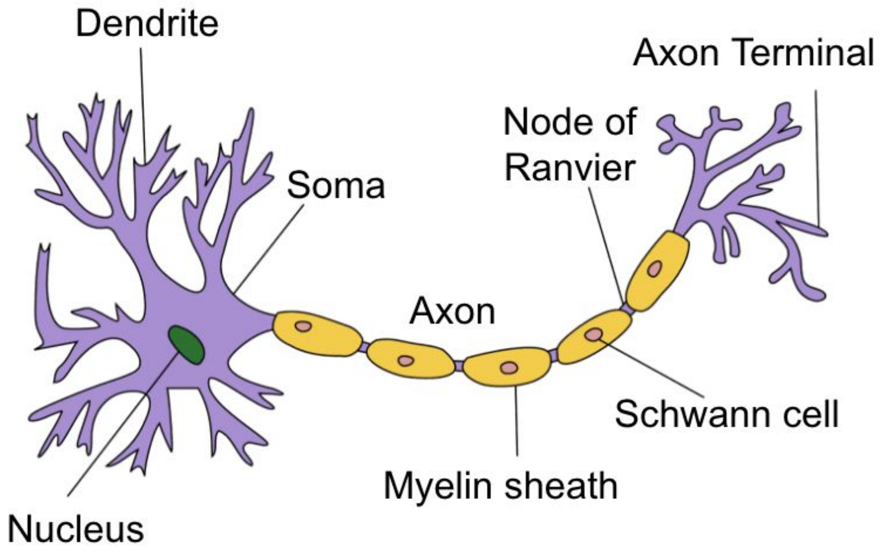
Deep Learning

LIS

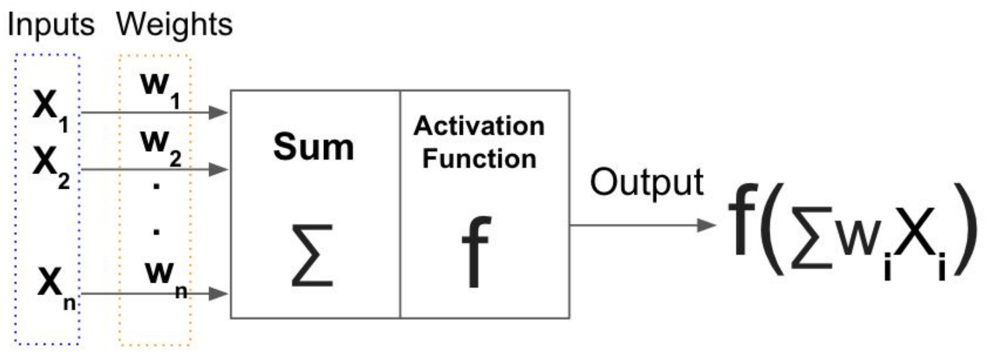
Jessie Li

March 2024

Artificial Neural Network (NN) - the basics



Structure of a typical neuron
(source: Wikipedia)



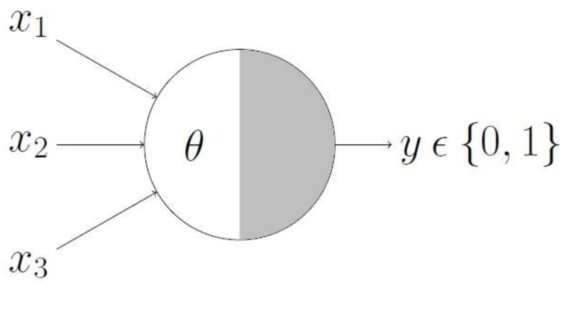
Structure of artificial neuron

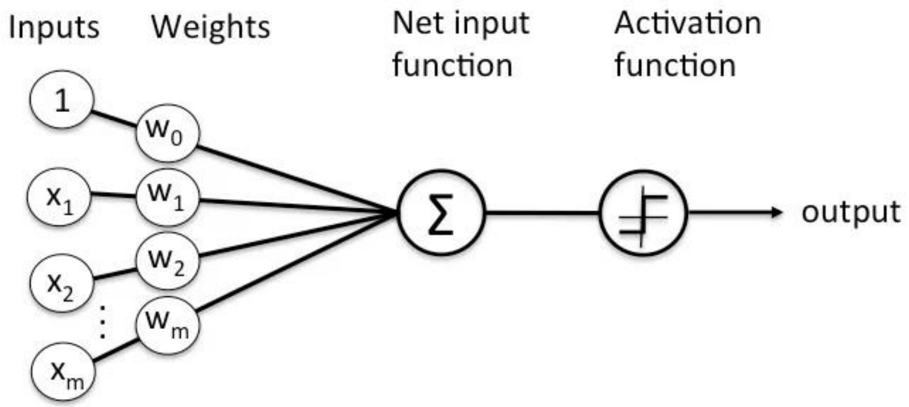
http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

History of neural network

1943: Neural networks were first proposed in 1944 by **Warren McCullough (neuroscientist) and Walter Pitts (logician)**, two University of Chicago researchers. NN had thresholds and weights, but they weren't arranged into layers, and the researchers didn't specify any training mechanism.

1958: The first trainable neural network, the **Perceptron**, was demonstrated by the Cornell University psychologist **Frank Rosenblatt.** The Perceptron's design was much like that of the modern neural net, except that it had only one layer with adjustable weights and bias, sandwiched between input and output layers.





https://medium.com/@karthikeyana97/history-of-neural-network-d1333760f0c5

1969-1986: Winter of NN

Perceptrons were an active area of research in both psychology and the fledgling discipline of computer science until 1969, when **Minsky and Papert** published a book titled “Perceptrons,” which demonstrated that executing certain fairly common computations on Perceptrons would be impractically time consuming.

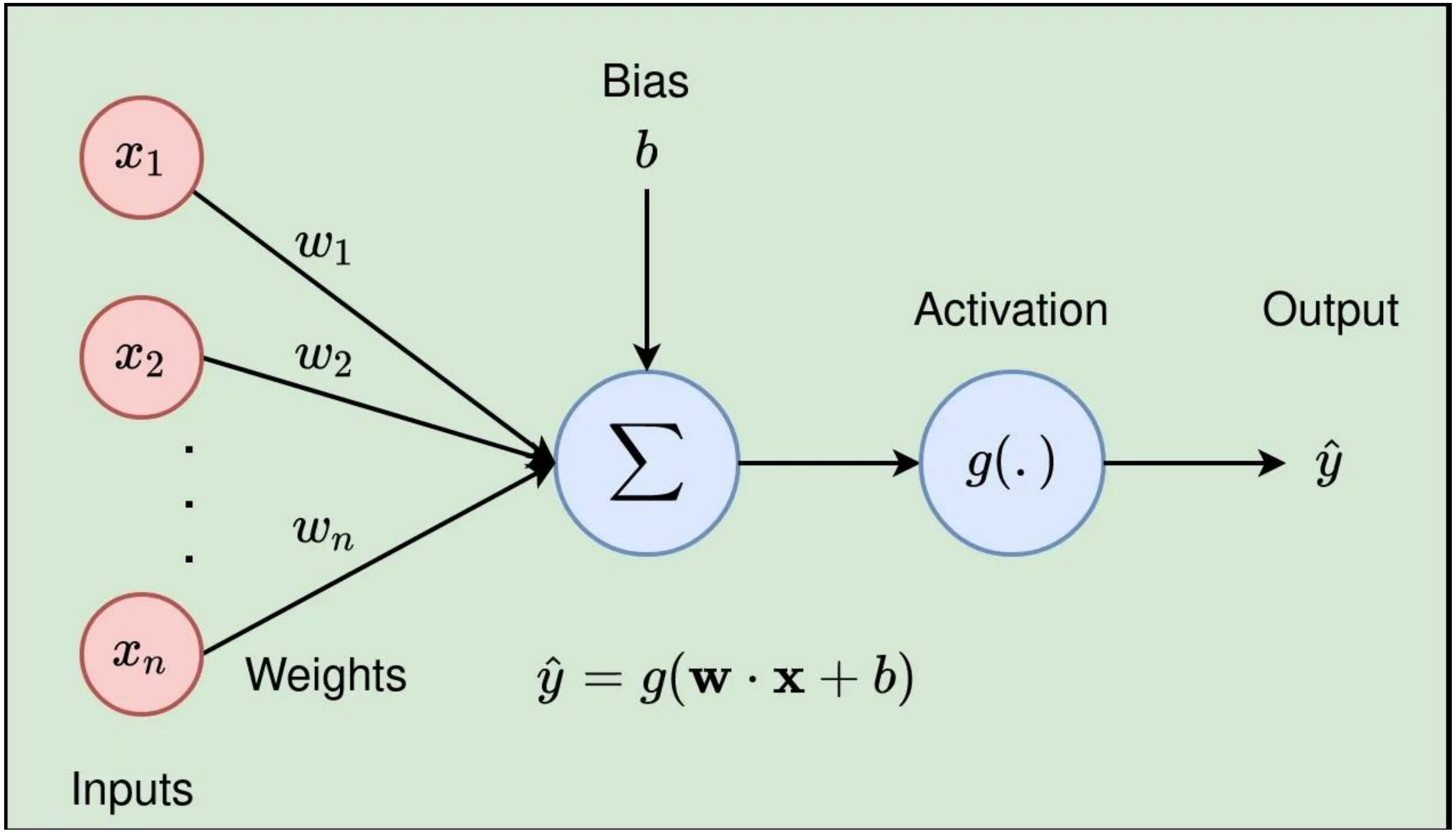
Backpropagation - 1986

Later, advances in hardware and the development of the backpropagation algorithm as well as recurrent neural networks and convolutional neural networks, renewed interest in ANNs.

Basics of continuous backpropagation were derived in the context of control theory by Henry J. Kelley in 1960. Arthur E. Bryson in 1961 used principles of dynamic programming. In 1970 Linnainmaa published the general method for automatic differentiation (AD) of discrete connected networks of nested differentiable functions (Base for current Backpropagation algorithm).

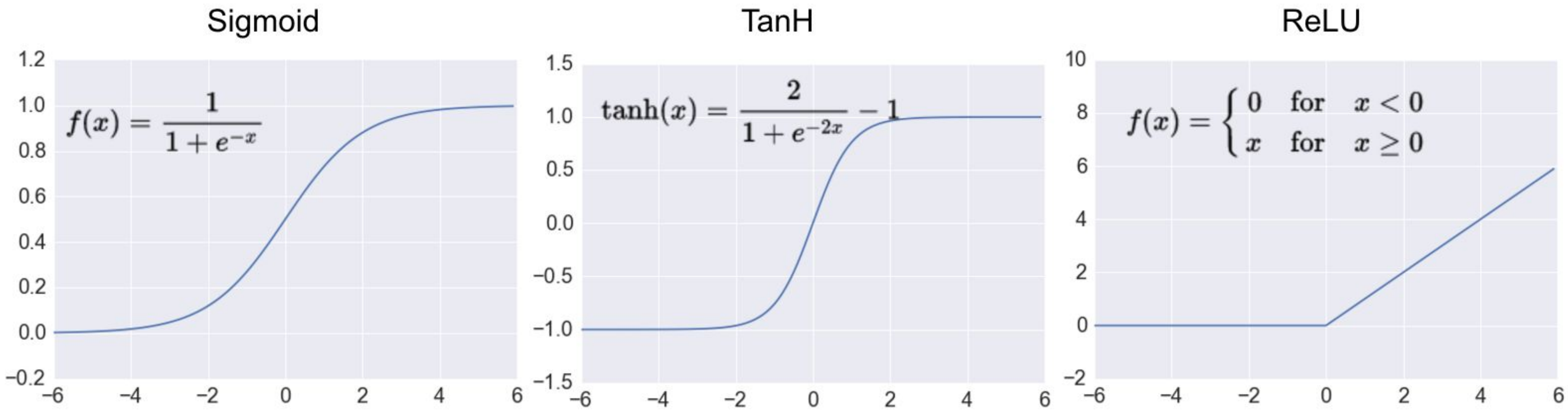
1986 Rumelhart, Hinton and Williams showed experimentally that this method can generate useful internal representations of incoming data in hidden layers of neural networks.

Backpropagation computes the gradient of a loss function with respect to the weights of the network for a single input–output example, and does so efficiently, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule; this can be derived through dynamic programming. Gradient descent, or variants such as stochastic gradient descent, are commonly used.



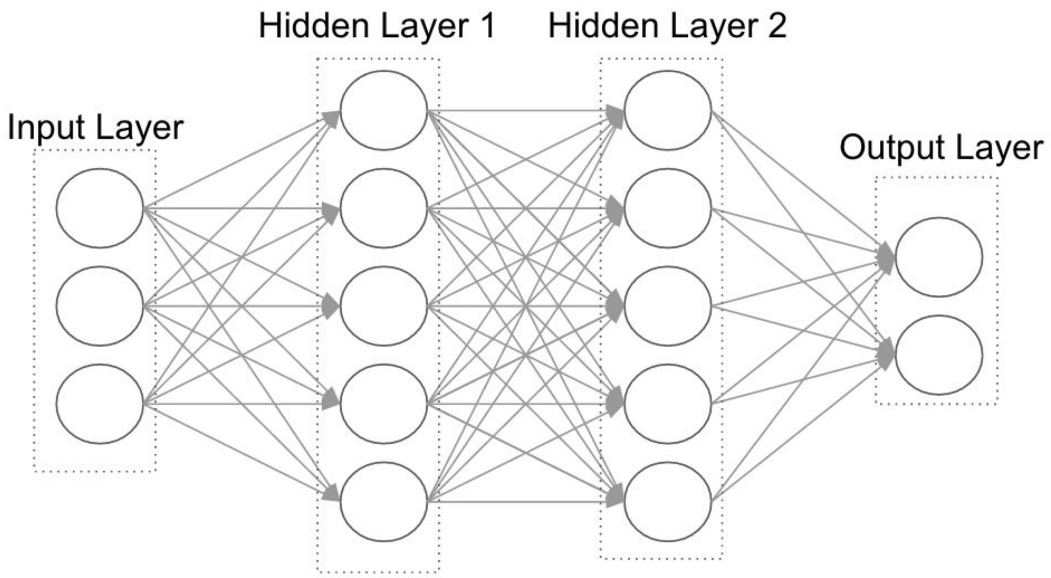
https://towardsdatascience.com/the-basics-of-neural-networks-neural-network-series-part-1-4419e343b2b

Activation functions



http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Feedforward neural network with two hidden layers



Feedforward neural network with 2 hidden layers

http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Backpropagation (explanation with and without equations)

Backpropagation is an optimization algorithm used for training neural networks. By assessing the discrepancy between the actual output and the desired output of the network, backpropagation efficiently computes the gradient of the **cost function**. This gradient is used to update the weights and biases in the network, thereby reducing errors and improving the network's performance.

The **chain rule** is an essential mathematical tool that backpropagation utilizes to compute the partial derivatives of the cost function with respect to each weight in the network.

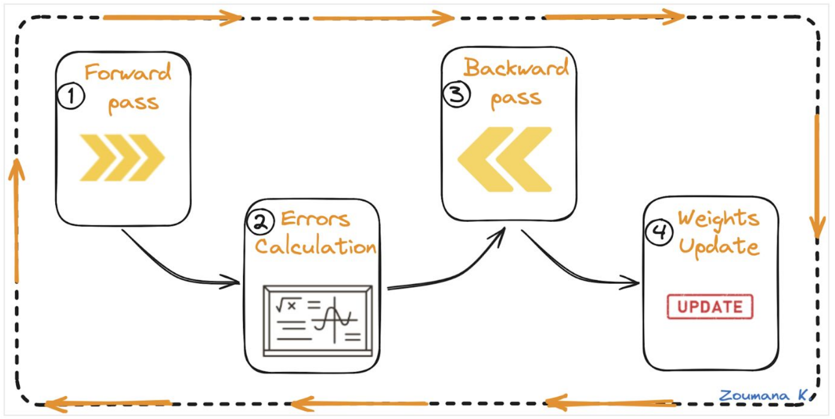
In the context of backpropagation, **gradient descent** plays a crucial role as the optimization algorithm that drives the update of weights and biases across the network. Acting on the gradient computed by backpropagation, it determines the direction in which the weights should be adjusted to minimize the cost function.

The **learning rate** is a crucial parameter in the backpropagation algorithm, dictating the size of the steps taken during the gradient descent process. Too large a learning rate can cause the network to oscillate or diverge, while too small a rate can slow down the learning process significantly.

Each iteration (epoch) has two main phases:

In the forward pass, input data is passed through the network, layer by layer, until it reaches the output layer. Each neuron in the network processes the input by applying an activation function.

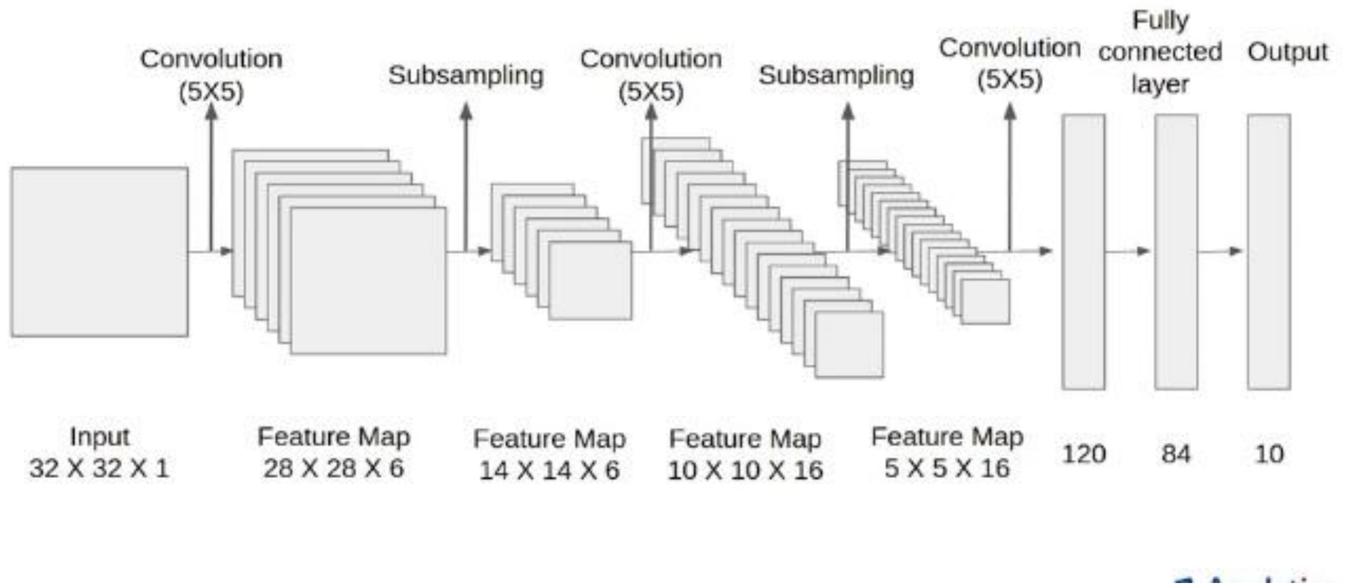
During the backward pass, the algorithm calculates the gradient of the loss function (error) at the output and propagates it backward through the network to update the weights, using gradient descent or a similar optimization algorithm.



Forward pass, error calculation, backward pass, and weights update

https://www.datacamp.com/tutorial/mastering-backpropagation

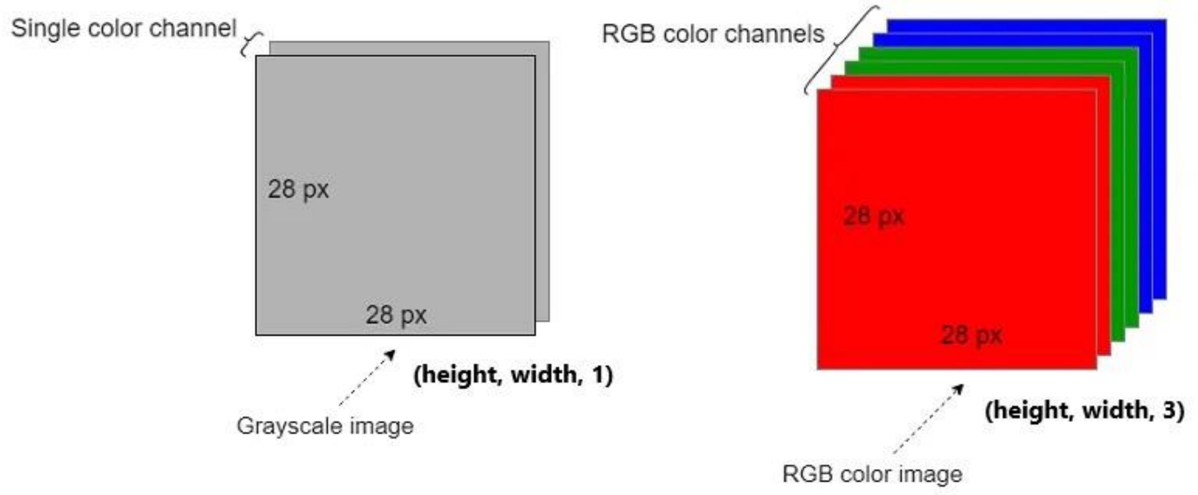
Convolutional NN (CNN) - LeNet (1990)



"Handwritten Digit Recognition with a Back-Propagation Network" by Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. NIPS 1990

CNN basics

Motivation: To use MLPs with images, we need to flatten the image. If we do so, spatial information (relationships between the nearby pixels) will be lost. So, accuracy will be reduced significantly. CNNs can retain spatial information as they take the images in the original format.

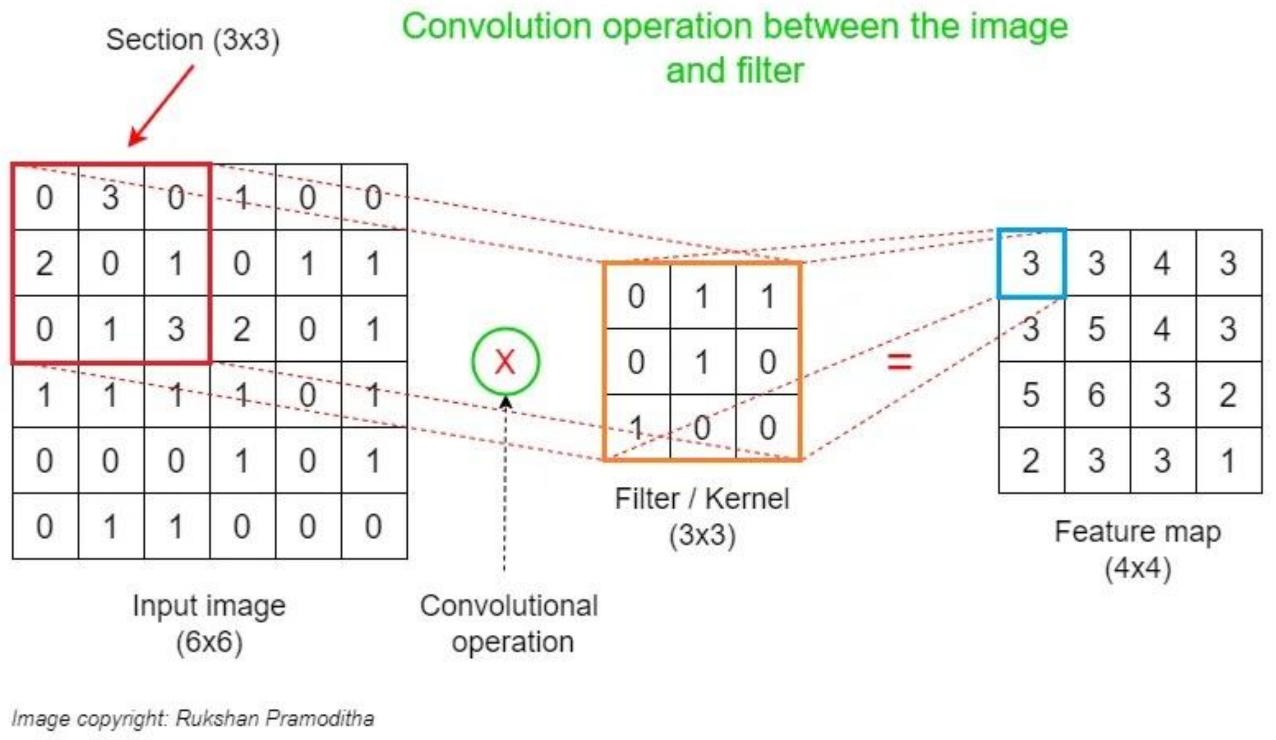


https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

Three main types of layers

- Convolutional layer
- Pooling layer
- Fully-connected (FC) / dense layer: A dense layer is a fully connected layer used in the neural network's end stages to change the output's dimensionality from the preceding layer.

Convolutional layer



0	3	0	1	0	0
2	0	1	0	1	1
0	1	3	2	0	1
1	1	1	1	0	1
0	0	0	1	0	1
0	1	1	0	0	0

0	1	1
0	1	0
1	0	0

3	3	4	3
3	5	4	3
5	6	3	2
2	3	3	1

https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

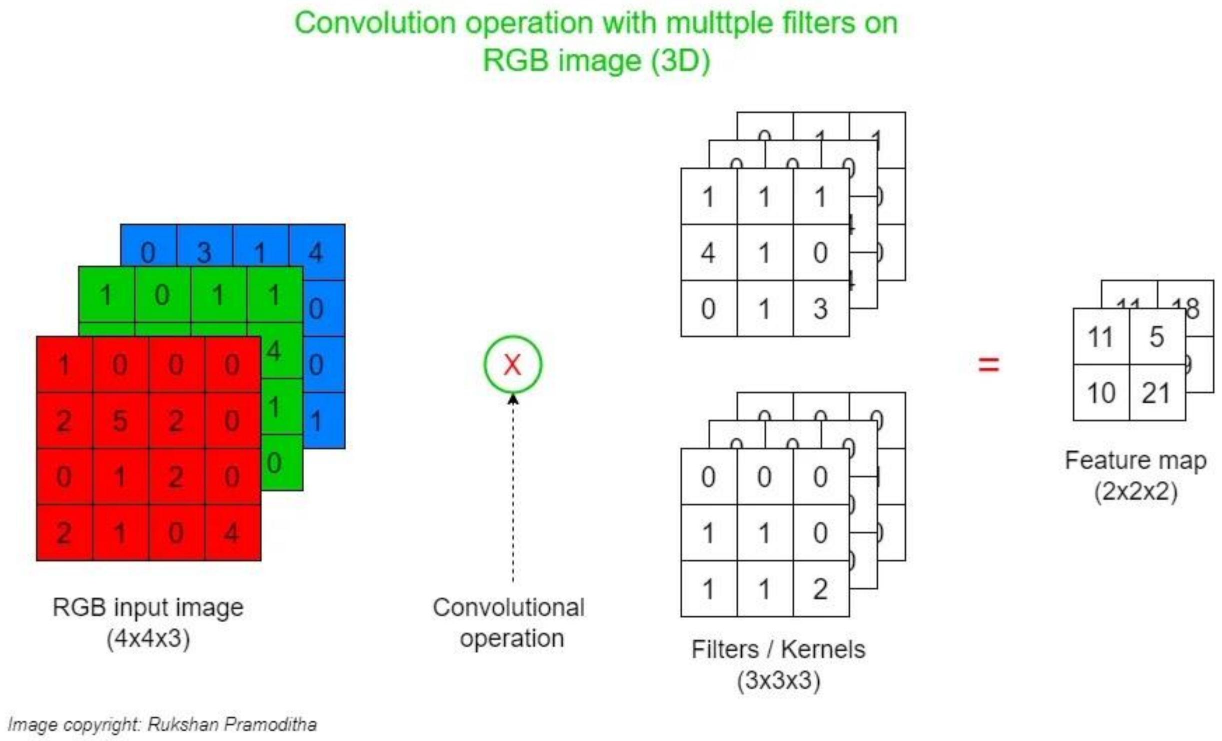
Stride & padding

Stride: The number of steps (pixels) that we shift the filter over the input image is called Stride. The shift can be done both horizontally and vertically.

Padding: If there are several convolutional layers in the CNN, the size of the feature map will be further reduced at the end so that we cannot do other operations on the feature map. To avoid this, we use apply Padding to the input image. It adds additional pixels with zero values to each side of the image. That helps to get the feature map of the same size as the input.

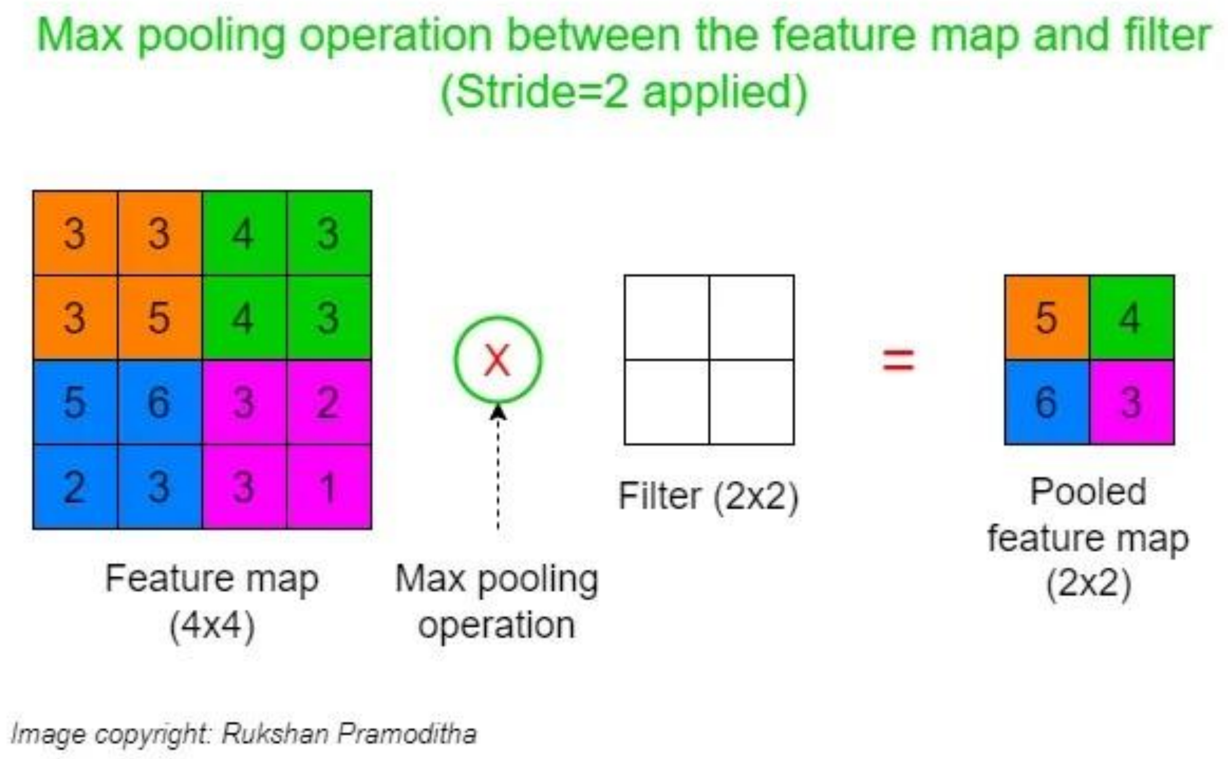
Receptive field: the Receptive Field (RF) is defined as the size of the region in the input that produces the feature. Basically, it is a measure of association of an output feature (of any layer) to the input region (patch).

Convolutional operation with multiple filters on RGB


Convolution operation with multtple filters on RGB image (3D)
RGB input image (4x4x3)
Convolutional operation
Filters / Kernels (3x3x3)
Feature map (2x2x2)
Image copyright: Rukshan Pramoditha

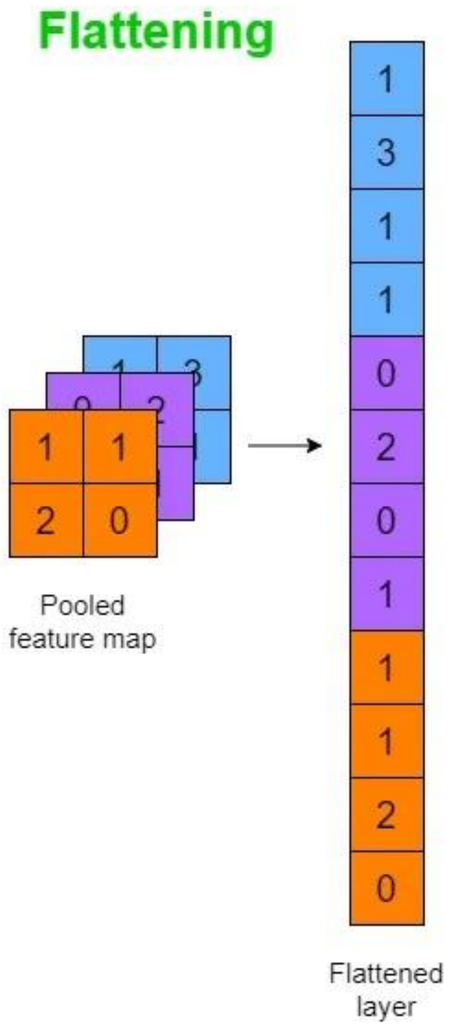
https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

Pooling



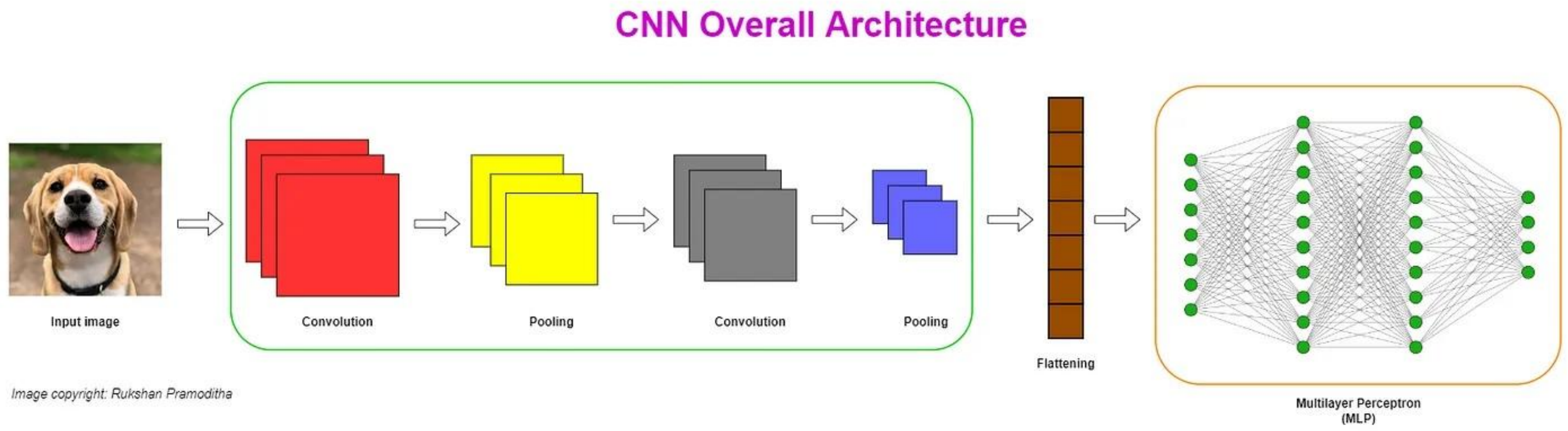
https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

Flattening



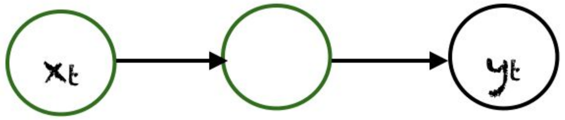
https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

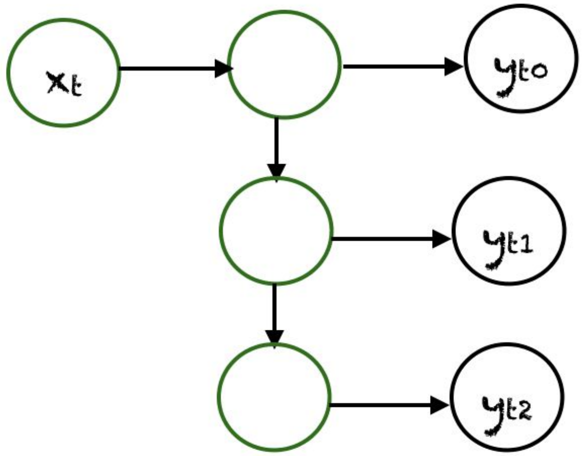
Final overall architecture

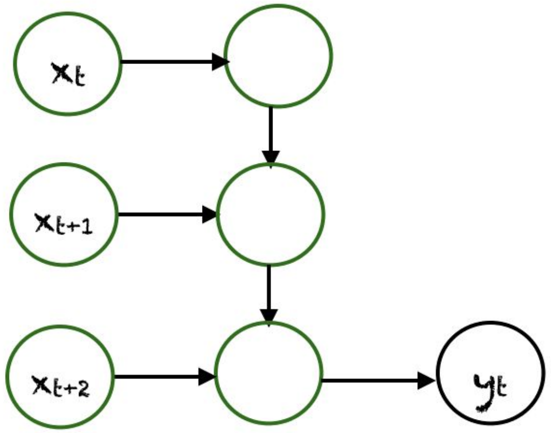


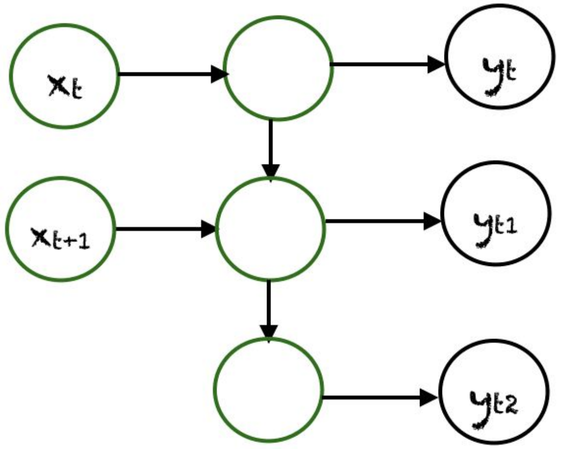
https://medium.com/towards-data-science/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f

Recurrent NN

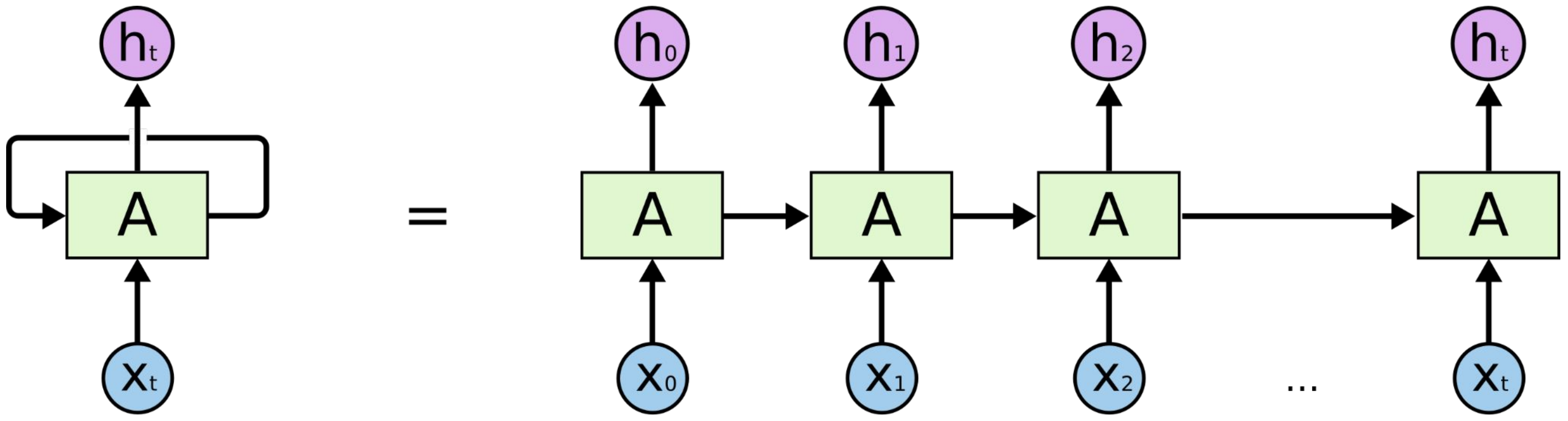






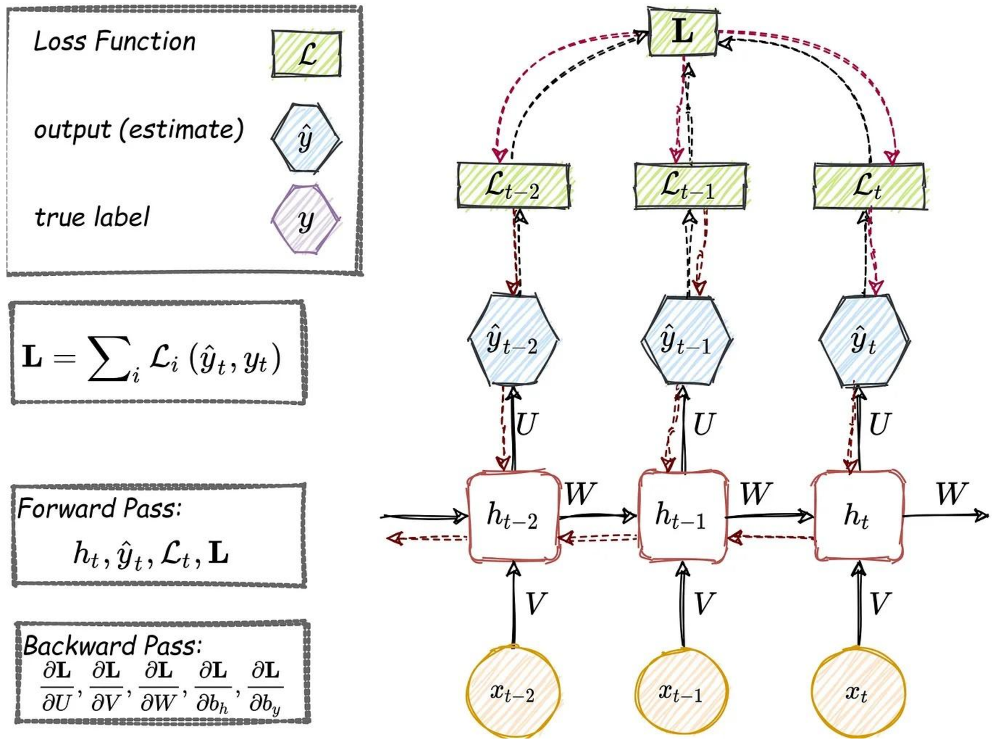


Recurrent NN



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

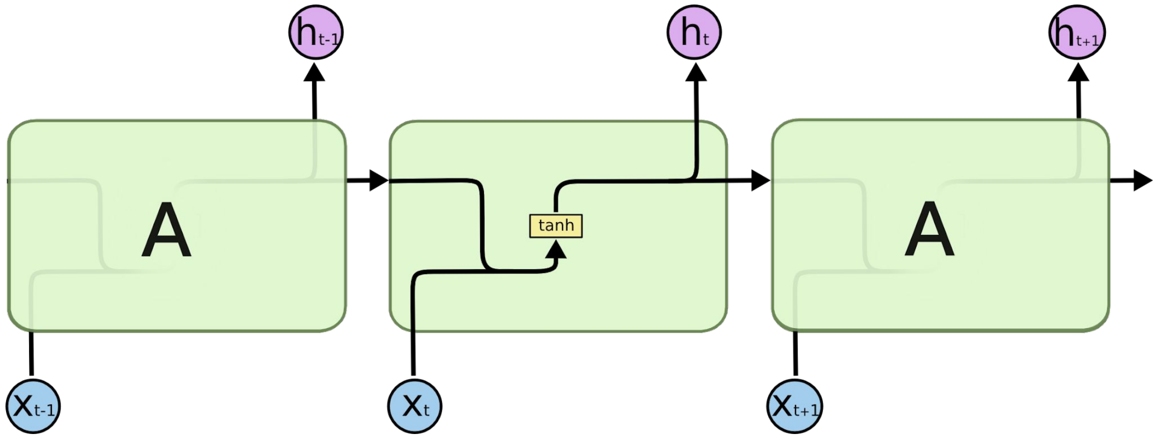
The vanishing or exploding gradient problem


Loss Function $\mathcal{L}$
output (estimate) $\hat{y}$
true label y
$\mathbf{L} = \sum_i \mathcal{L}_i(\hat{y}_t, y_t)$
Forward Pass: $h_t, \hat{y}_t, \mathcal{L}_t, \mathbf{L}$
Backward Pass: $\frac{\partial \mathbf{L}}{\partial U}, \frac{\partial \mathbf{L}}{\partial V}, \frac{\partial \mathbf{L}}{\partial W}, \frac{\partial \mathbf{L}}{\partial b_h}, \frac{\partial \mathbf{L}}{\partial b_y}$
$\mathbf{L}$
$\mathcal{L}_{t-2}$
$\mathcal{L}_{t-1}$
$\mathcal{L}_t$
$\hat{y}_{t-2}$
$\hat{y}_{t-1}$
$\hat{y}_t$
U
h_{t-2}
h_{t-1}
h_t
W
V
x_{t-2}
x_{t-1}
x_t

$$\frac{\partial \mathbf{L}}{\partial W} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}_i}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

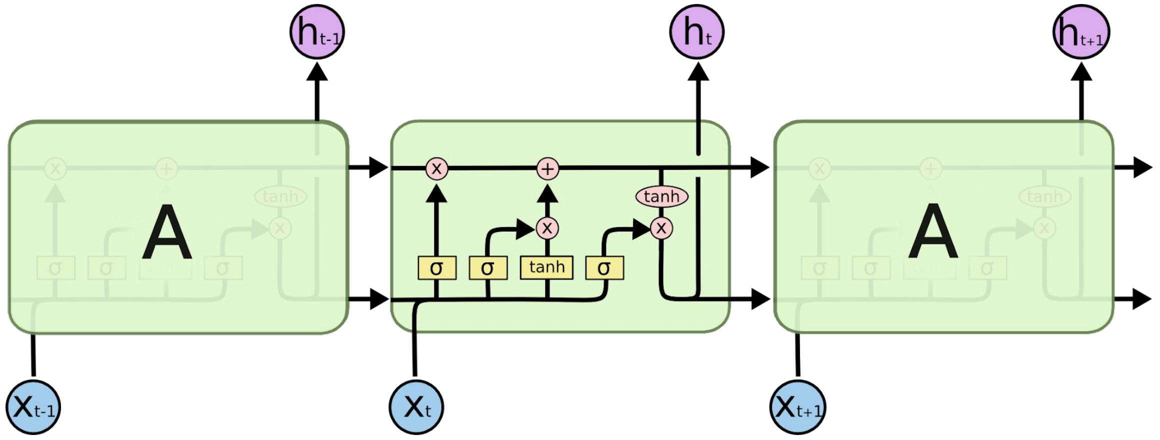
1. ***Vanishing gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$
2. ***Exploding gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$

https://medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22 vanilla RNN



LSTM

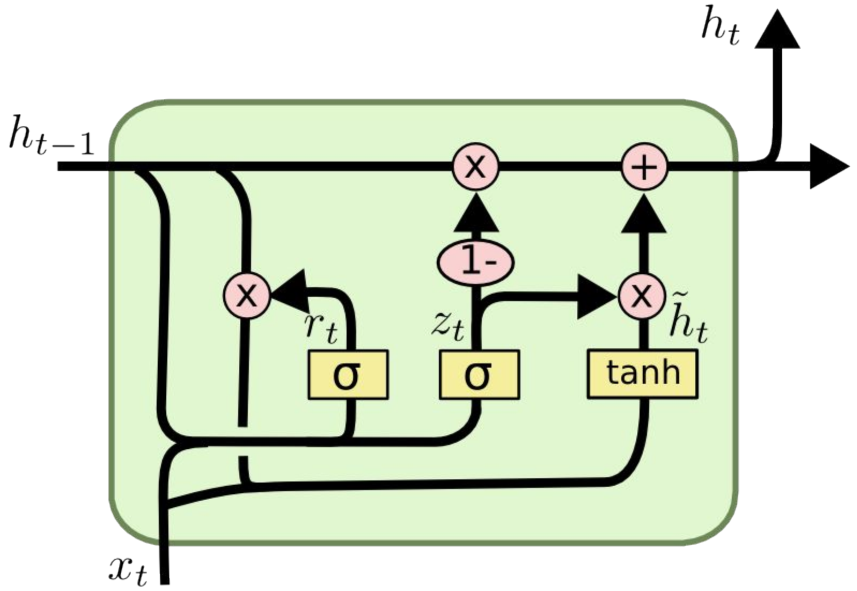
Hochreiter & Schmidhuber (1997)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long short-term memory (LSTM)

Hochreiter & Schmidhuber (1997)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

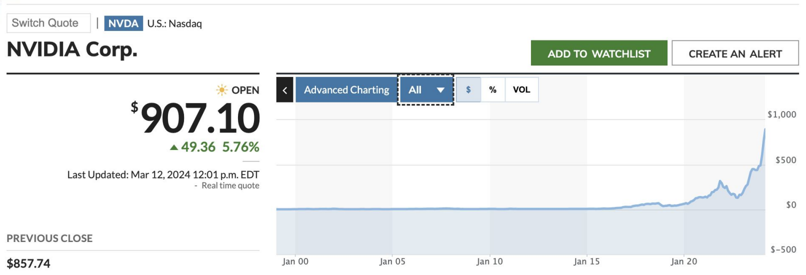
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

New ERA of NN

The new record on the MNIST (National Institute of Standards and Technology) dataset set (60+10K handwritten digits) by Ciresan et al. in 2011 marked a significant milestone in the field of deep learning. They achieved this milestone by employing the use of GPUs (Graphics Processing Units) to accelerate training, demonstrating the power of parallel processing for training deep neural networks efficiently. This breakthrough highlighted the potential of deep learning methods, particularly when combined with hardware optimizations, and paved the way for subsequent advancements in the field.

GPU



NVDA U.S.: Nasdaq

NVIDIA Corp.

OPEN

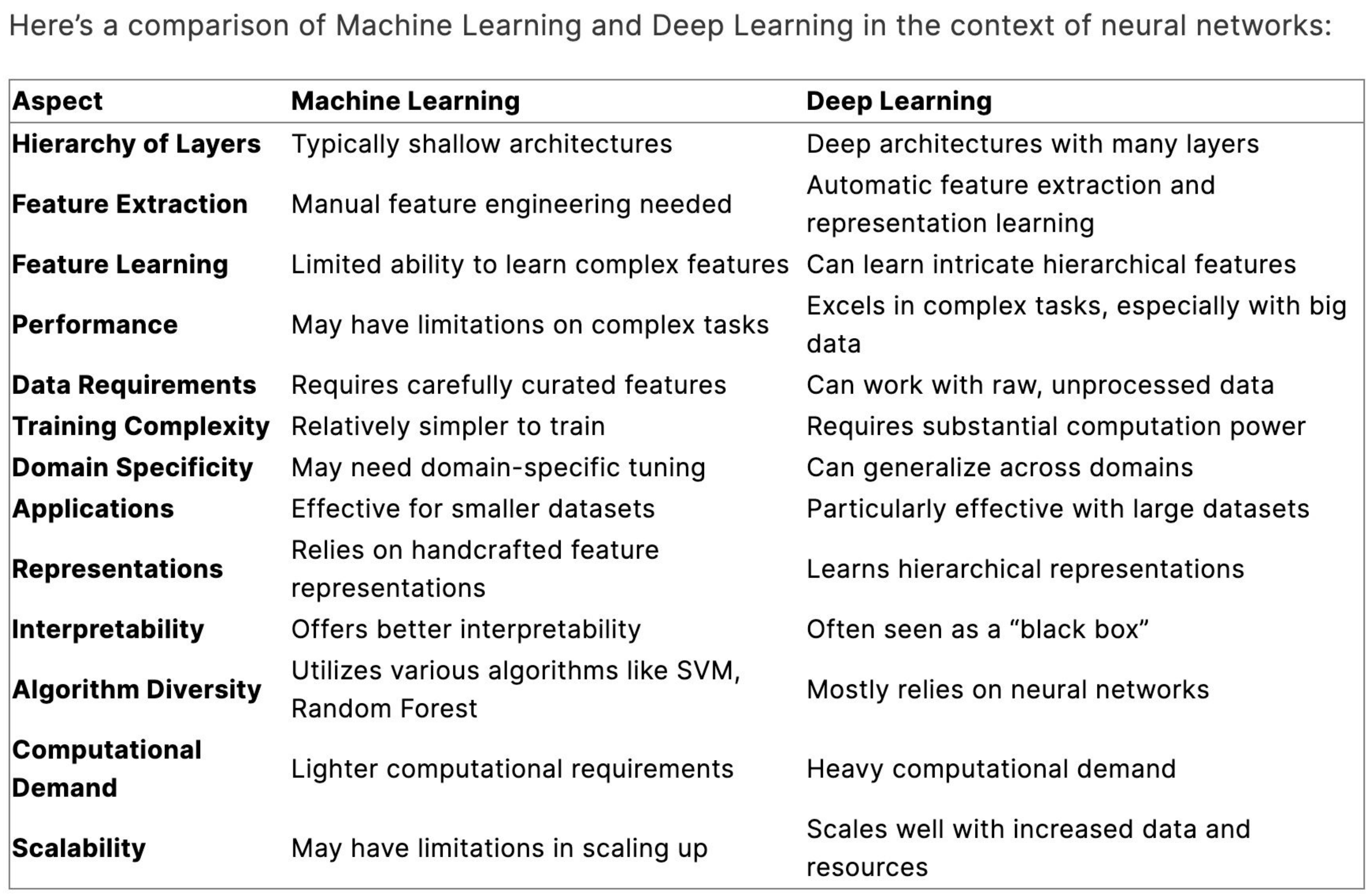
$907.10

▲49.36 5.76%

Last Updated: Mar 12, 2024 12:01 p.m. EDT
- Real time quote

PREVIOUS CLOSE

$857.74



Here’s a comparison of Machine Learning and Deep Learning in the context of neural networks:

Aspect	Machine Learning	Deep Learning
Hierarchy of Layers	Typically shallow architectures	Deep architectures with many layers
Feature Extraction	Manual feature engineering needed	Automatic feature extraction and representation learning
Feature Learning	Limited ability to learn complex features	Can learn intricate hierarchical features
Performance	May have limitations on complex tasks	Excels in complex tasks, especially with big data
Data Requirements	Requires carefully curated features	Can work with raw, unprocessed data
Training Complexity	Relatively simpler to train	Requires substantial computation power
Domain Specificity	May need domain-specific tuning	Can generalize across domains
Applications	Effective for smaller datasets	Particularly effective with large datasets
Representations	Relies on handcrafted feature representations	Learns hierarchical representations
Interpretability	Offers better interpretability	Often seen as a “black box”
Algorithm Diversity	Utilizes various algorithms like SVM, Random Forest	Mostly relies on neural networks
Computational Demand	Lighter computational requirements	Heavy computational demand
Scalability	May have limitations in scaling up	Scales well with increased data and resources

https://www.analyticsvidhya.com/blog/2022/01/introduction-to-neural-networks/