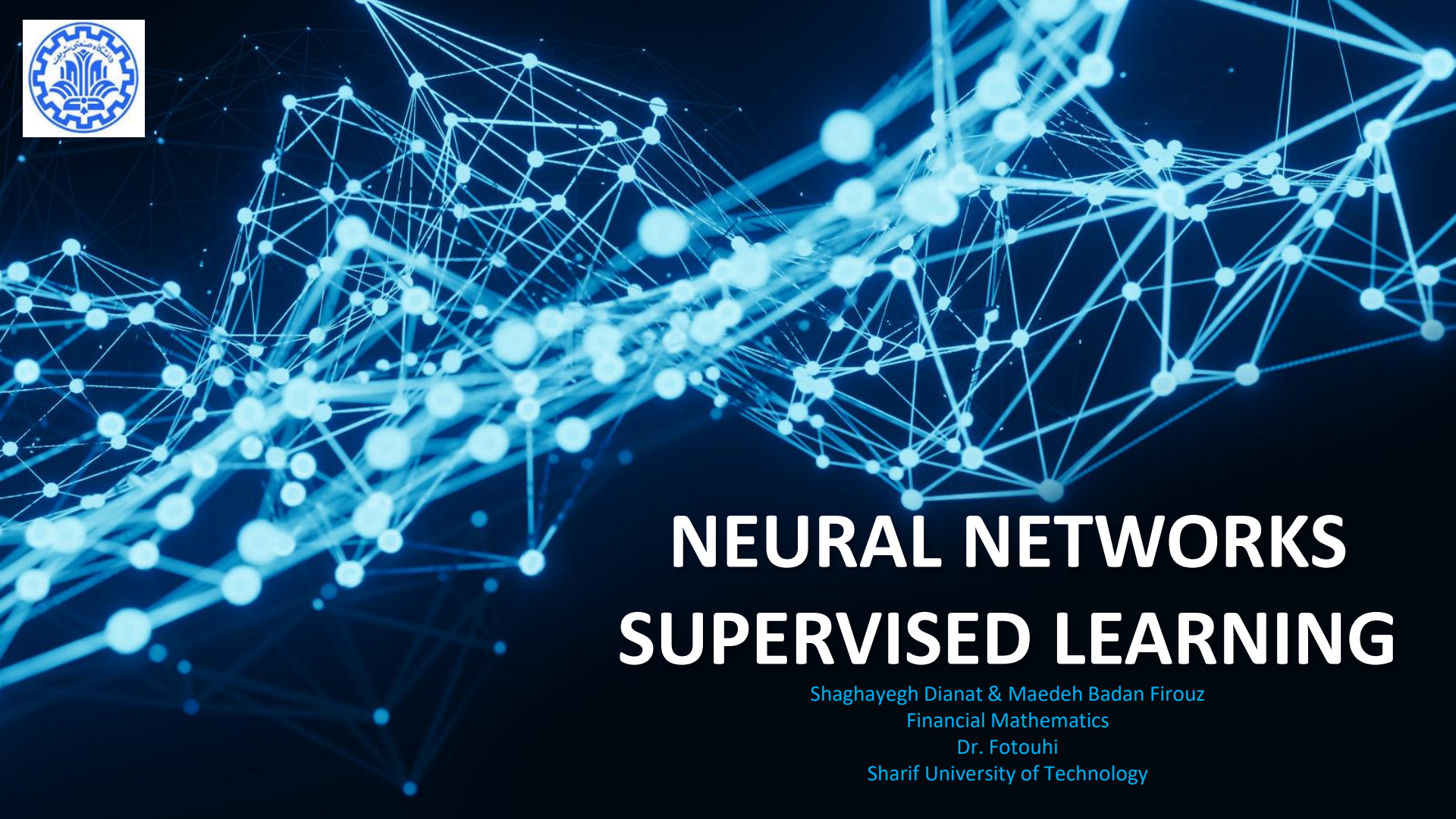


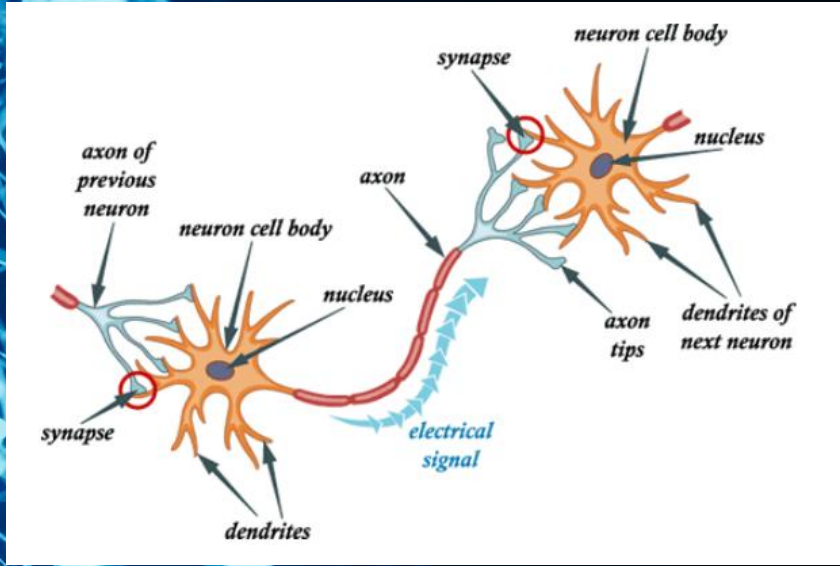
NEURAL NETWORKS SUPERVISED LEARNING

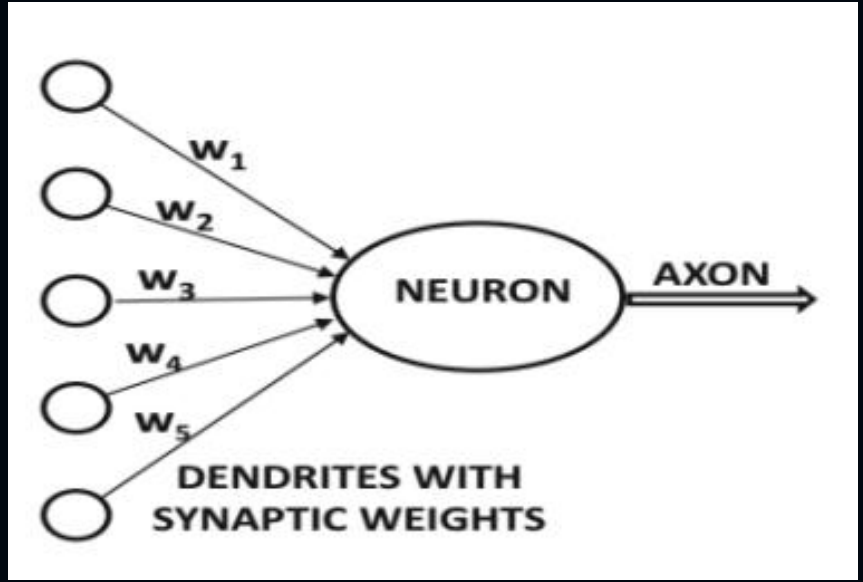
Shaghayegh Dianat & Maedeh Badan Firouz
Financial Mathematics
Dr. Fotouhi
Sharif University of Technology

What is a neural network?

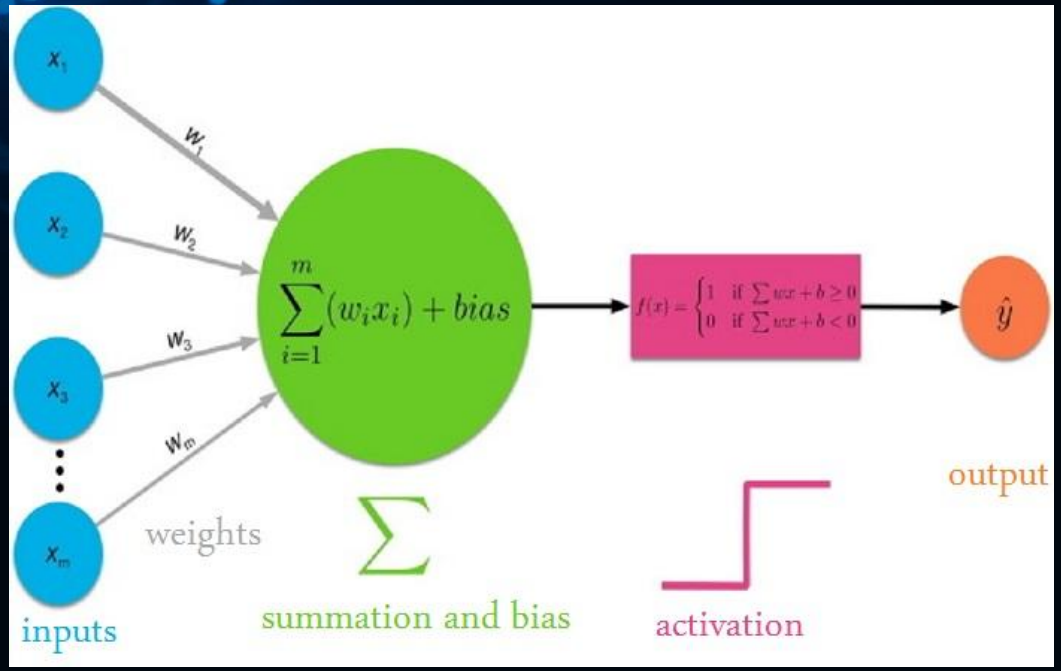
- It is the result of human efforts to simulate biological neurons!

Biological Neuron VS Artificial Neuron





Perceptron



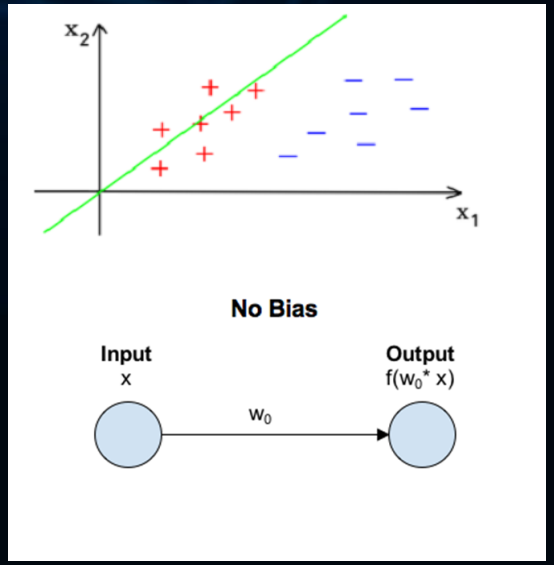
Input layer

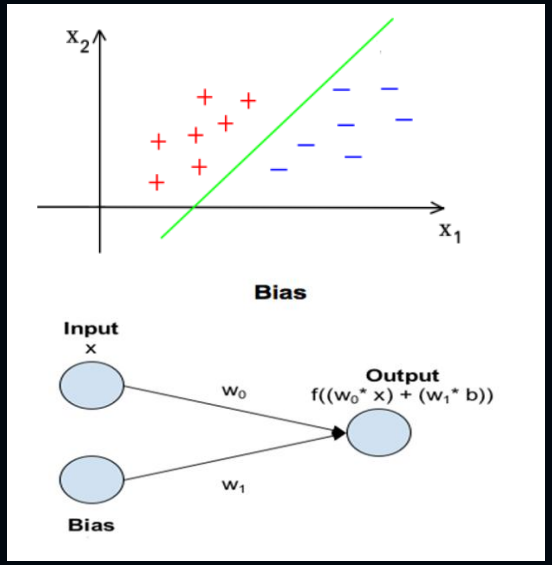
Hidden layer

Output layer

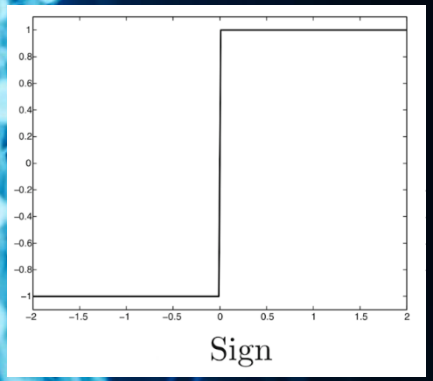
Bias

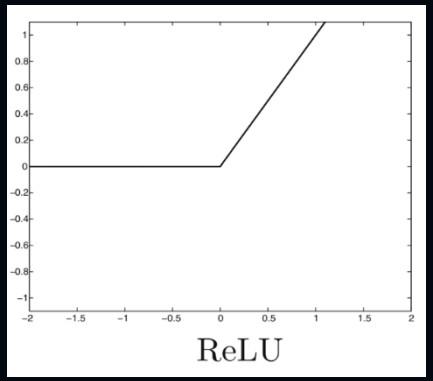
A bias value allows you to shift the activation function to the left or right, which may be critical for successful learning.

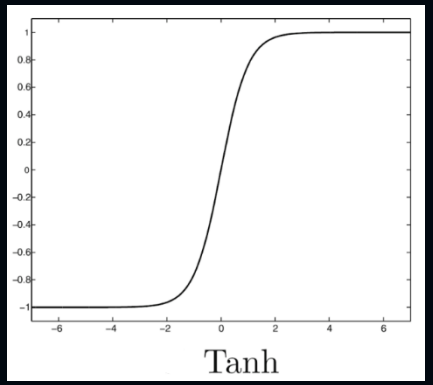


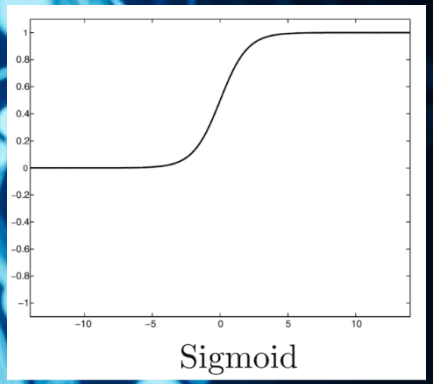


Some Elementwise Activation Functions









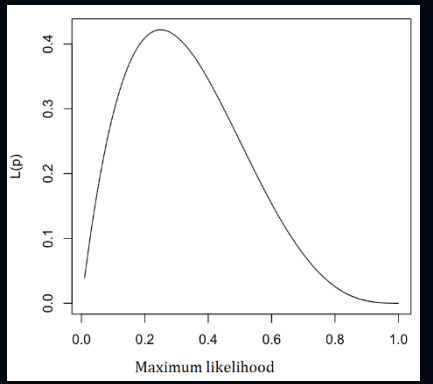
Sign function: $f(x) = sign(x)$

Sigmoid function: $f(x) = \frac{1}{1 - e^{-x}}$

ReLU function: $f(x) = \max\{x, 0\}$

Tangh function: $f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$

Maximum likelihood: $f(x) = \sum ln(x) - \sum ln(1 - x)$



Loss Function

- Loss function: A loss function measures how good a neural network model is in performing a certain task.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

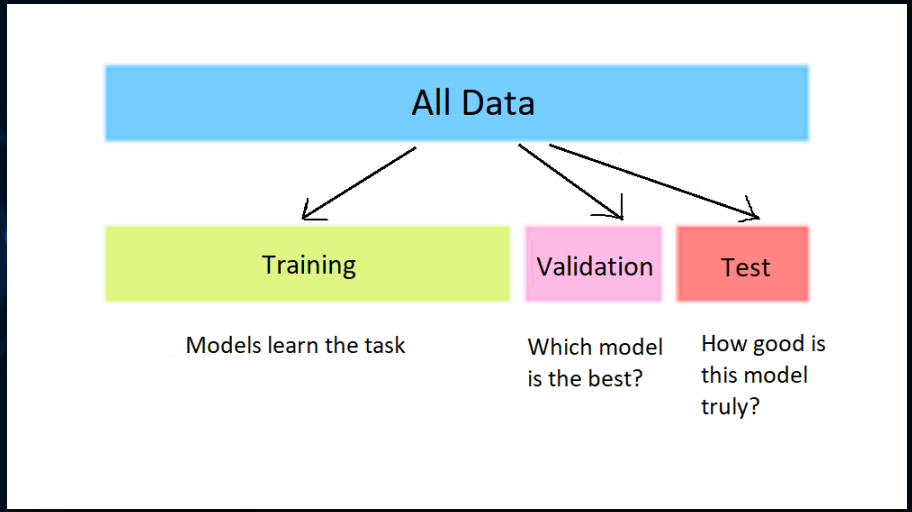
MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

MAE = mean absolute error
y_i = prediction
x_i = true value
n = total number of data points

Dataset

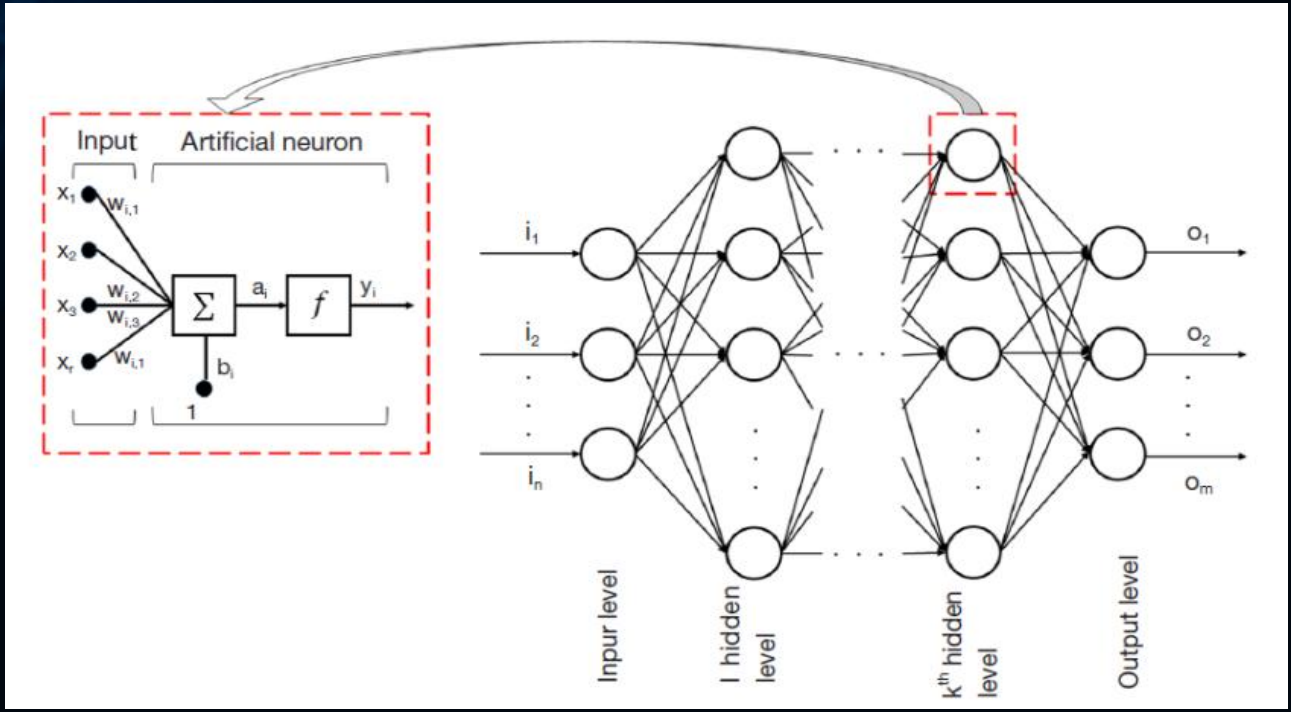
- We usually divide the dataset into 3 sets before start training network:



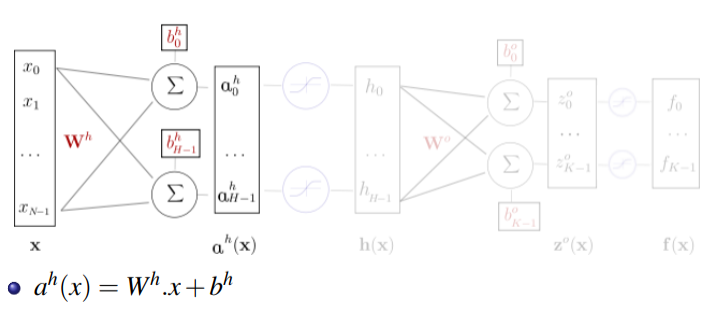
Multilayer Perceptron

Maximum number of the parameters in a model with:
F features,
H hidden layers,
M neuron in each layer and
T targets:

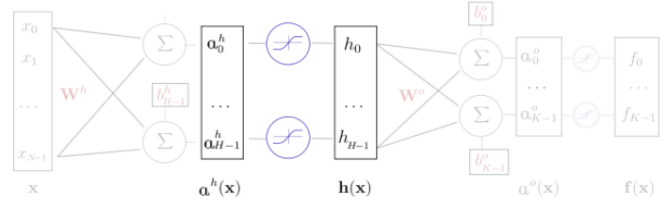
$$(F+1)M + M(M+1)(H-1) + (M+1)T$$



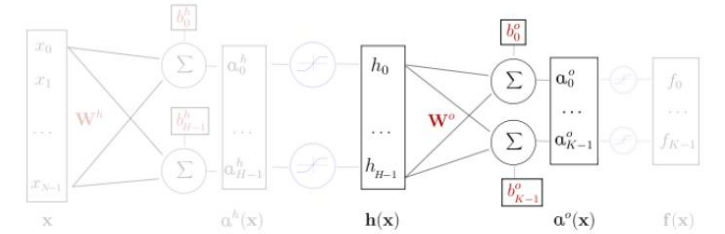
Forward Propagation



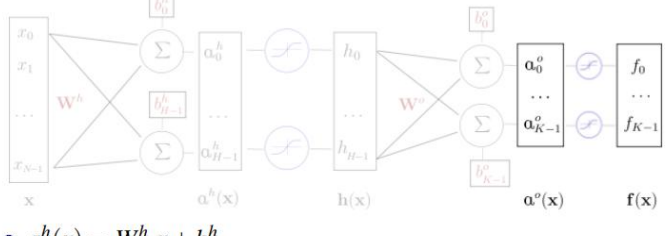
- $a^h(x) = W^h.x + b^h$



- $a^h(x) = W^h.x + b^h$
- $h(x) = \Phi(a^h(x)) = \Phi(W^h.x + b^h)$



- $a^h(x) = W^h.x + b^h$
- $h(x) = \Phi(a^h(x)) = \Phi(W^h.x + b^h)$
- $a^o(x) = W^o.h(x) + b^o$

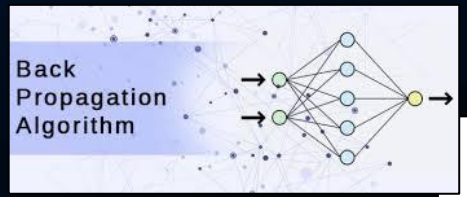


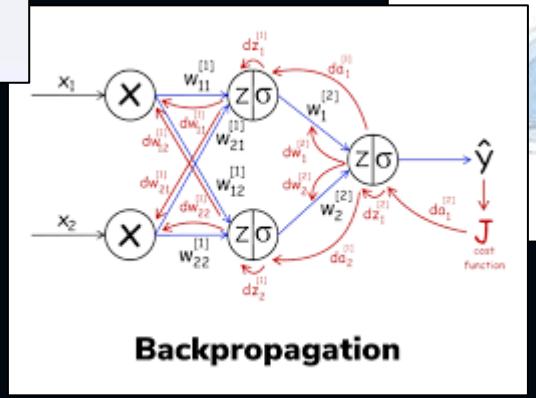
- $a^h(x) = W^h.x + b^h$
- $h(x) = \Phi(a^h(x)) = \Phi(W^h.x + b^h)$
- $a^o(x) = W^o.h(x) + b^o$
- $f(x) = \Phi(a^o(x)) = \Phi(W^o.h(x) + b^o)$

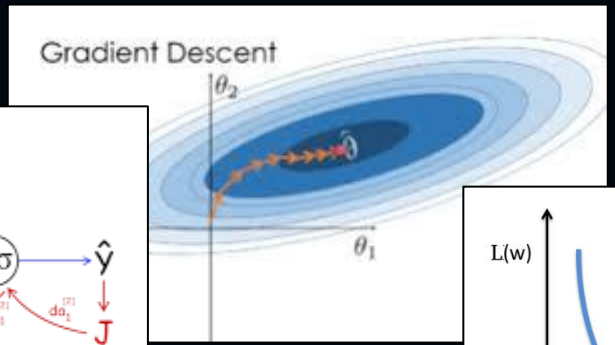
- Epoch: An epoch means training the neural network with all the training data for one cycle. In an epoch, we use all of the data exactly once.

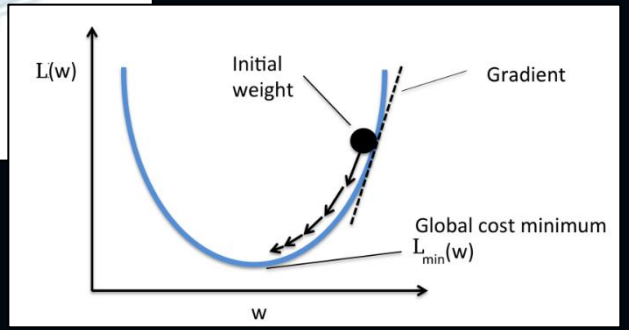
How is a neural network trained?

We will describe some basics to answer this question.

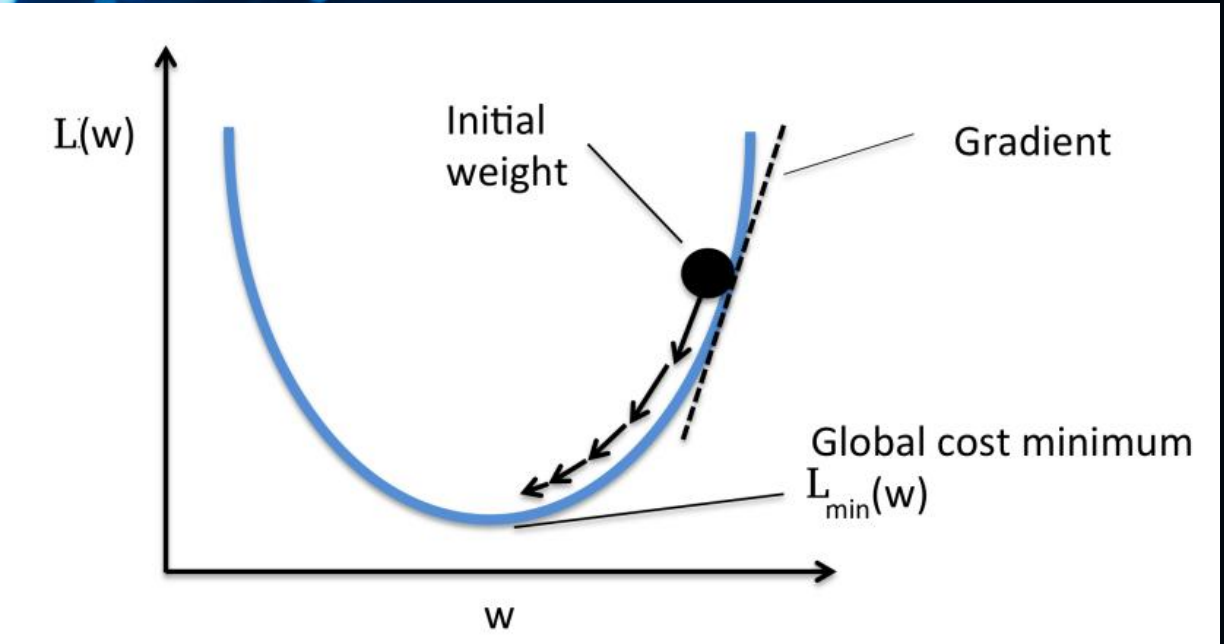








Gradient Descent Algorithm

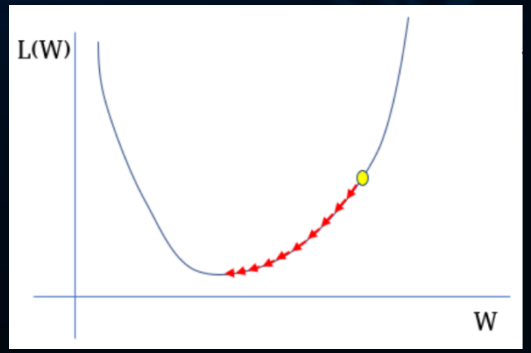


$$w_{i+1} = w_i - \underbrace{\alpha}\nabla L(w_i)$$

Learning rate

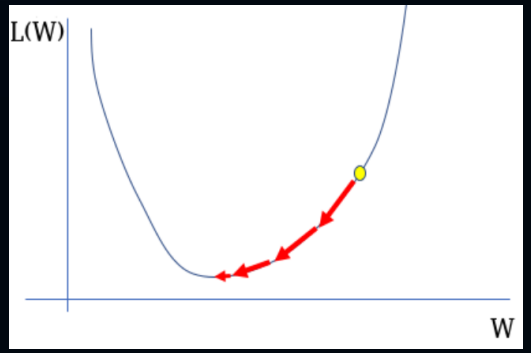
Matter of Learning Rate

Too low



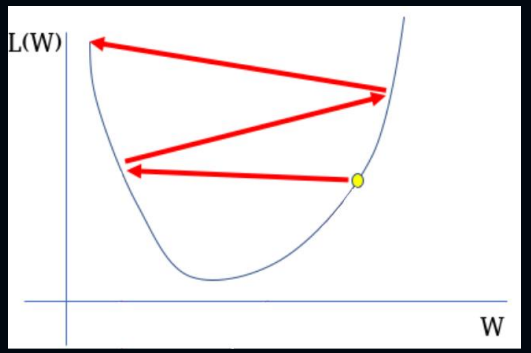
A small learning rate requires many updates before reaching the minimum point

Just right



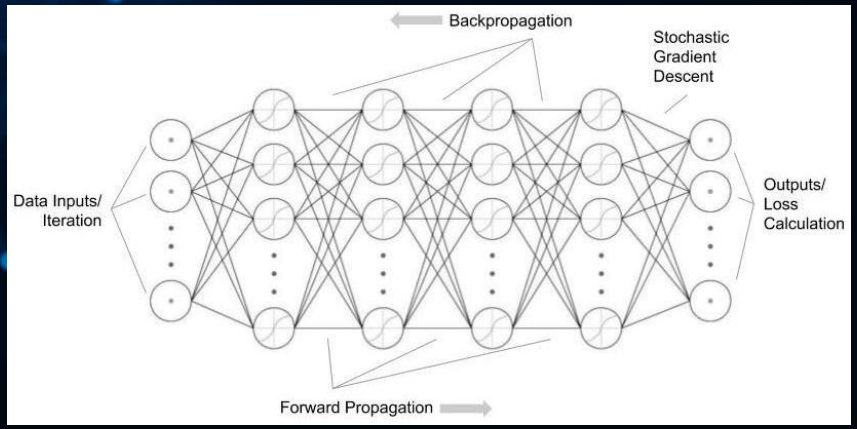
Suitable learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

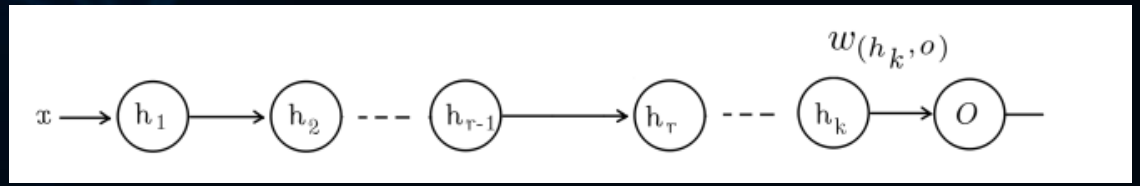
Backpropagation



- Backpropagation is short for "backward propagation of errors,“. This is an algorithm for supervised learning of artificial neural networks using gradient descent.
- Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights.

Backpropagation for MSE

- To update any weight of the output layer, we use the gradient of the loss function with respect to that weight.

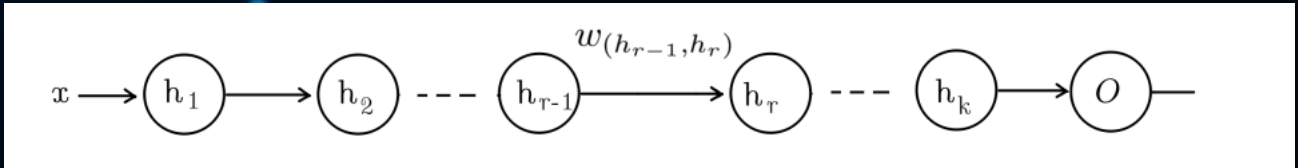


$$\frac{\partial L}{\partial w_{(h_k,o)}} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial w_{(h_k,o)}} = \Delta(o,o)h_k\Phi'(a_o)$$

$$\Delta(o,o) := \frac{\partial L}{\partial o} = -(Y-O) \qquad \frac{\partial o}{\partial w_{(h_k,o)}} = \frac{\partial o}{\partial a_o}\frac{\partial a_o}{\partial w_{(h_k,o)}} = h_k\Phi'(a_o)$$

Updating middle weights

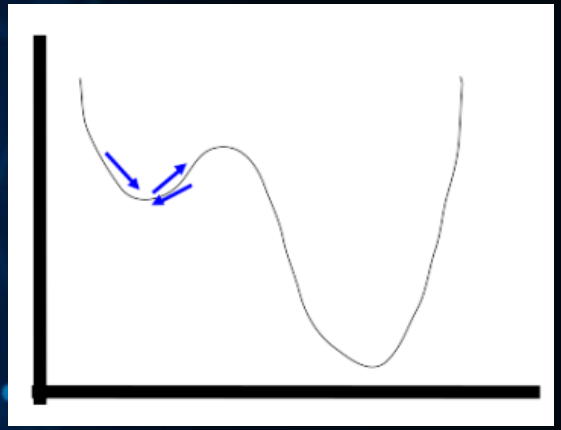
- To update a connection weight, we should compute the gradient of the loss function with respect to that weight.

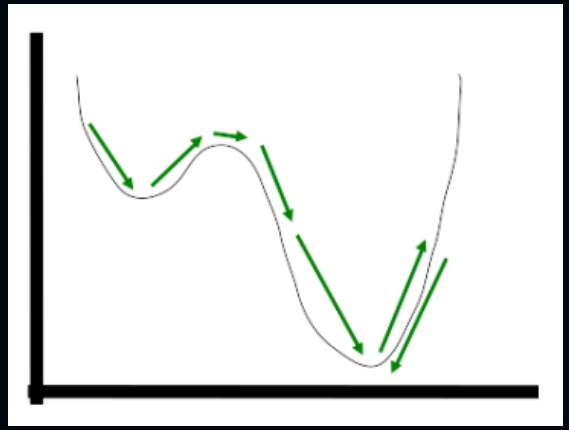


$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}$$

The Bug of Gradient Descent

- The gradient descent algorithm can lead to a local minimum
- To speed up the learning process and attempt to avoid local minima several variations on the basic gradient descent algorithm have been developed



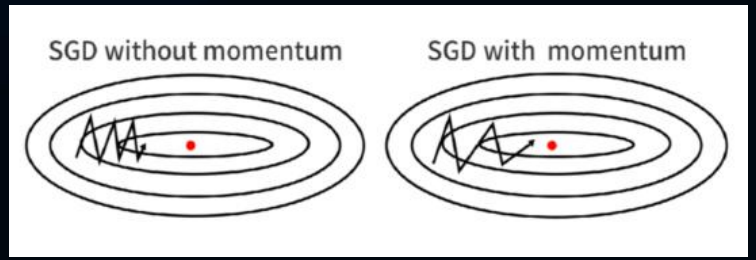


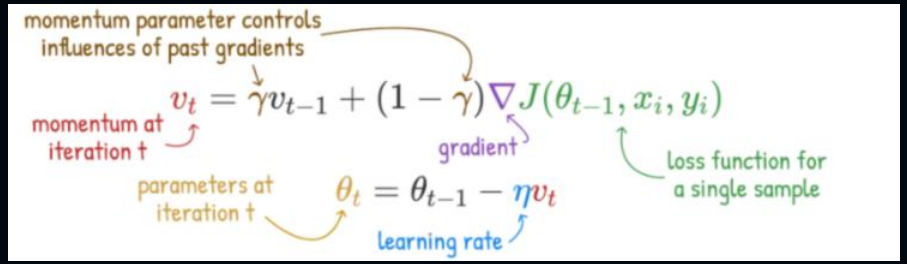
Variations on Gradient Descent

- Momentum
- Mini-batch SGD
- Adaptive learning rate
  - AdaGrad
  - RMSProp
  - ADAM
- Learning rate decay
- dropouts

Momentum

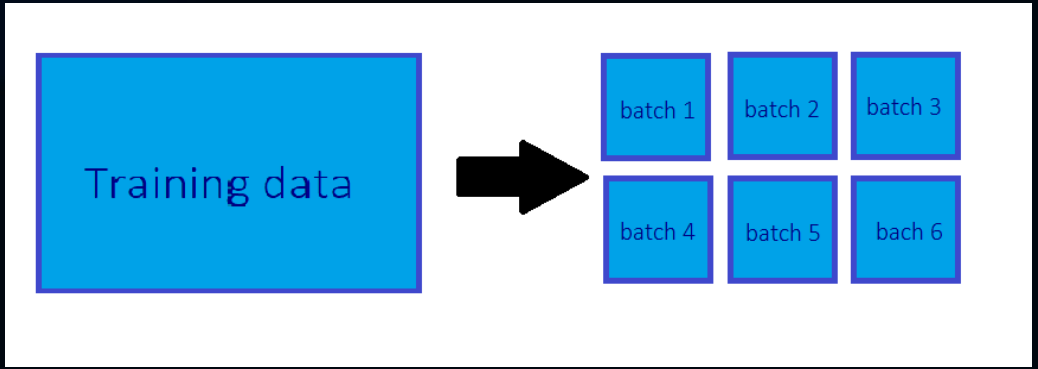
- Stochastic gradient descent with momentum uses an exponentially weighted average of past gradients to update the momentum term and the model's parameters at each iteration.
- It helps the optimizer maintain a more stable direction and speed up convergence.





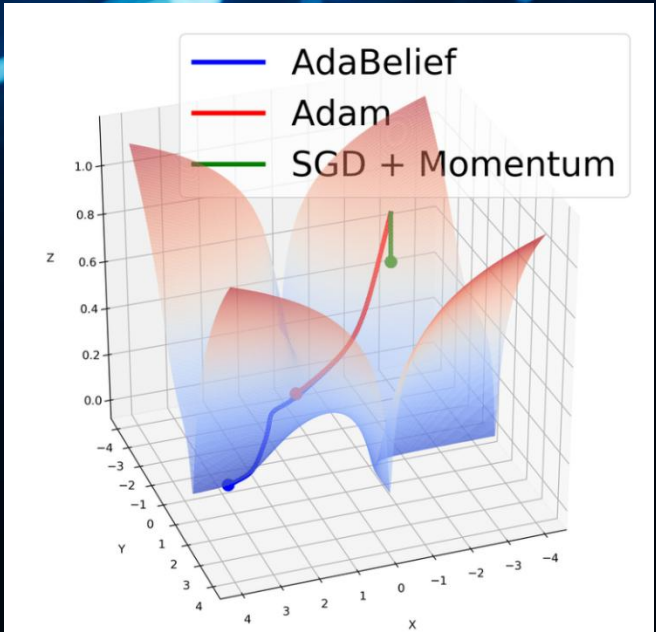
Mini-batch stochastic gradient descent

- This randomly splits the training data set into small subsets known as mini-batches.
- An epoch is a set of iterations that make one use of the whole training set. The number of epochs is the number of complete passes through the training dataset.



$$w_{i+1} = w_i - \alpha \nabla J(x^{i:i+b}, y^{i:i+b}, w_i)$$

Adaptive learning rate

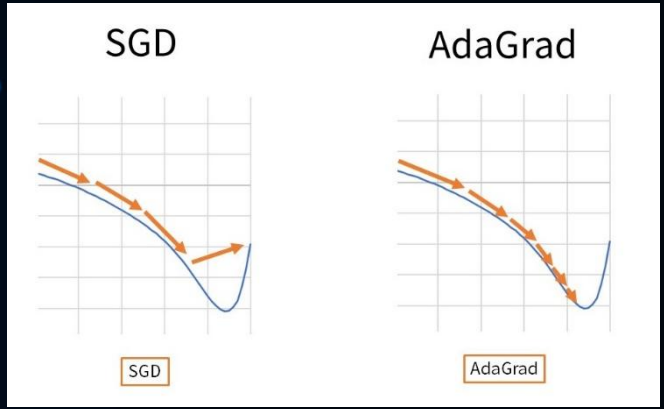


- Adam is different to classical stochastic gradient descent.
- ADAM method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.
- Adam combines the advantages of the other extensions of stochastic gradient descent: Momentum, AdaGrad and RMSProp.

AdaGrad

- Adaptive Gradient Algorithm (Adagrad) is an algorithm for gradient-based optimization. The learning rate is adapted component-wise to the parameters by incorporating knowledge of past observations.

$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{\varepsilon + \sum_{k=1}^{t} (\nabla J(w_{k,i}))^2}} \nabla J(w_{t,i})$$

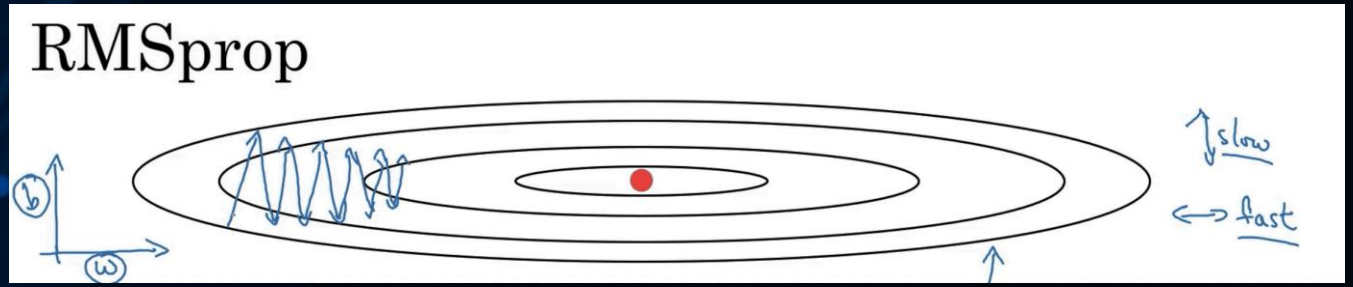


RMSProp

- Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight.
- It is very close to AdaGrad, except that it does not provide the sum of the gradients, but an exponentially decaying average.

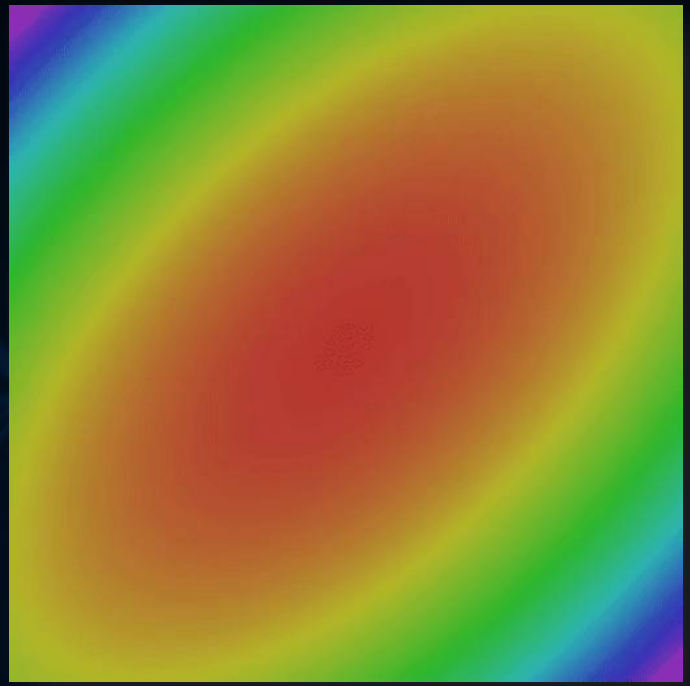
$$w_{t+1,i} = w_{t,i} - \frac{\alpha}{\sqrt{\varepsilon + E[(\nabla J(w_{t,i}))^2]_t}} \nabla J(w_{t,i})$$

$$E[(\nabla J(w_{t,i}))^2]_t = (1-\gamma)(\nabla J(w_{t,i}))^2 + \gamma E[(\nabla J(w_{t,i}))^2]_{t-1}$$



RMSProp VS SGC with Momentum

- Although SGD with momentum is able to find the global minimum faster, this algorithm takes a much longer path that could be dangerous.



ADAM

$$m_0 = 0, v_0 = 0$$

$$m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta)$$ Momentum

$$v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2)\nabla_\theta \mathcal{L}(\theta)^2$$ RMS Prop

$$\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1}} + 1e^{-5}} m_{t+1}$$ RMS Prop + Momentum

- ADAM is the result of combining SGD with Momentum and RMSProp.

Improvement

$$m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta)$$ Momentum

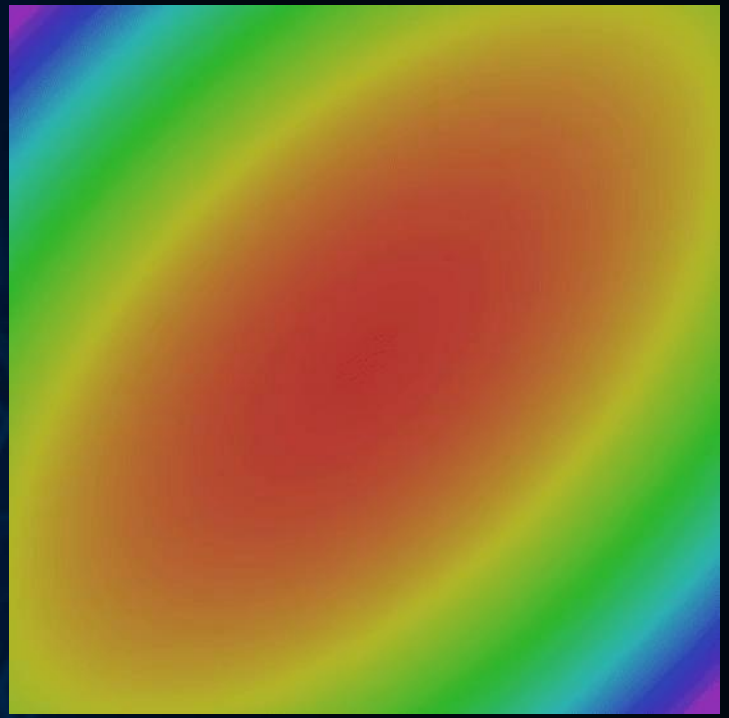
$$v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2)\nabla_\theta \mathcal{L}(\theta)^2$$ RMS Prop

$$\hat{m}_{t+1} \leftarrow \frac{m_{t+1}}{1 - \beta_1^t}$$

$$\hat{v}_{t+1} \leftarrow \frac{v_{t+1}}{1 - \beta_2^t}$$ Bias Correction

$$\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1}} + 1e^{-5}} m_{t+1}$$ RMS Prop + Momentum

An overview



SGD with Momentum

$$v_{t+1} \leftarrow \rho v_t + \nabla_\theta \mathcal{L}(\theta)$$

$$\theta_j \leftarrow \theta_j - \epsilon v_{t+1}$$

AdaGrad

$$g_0 = 0$$

$$g_{t+1} \leftarrow g_t + \nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

RMS Prop

$$g_0 = 0, \alpha \simeq 0.9$$

$$g_{t+1} \leftarrow \alpha \cdot g_t + (1-\alpha)\nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

Learning rate decay

- Learning rate decay is a technique used in machine learning models, especially deep neural networks. It is sometimes referred to as learning rate scheduling or learning rate annealing.
- There are various ways to reduce the learning rate: some reduce it gradually over time, while others drop it sharply after a set number of training rounds. The key is to find a balance that lets the model learn efficiently without missing the best possible solution.

A typical decay

$$\alpha_t = \frac{\alpha_0}{1 + (drop\ rate)t}$$

Step decay

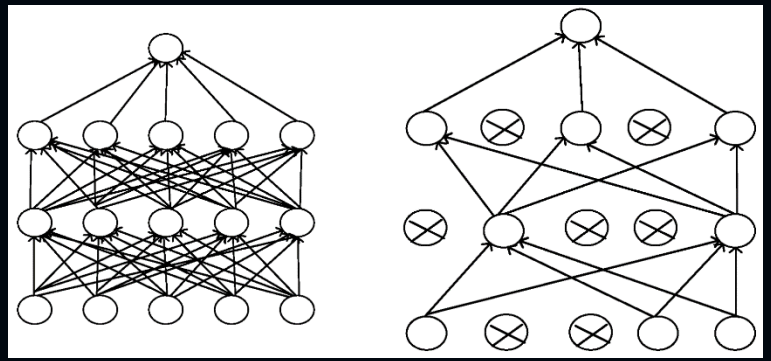
$$\alpha_{new} = \alpha_{old} * (drop\ rate)^{\frac{epoch}{step\ size}}$$

Exponential decay

$$\alpha_{new} = \alpha_{old} * e^{-decay\ rate * epoch}$$

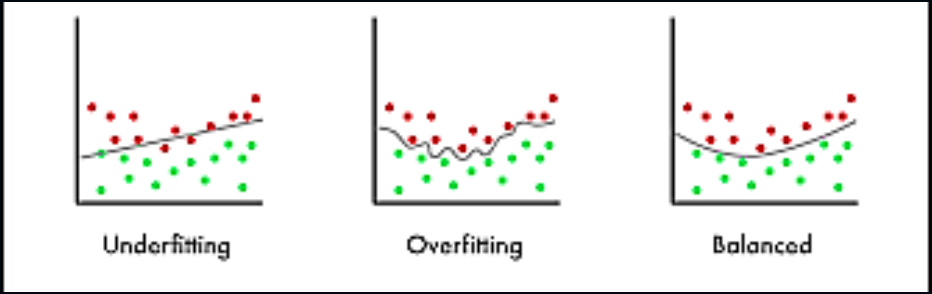
dropouts

- The dropout method runs gradient descent using different networks sampled by dropping out nodes, and the weights learned are averaged in the end.
- Considering each sampled network as a model to be learned, the method combines the learned parameters of different models when obtaining the final result.



Overfitting

- Overfitting is an undesirable machine learning behavior that occurs when the machine learning model gives accurate predictions for training data but not for new data.

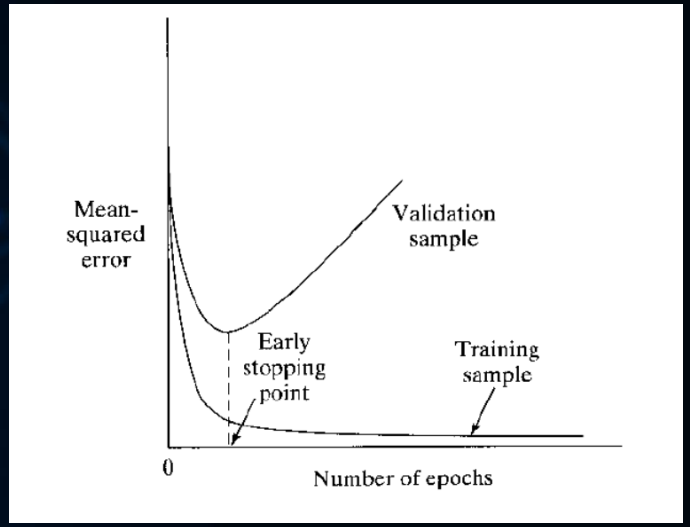


Avoid Overfitting

- One possible approach is to reduce the size of the network. However, large networks have the potential to be more powerful than small networks.
- Provide more training samples (not always possible).
- Early stopping: Stop learning before overfitting happens.
- Use regularization terms to dynamically adjust network complexity.
- Use the random dropout technique for hidden neurons.

Early Stopping Method

- Split training samples into a training set (80%) and a validation set (20%).
- Stop learning when the loss decreases on the train set but increases on the validation set.

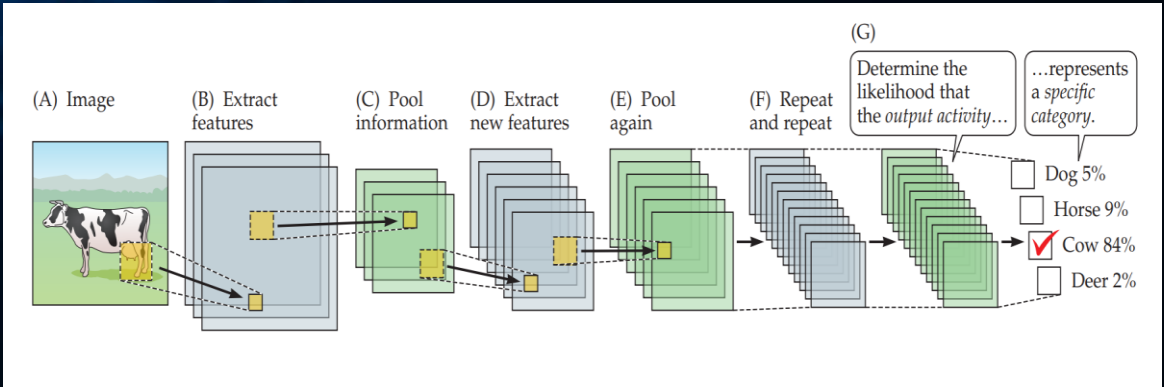


Regularization

- Regularization reduces overfitting by penalizing larger weights, encouraging the model to prioritize simpler hypotheses.
- It works by introducing penalties term or constraints on the model's parameters during training. By doing so, regularization prevents the model from fitting the training data too closely, which is a common cause of overfitting.

$$Loss = MSE + \alpha * \Sigma|w|$$

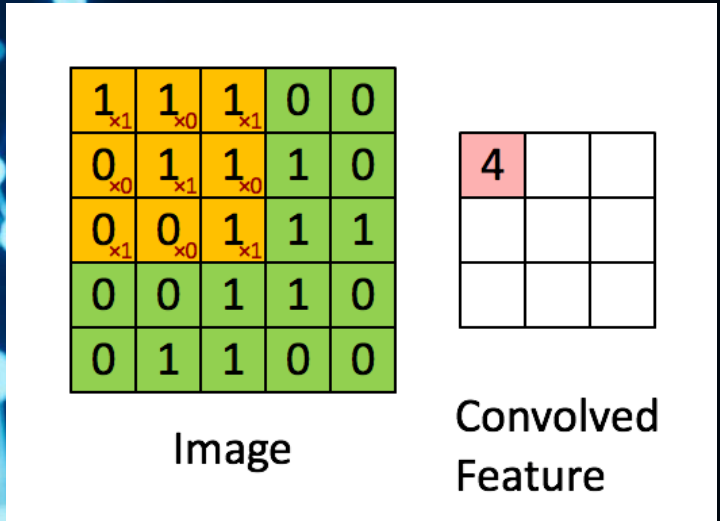
Convolutional Neural Networks



- A CNN connects the neurons in one layer to only a subset of the neurons in the previous layer.
- The role of CNN is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.
- This is important when we want to design an architecture that is not only good at learning features but also scalable to massive datasets.

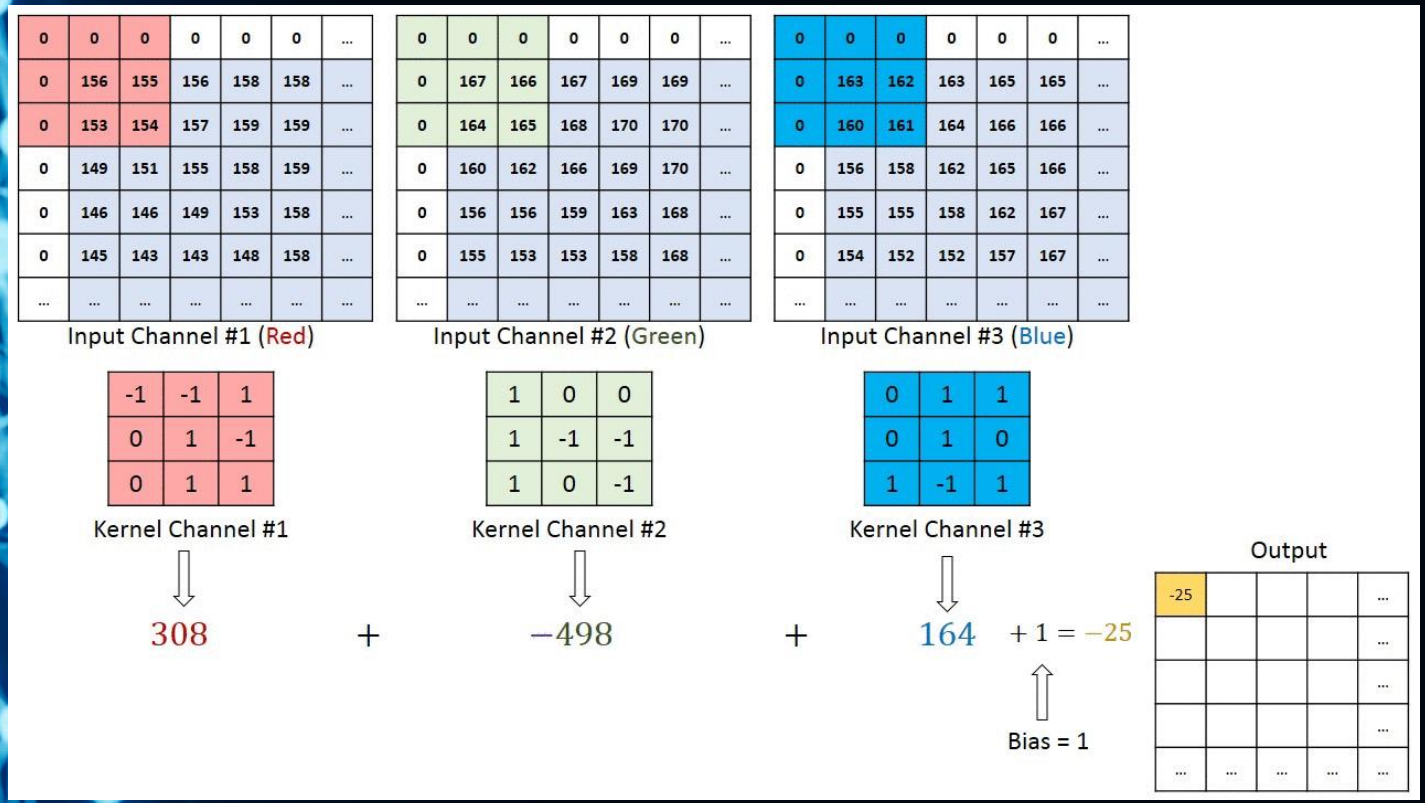
Convolution

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

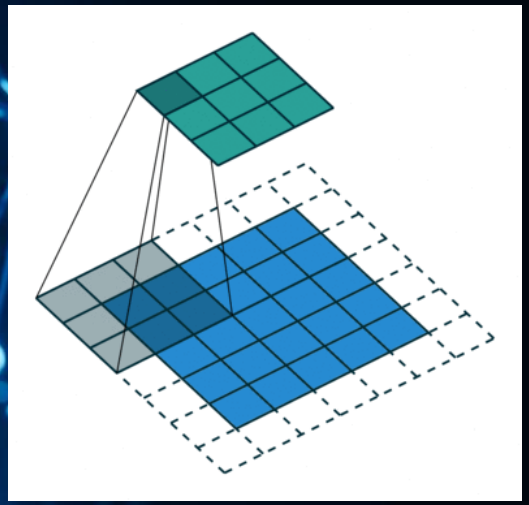


- Yellow Matrix: Filter/ Kernel
- Each of submatrices: receptive field
- Red Matrix: Feature map

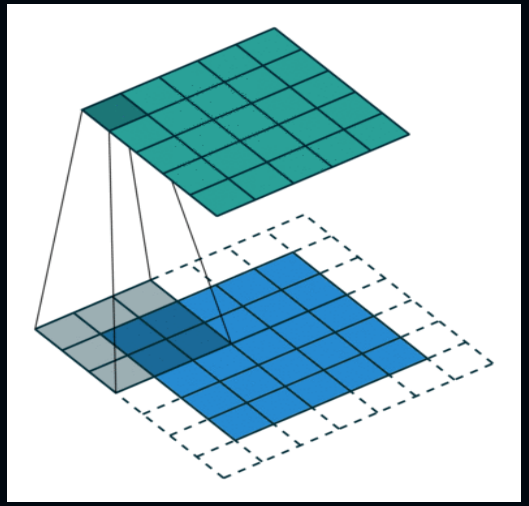
- Padding: artificially fill borders of image



- Stride determines how many squares or pixels our filters skip when they move across the image, from left to right and from top to bottom



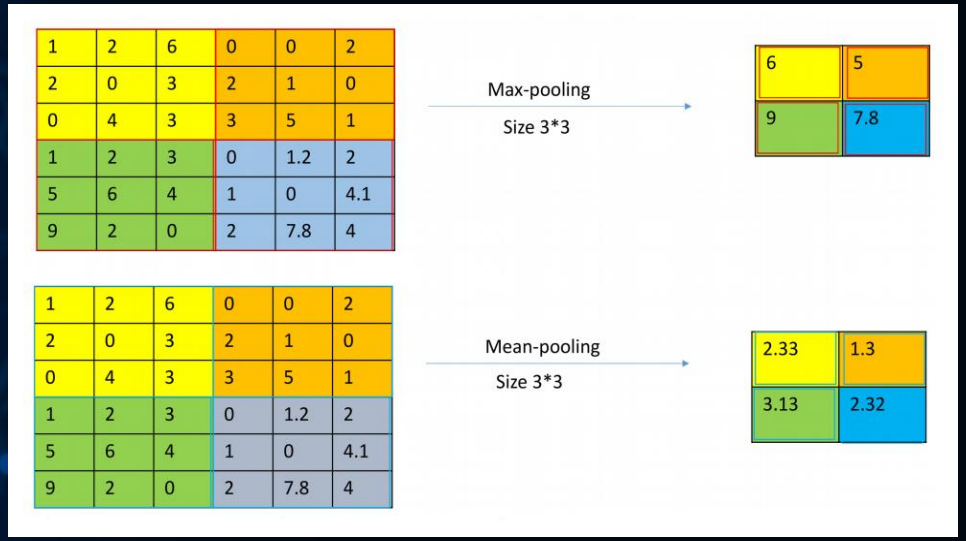
Stride = 2



Stride = 1

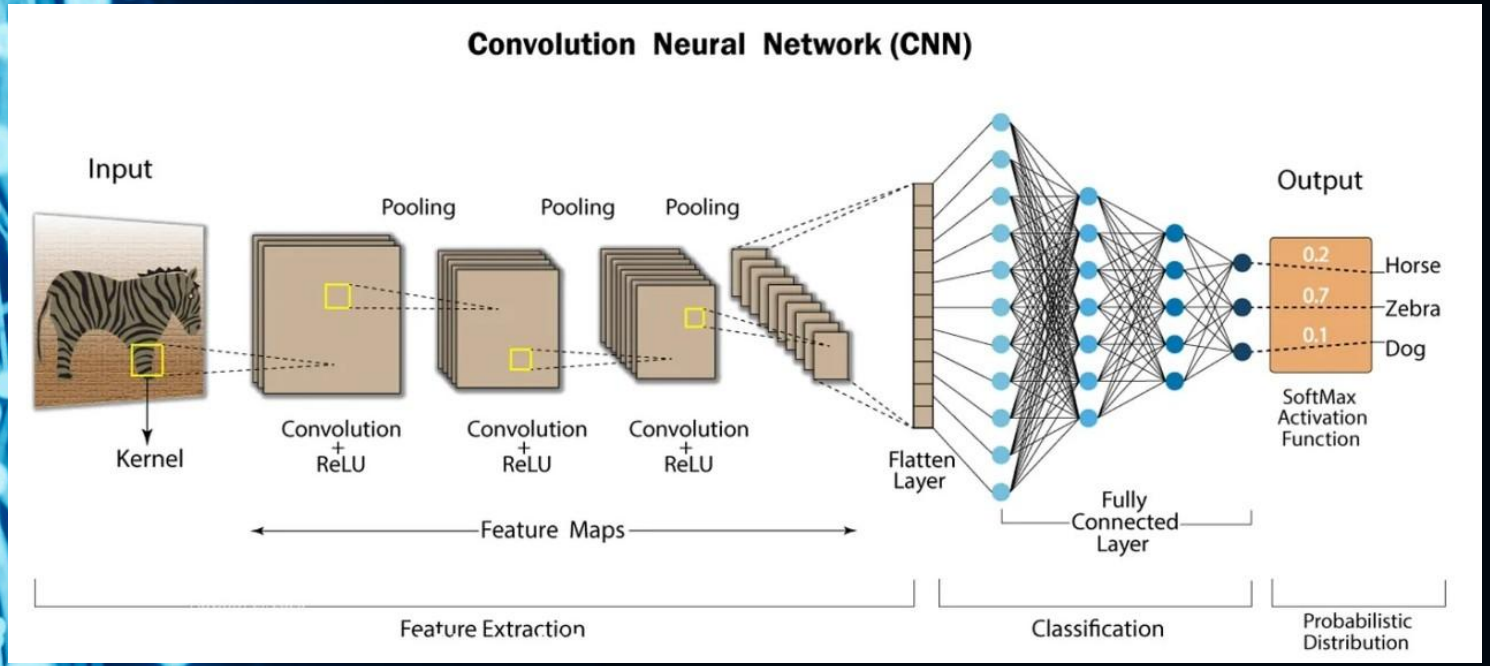
Pooling

- The pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- This decreases the computational power required to process the data through dimensionality reduction.

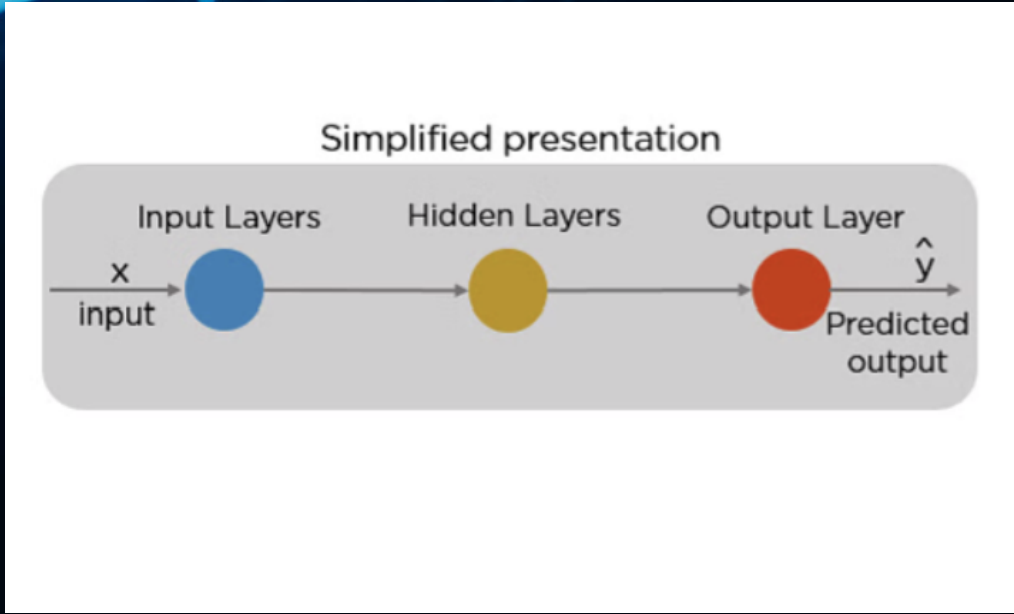


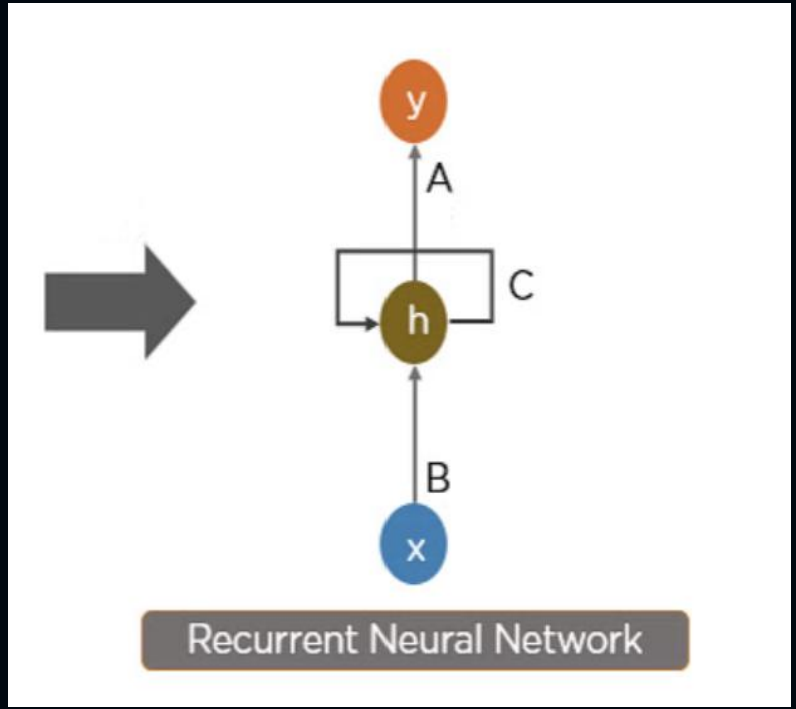
Fully Connected Layer

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



Feedforward NN vs Recurrent NN





Recurrent Neural Networks

RNN is a deep learning model trained to process and convert a sequential data input into a specific sequential data output.

An RNN preserves the temporal sequence in which observations occur because it wants to allow for changes in its prediction model through time.

In an RNN, the activation function at time t is applied to a total of:

→ a linear combination of the values at the neurons of layer l−1 at time t

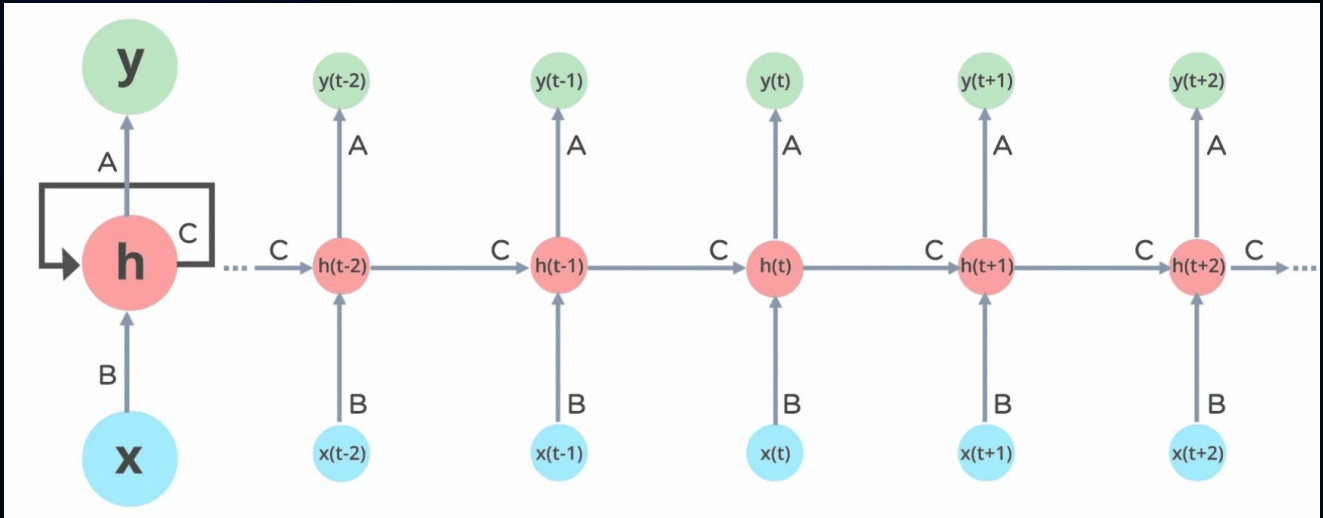
↻ a linear combination of the values at the neurons of layer l for the observation at the previous time t−1.

Recurrent Neural Network after unfolding

$$h_t = f\left(h_{t-1}, x_t\right)$$

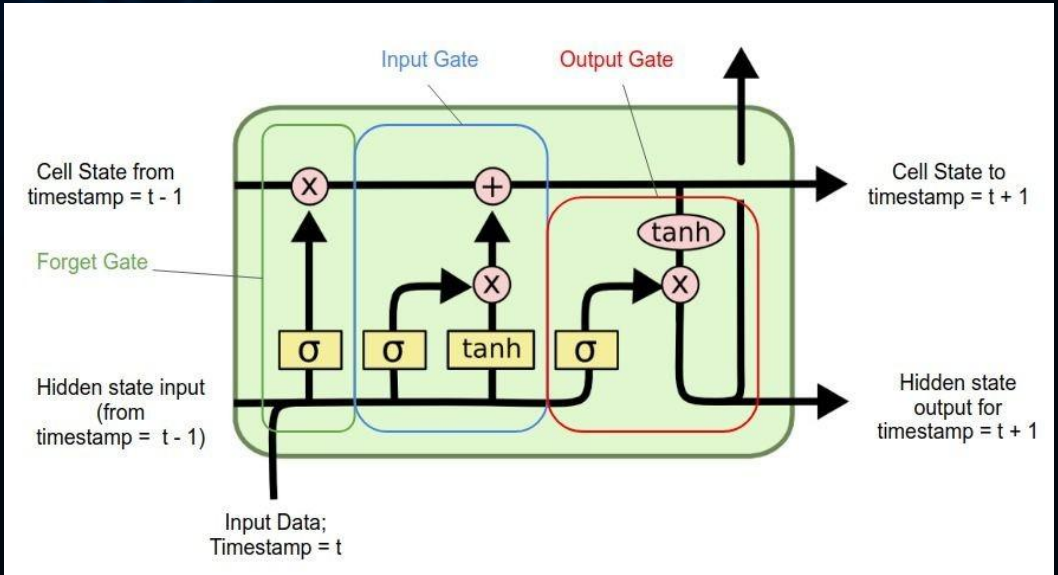
$$h_t = \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t\right)$$

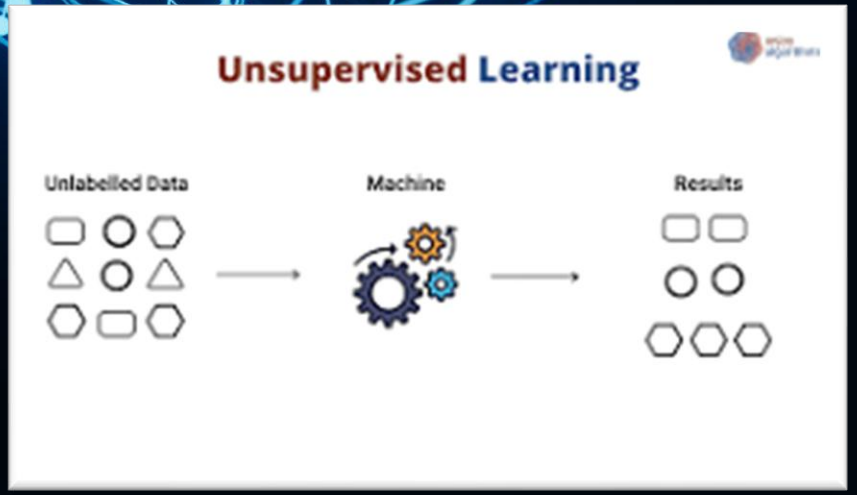
$$y_t = W_{hy} h_t$$



Long Short-Term Memory

- Long short-term memory (LSTM) is an improved version of recurrent neural network (RNN) aimed at dealing with the vanishing gradient problem present in traditional RNNs.





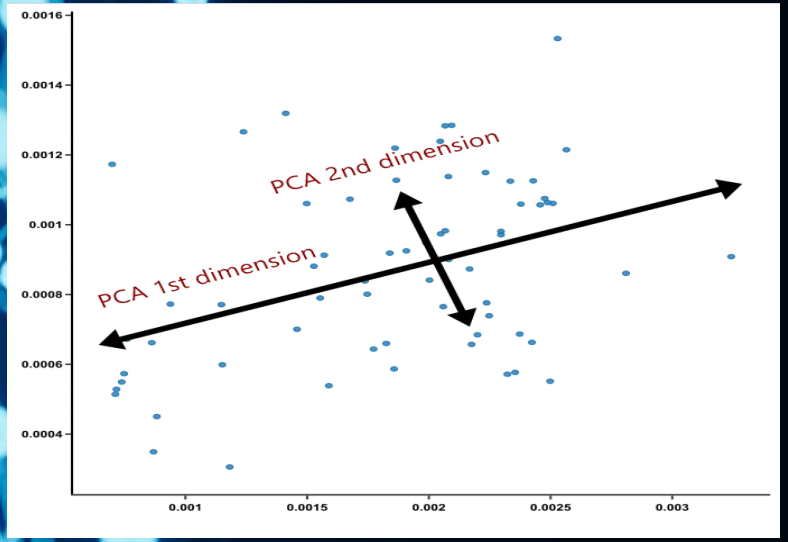
Unsupervised Learning

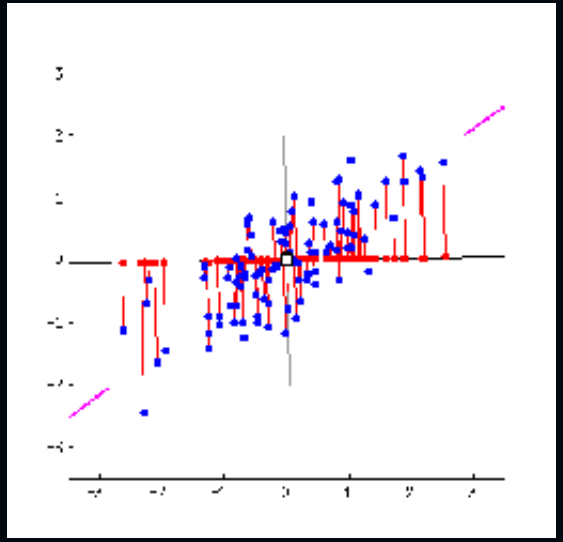
- Unsupervised learning in artificial intelligence is a type of machine learning that learns from data without human supervision. It Works with datasets without considering the target variable.

- Unlike supervised learning, unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.

Principal Component Analysis

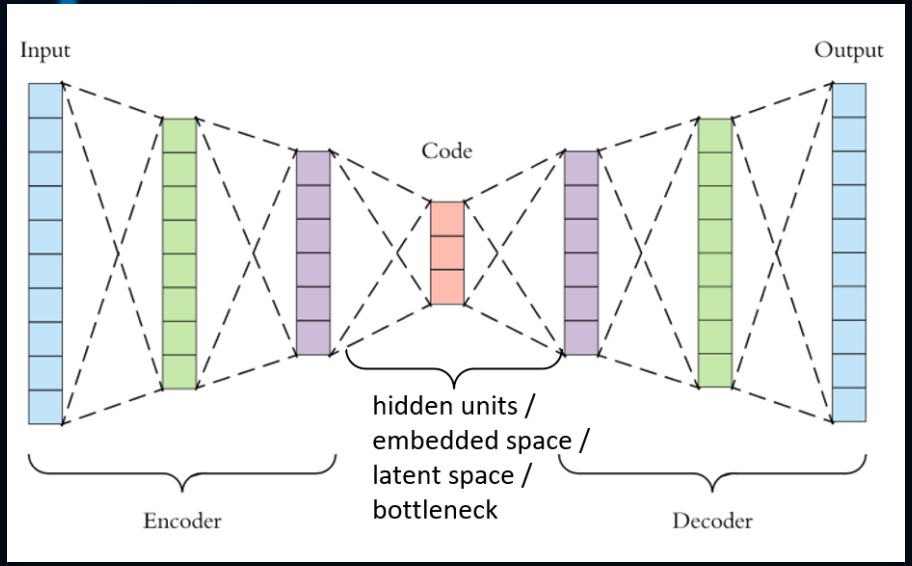
- PCA is a way to bring out strong patterns from large and complex datasets.
- Principal components capture the most variation in a dataset. (PC1 and PC2)



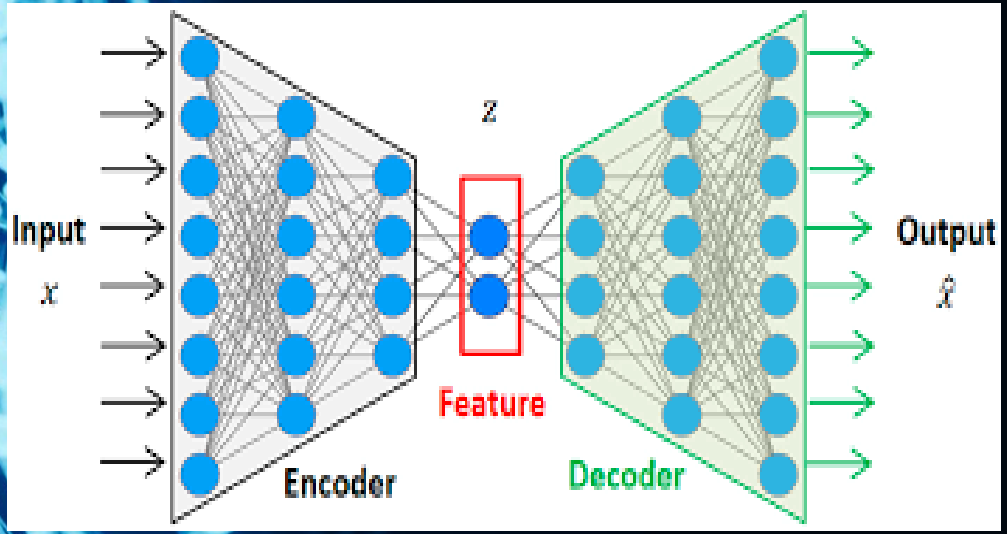


Autoencoder

- An autoencoder is a neural network designed to reduce the dimensionality of data.
- It is designed to explain most of the variation in a data set that has m features with a new set of less than m manufactured features.

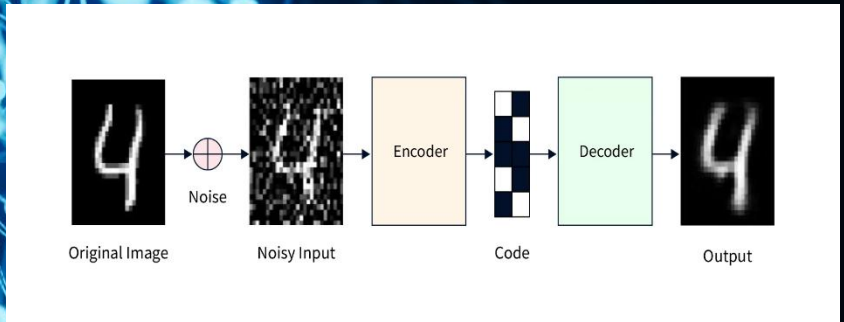


Autoencoder

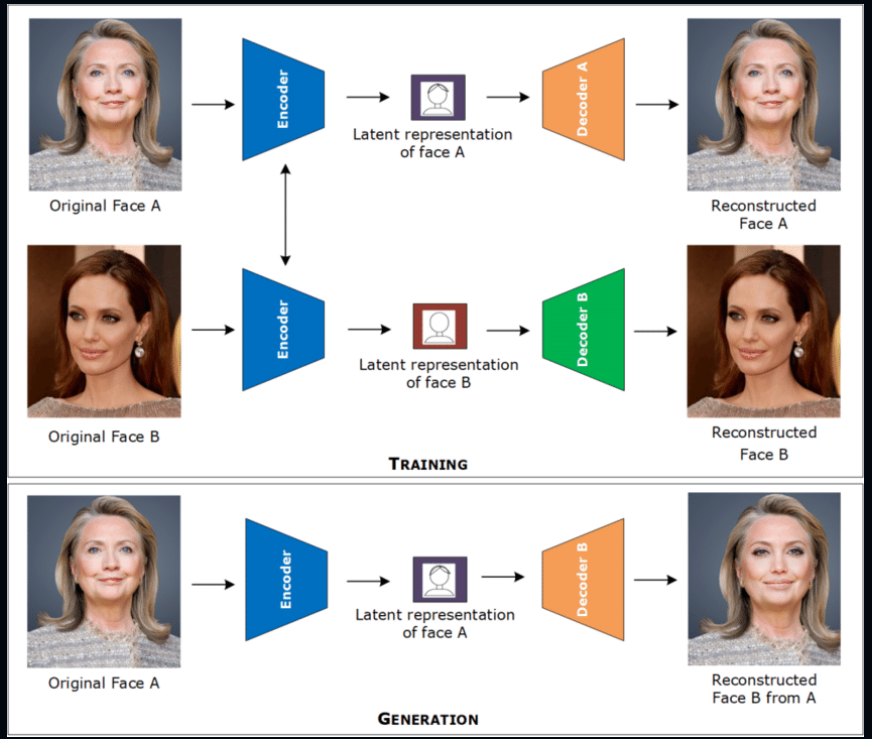


- Encoder: compresses the input data into latent space / embedded space
- Feature: the compressed input
- Decoder: Reconstructing input from the latent code

Some applications of Autoencoder



Denoising



Fake face

Black-Scholes-Merton Formula

Black-Scholes Formula for call options:

$$C = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

$$d_1 = \frac{ln(\frac{S_0}{K}) + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} \qquad d_2 = \frac{ln(\frac{S_0}{K}) + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

$C =$	call option price
$\Phi =$	CDF of the normal distribution
$S_0 =$	asset price
$K =$	strike price
$r =$	risk-free interest rate
$T =$	time to maturity
$\sigma =$	volatility of the asset

Monte Carlo

- Monte Carlo is used in corporate finance to model components of project cash flow, which are impacted by uncertainty.

$$\text{Option Value} = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} max(S_T^{(i)} - k, 0)$$

$N =$ number of simulations

$S_T^{(i)} =$ simulated stock price at maturity for the i-th path

Impact of increasing mini-batch iteration

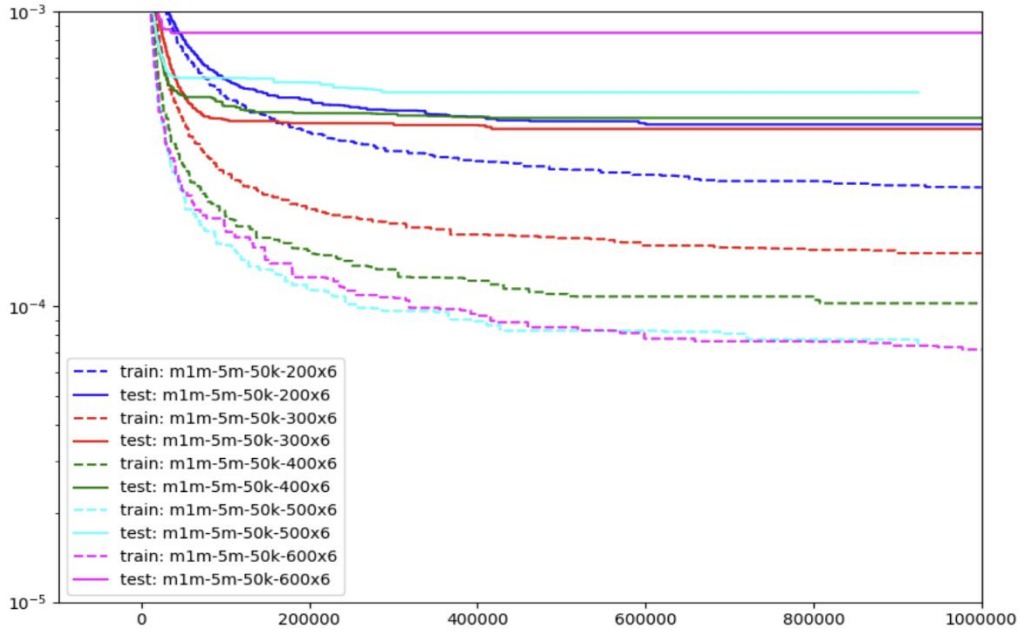


Figure 4: Minimum loss as a function of mini-batch iteration for five different models with 200, 300, 400, 500 and 600 nodes per layer.

Impact of training data size

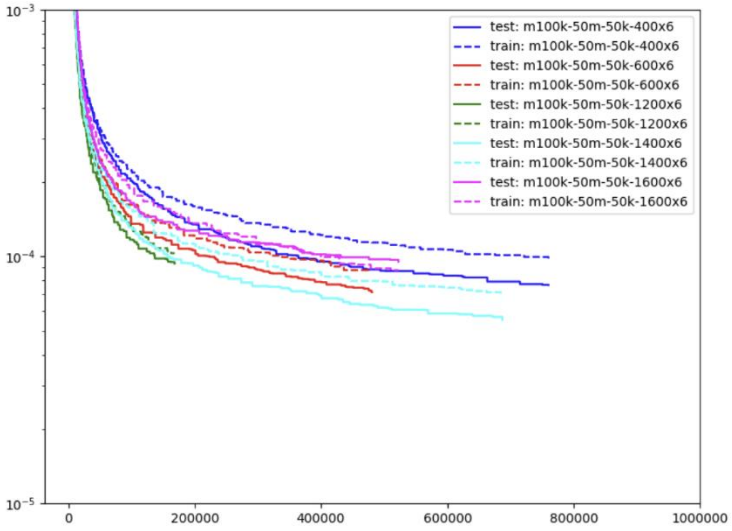


Figure 5: Minimum loss as a function of mini-batch iteration for five different models with 400, 600, 1200, 1400 and 1600 nodes per layer.

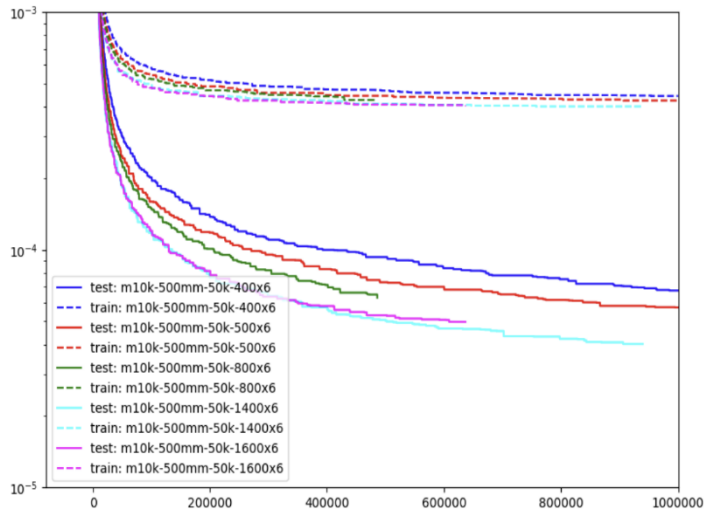


Figure 6: Minimum loss as a function of mini-batch iteration for five different models with 400, 600, 1200, 1400 and 1600 nodes per layer.