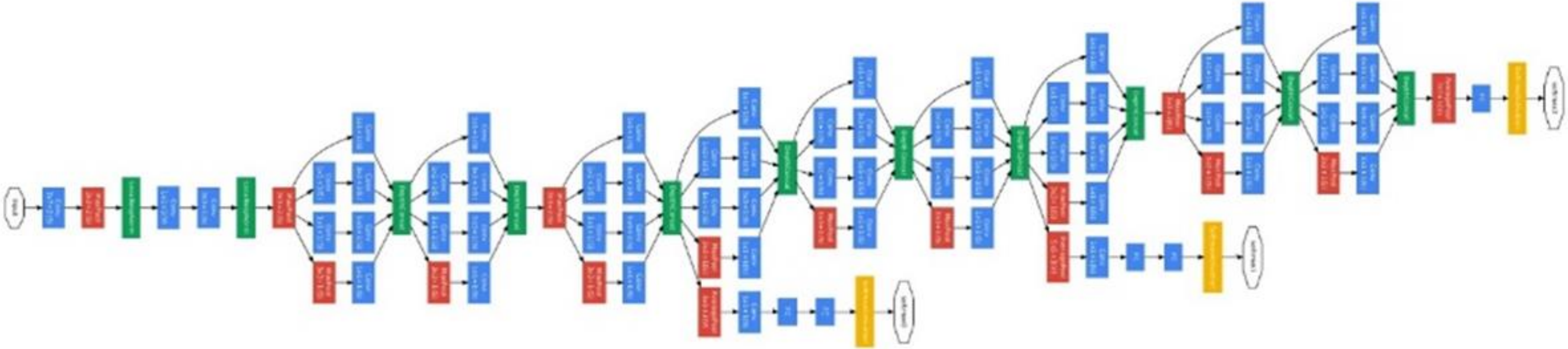
VGG16


224 x 224 x3
224 x 224 x64
Input
112 x 112 x128
56 x 56 x256
28 x 28 x512
14 x 14 x512
7 x 7 x512
1 x 1 x 4096
1 x 1 x 1000
Convolution + ReLU
Max pooling
Fully connected + ReLU
Softmax

VGG19

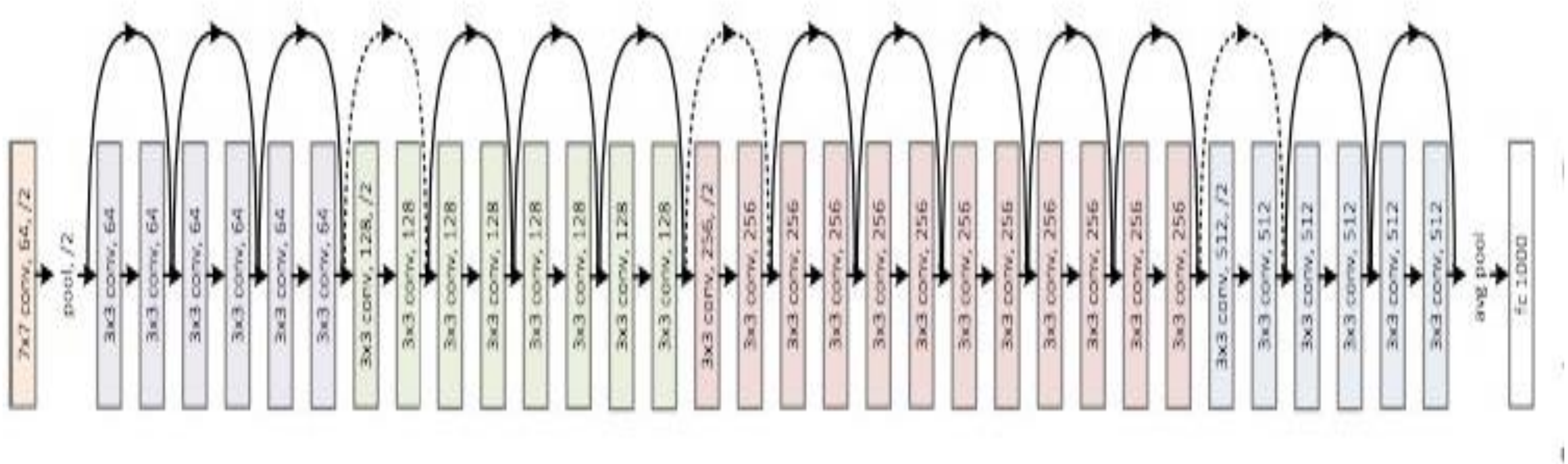

conv4_1
conv5_4
conv4_2 (content representation)
conv3_1
conv2_1
conv1_1
conv1_2
conv1_1
conv2_1
conv3_1
conv4_1
conv5_1
Style representation

GoogleNet (Inception v1)



22 layers deep

ResNet50


7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000